ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. «Πληροφοριακά Συστήματα & Υπηρεσίες»

Ειδίκευση: Προηγμένα Πληροφοριακά Συστήματα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Έλεγχος Λογισμικού»

Βασίλειος Τσακιράκης

Α.Μ.: ΜΕ1915

Επιβλέπουσα : Ανδριάνα Πρέντζα, Καθηγήτρια

ΠΕΙΡΑΙΑΣ

ΦΕΒΡΟΥΑΡΙΟΣ 2022

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Έλεγχος Λογισμικού»

Βασίλειος Τσακιράκης
Α.Μ.: ΜΕ1915

ΠΕΡΙΛΗΨΗ

Η παρούσα μεταπτυχιακή διπλωματική εργασία έχει θέμα τον «Έλεγχο Λογισμικού» και εκπονήθηκε στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς στα πλαίσια του μεταπτυχιακού προγράμματος «Πληροφοριακά Συστήματα & Υπηρεσίες» υπό την επίβλεψη της Καθηγήτριας κα. Ανδριάνας Πρέντζα.

Σκοπός αυτής της εργασίας είναι να δημιουργηθούν σενάρια ελέγχου λογισμικού για συγκεκριμένη ιστοσελίδα μέσω κατάλληλων μεθόδων και τεχνικών, στη συνέχεια να αυτοματοποιηθούν επιλέγοντας τα κατάλληλα εργαλεία, και τέλος να εξαχθούν χρήσιμα συμπεράσματα σχετικά με την αυτοματοποιημένη και μη-αυτοματοποιημένη διαδικασία.

Αρχικά γίνεται μια βιβλιογραφική ανασκόπηση για το τι είναι Κύκλος Ζωής Ανάπτυξης Λογισμικού και αναφέρονται κάποια από τα σημαντικότερα μοντέλα κύκλου ζωής λογισμικού έτσι ώστε ο αναγνώστης να κατανοήσει τον λόγο ύπαρξης και αναγκαιότητας του ελέγχου λογισμικού που ουσιαστικά είναι κομμάτι του Κύκλου Ζωής Ανάπτυξης Λογισμικού.

Στη συνέχεια, αναλύεται η έννοια του Ελέγχου Λογισμικού και οι 7 βασικές αρχές του και μετέπειτα παρουσιάζονται σε θεωρητικό επίπεδο οι διαδικασίες, οι φάσεις, τα επίπεδα, τα είδη, οι τεχνικές και οι κατηγορίες της προαναφερθείσας έννοιας.

Έπειτα εισάγεται η έννοια του αυτοματοποιημένου ελέγχου λογισμικού καθώς και η αναγκαιότητά της στα σύγχρονα και ταυτόχρονα πολύπλοκα λογισμικά που δημιουργούνται. Στο συγκεκριμένο κομμάτι της εργασίας, παρουσιάζονται επιπρόσθετα τα πλεονεκτήματα και τα μειονεκτήματα που έχει ως πρακτική και κάποια βασικά εργαλεία για διαφορετικά είδη αυτοματοποιημένων ελέγχων.

Τέλος, σύμφωνα με τη βιβλιογραφική ανασκόπηση που έχει πραγματοποιηθεί, δημιουργούνται και εκτελούνται σενάρια ελέγχου λογισμικού βασισμένα στην τεχνική του Μαύρου Κουτιού στην ιστοσελίδα SauceDemo που αποτελεί ένα διαδικτυακό κατάστημα με προϊόντα που έχει φτιαχτεί ειδικά για αυτοματοποιημένους ελέγχους λογισμικού. Τα σενάρια αυτά αυτοματοποιούνται μέσω του πλαισίου αυτοματοποιημένου ελέγχου Robot και της γλώσσας προγραμματισμού Python και τα συμπεράσματα που προκύπτουν καθώς και μελλοντικές βελτιώσεις που πιθανόν να υπάρξουν καταγράφονται αναλυτικά.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Έλεγχος Λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Κύκλος Ζωής Ανάπτυξης Λογισμικού, Έλεγχος Λογισμικού, Αυτοματοποιημένος Έλεγχος Λογισμικού, Τεχνική Μαύρου Κουτιού, Πλαίσιο Ελέγχου Robot

ABSTRACT

The subject of this thesis is «Software Testing» and was carried out at the Department of Digital Systems of University of Piraeus in the framework of the postgraduate program "Information Systems & Services" under the supervision of Professor Mrs. Andriana Prentza.

The purpose of this dissertation is to create software test cases for a specific website through appropriate methods and techniques, afterwards to automate them by selecting the appropriate tools and finally to draw useful conclusion about the automated and non-automated process.

Initially a literature review of what is Software Development Life Cycle is done and some of the most important software development life cycle models are mentioned so the reader understands the reason for the existence and necessity of software testing which is essentially part of the Software Development Life Cycle.

Subsequently, the concept of Software Testing and its 7 basic principles are analyzed and then the processes, phases, levels, types, techniques and categories of the aforementioned concept are presented in a theoretical background.

Thereafter the concept of test automation as well as its necessity in the modern and complex software systems is introduced. In this part of the thesis, the advantages and disadvantages of test automation as well as some basic tools for different types of automation testing are presented.

Finally, according to all the literature review that has been done, test cases are created and executed based on Black Box technique on the SauceDemo website which is an online store with products that have been made especially for automation testing. These test cases are automated through the Robot framework and the Python programming language and the conclusion that emerge as well as future improvements that may occur are recorded in detail.

SUBJECT AREA: Software Testing

KEYWORDS: Software Development Life Cycle, Software Testing, Test Automation, Black Box Technique, Robot Framework

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Διπλωματικής Εργασίας, θα ήθελα να ευχαριστήσω θερμά την κα. Ανδριάνα Πρέντζα για τη συνεργασία και την πολύτιμη συμβολή της. Από την πρώτη στιγμή με εμπιστεύτηκε και με τις παρατηρήσεις και συμβουλές της με βοήθησε στην βελτίωση της εργασίας και τελικά στην ολοκλήρωση της.

Εν συνέχεια, θα ήθελα να ευχαριστήσω την υποψήφια διδάκτωρ του τμήματος Ψηφιακών Συστημάτων Σιαπέρα Μαρία για την καθοδήγηση της και τις πολύτιμες συμβουλές της κατά τη διάρκεια υλοποίησης της εργασίας. Η άμεση και συχνή επικοινωνία με βοήθησε να καταλάβω καλύτερα τον τρόπο που πρέπει να δουλέψω για να πετύχω το καλύτερο δυνατό αποτέλεσμα.

Ένα μεγάλο ευχαριστώ στην κοπέλα μου που μου συμπαραστάθηκε όλο αυτόν τον καιρό και είχε την υπομονή να με βοηθήσει σε ότι χρειάστηκα. Η συμβολή της ήταν και είναι ανεκτίμητη.

Τέλος, δεν θα μπορούσα να μην αναφερθώ στους γονείς μου και να τους ευχαριστήσω από καρδιάς για την εμπιστοσύνη και την στήριξη που μου δείχνουν όλα αυτά τα χρόνια σε ότι απόφαση και αν πάρω στη ζωή μου. Χωρίς αυτούς τίποτα δεν θα ήταν εφικτό στην μέχρι τώρα επαγγελματική αλλά και προσωπική μου πορεία. Η εργασία αυτή αφιερώνεται σε αυτούς.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο και στόχοι της Μεταπτυχιακής Διπλωματικής Εργασίας

Τις τελευταίες δύο δεκαετίες η τεχνολογία αποτελεί μια από τις πιο ραγδαία αναπτυσσόμενες επιστήμες. Καθημερινά ξοδεύονται υπέρογκα ποσά για τη δημιουργία λογισμικών σε διαφορετικούς τομείς όπως για παράδειγμα στην υγεία , στη διασκέδαση αλλά και στην ενημέρωση. Επιπρόσθετα οι χρήστες είναι εξοικειωμένοι με τις νέες τεχνολογίες οπότε αυτόματα και η ποιότητα που περιμένουν από τα λογισμικά είναι υψηλή. Όλα τα παραπάνω δημιουργούν μεγάλο ανταγωνισμό, οπότε τα σύγχρονα λογισμικά είναι ιδιαίτερα πολύπλοκα και δύσκολα στη δημιουργία τους. Αυτό δημιουργεί την ανάγκη να υπάρχει συχνός αλλά και ιδιαίτερα προσεκτικός και σωστός Έλεγχος Λογισμικού. Ο όρος «Έλεγχος Λογισμικού» που αποτελεί και το αντικείμενο αυτής της εργασίας είναι ένας ευρύς όρος που καλύπτει ένα μεγάλο φάσμα δραστηριοτήτων. Έλεγχος Λογισμικού μπορεί να γίνει σε ένα μέρος του κώδικα, στο γραφικό περιβάλλον του χρήστη, στη βάση δεδομένων του λογισμικού, στην απόδοση του συστήματος αλλά και γενικά σε όλες τις περιοχές που σχετίζονται με το υπό-δημιουργία λογισμικό. Όλα τα παραπάνω θα αναλυθούν εκτενώς στη συγκεκριμένη εργασία.

Επιπρόσθετα είναι κατανοητό πως όλοι αυτοί οι πολύπλοκοι έλεγχοι λόγω και της πολυπλοκότητάς των λογισμικών δεν μπορούν να επιτευχθούν επαναληπτικά και συστηματικά με τον παραδοσιακό τρόπο ελέγχου λογισμικού που είναι ο «χειρωνακτικός». Ως «χειρωνακτικό» έλεγχο θεωρούμε τον έλεγχο που πραγματοποιείται από έναν χρήστη αλληλοεπιδρώντας με την εφαρμογή χωρίς τη χρήση εργαλείων. Έτσι έχει προκύψει ένας νέος τρόπος ελέγχου λογισμικού και αυτός είναι ο αυτοματοποιημένος. Μέσω συγκεκριμένων τεχνολογιών και εργαλείων δημιουργούνται οι απαραίτητοι αυτοματοποιημένοι έλεγχοι και έτσι ο ελεγκτής δεν χρειάζεται να επαναλαμβάνει τους ίδιους ελέγχους καθημερινά με τον «χειρωνακτικό» τρόπο. Αυτό κερδίζει χρόνο και χρήμα στις εταιρίες αλλά και στους ανθρώπους που εκτελούν τους ελέγχους, και ταυτόχρονα βελτιώνει την ποιότητα των λογισμικών που παράγονται.

Στόχος αυτής της εργασίας είναι η μελέτη και η κατανόηση όλων των παραπάνω εννοιών και στη συνέχεια η πρακτική εφαρμογή τους σε συγκεκριμένη μελέτη περίπτωσης. Αναλυτικότερα, επιλέγοντας μια ιστοσελίδα θα πραγματοποιηθεί ανάλυση και δημιουργία σεναρίων ελέγχου. Στη συνέχεια επιλέγοντας τα κατάλληλα εργαλεία θα δημιουργηθούν και θα εκτελεστούν οι αυτοματοποιημένοι έλεγχοι λογισμικού. Τα αποτελέσματα αυτών των ελέγχων και τα συμπεράσματα που θα προκύψουν θα καταγραφούν και θα αναλυθούν στο τέλος αυτής της εργασίας.

1.2 Δομή της Μεταπτυχιακής Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία χωρίζεται σε έξι κεφάλαια και αυτά είναι τα παρακάτω:

Κεφάλαιο 1: Στο κεφάλαιο αυτό αναφέρεται το αντικείμενο και οι στόχοι της διπλωματικής εργασίας. Επίσης παρουσιάζεται η δομή της εργασίας ανάλογα με τα κεφάλαια που δημιουργήθηκαν.

Κεφάλαιο 2: Στο κεφάλαιο αυτό παρουσιάζεται η έννοια του Κύκλου Ζωής Ανάπτυξης Λογισμικού, οι φάσεις που περιλαμβάνει καθώς και κάποια από τα πιο διαδεδομένα μοντέλα που υπάρχουν αυτή τη στιγμή.

Κεφάλαιο 3: Στο κεφάλαιο τρία που αποτελεί και τον θεωρητικό «κορμό» αυτής της εργασίας παρουσιάζεται και αναλύεται η έννοια του Ελέγχου Λογισμικού, οι διαδικασίες, οι φάσεις, τα είδη και τα επίπεδα που έχει και επιπρόσθετα παρουσιάζονται οι βασικές κατηγορίες και τεχνικές ελέγχου που χρησιμοποιούνται συχνά.

Κεφάλαιο 4: Στο κεφάλαιο αυτό εισάγεται η έννοια του αυτοματοποιημένου ελέγχου λογισμικού. Πραγματοποιείται μια σύγκριση αναλύοντας τα πλεονεκτήματα και τα μειονεκτήματα που έχει σε σχέση με το «χειρωνακτικό» έλεγχο λογισμικού και τέλος παρουσιάζονται τα πιο διαδεδομένα εργαλεία αυτοματοποιημένου ελέγχου λογισμικού για διαφορετικούς ελέγχους, όπως για παράδειγμα την απόδοση του συστήματος ή τη συνολική αποδοχή του από τον πελάτη.

Κεφάλαιο 5: Το κεφάλαιο αυτό αποτελεί το πρακτικό κομμάτι της παρούσας διπλωματικής εργασίας. Αρχικά γίνεται μια σύντομη περιγραφή της ιστοσελίδας που θα πραγματοποιηθεί η ανάλυση και στη συνέχεια ο αυτοματοποιημένος έλεγχος λογισμικού. Στη συνέχεια δημιουργούνται τα σενάρια ελέγχου που θα αυτοματοποιηθούν. Μετέπειτα επιλέγονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιηθήκαν στα πλαίσια του αυτοματοποιημένου ελέγχου και τέλος εκτελούνται οι έλεγχοι και παρουσιάζονται τα αποτελέσματα που προκύπτουν.

Κεφάλαιο 6: Το συγκεκριμένο κεφάλαιο είναι το τελευταίο κεφάλαιο της εργασίας και καταγράφονται τα συμπεράσματα και οι μελλοντικές βελτιώσεις που είναι εφικτό να γίνουν στα σενάρια ελέγχου που δημιουργήθηκαν καθώς και στον τρόπο που αυτοματοποιήθηκαν.

2. ΚΥΚΛΟΣ ΖΩΗΣ ΛΟΓΙΣΜΙΚΟΥ

Στο κεφάλαιο αυτό θα υπάρξει μια θεωρητική προσέγγιση του Κύκλου Ζωής Ανάπτυξης Λογισμικού (Software Development Life Cycle – SDLC) καθώς και των πιο διαδεδομένων μοντέλων κύκλου ζωής λογισμικού. Ο σκοπός αυτού του κεφαλαίου είναι η κατανόηση των βημάτων δημιουργίας ενός προγράμματος και μέσω αυτών των βημάτων να αναδειχτεί η σημαντικότητα και η αναγκαιότητα του ελέγχου λογισμικού σε μια εφαρμογή.

2.1 Φάσεις Κύκλου Ζωής Λογισμικού

Ο κύκλος ζωής ανάπτυξης ενός συστήματος ορίζεται ως μια σειρά από στάδια ή δραστηριότητες όπου κάθε εμπλεκόμενη ομάδα θα πρέπει να ακολουθήσει έτσι ώστε να επιτευχθεί ο κοινός σκοπός δηλαδή η δημιουργία ενός προγράμματος που να ικανοποιεί τις απαιτήσεις του πελάτη [1]. Κάθε φάση του κύκλου ζωής έχει συγκεκριμένες διαδικασίες που πρέπει να ακολουθηθούν και σε συγκεκριμένο χρονικό διάστημα. Συνήθως αυτές οι φάσεις χωρίζονται σε 5 διαφορετικά στάδια. Αυτά είναι, όπως φαίνονται και στην εικόνα 1, η ανάλυση και συλλογή απαιτήσεων, η σχεδίαση, η υλοποίηση, ο έλεγχος λογισμικού που είναι και το θέμα της συγκεκριμένης εργασίας και τελευταίο η εφαρμογή του λογισμικού στο περιβάλλον του πελάτη και η συντήρησή του [1] [2] [3].

Εικόνα 1: **Κύκλος ζωής ανάπτυξης λογισμικού [4]**

Ανάλυση και συλλογή απαιτήσεων (Requirement Analysis)

Η πρώτη φάση που θα αναλύσουμε αποτελεί ουσιαστικά και την αρχή ενός έργου, ωστόσο είναι ιδιαίτερα σημαντική γιατί αν δεν υπάρξει σωστή ανάλυση, τότε όλο το έργο δεν θα έχει το επιθυμητό αποτέλεσμα. Αναλυτικότερα σε αυτή τη φάση καταγράφονται με λεπτομέρεια οι απαιτήσεις και οι προσδοκίες του πελάτη από την ομάδα ανάλυσης. Αυτή η διαδικασία απαιτεί συνεχή επικοινωνία μεταξύ πελάτη και της ομάδα της ανάλυσης. Επίσης υπάρχει μια πρώτη εκτίμηση του κόστους του έργου ανάλογα με τις απαιτήσεις του πελάτη έτσι ώστε να δουν όλοι

οι εμπλεκόμενοι φορείς αν μπορεί να επιτευχθεί το έργο σε συγκεκριμένα οικονομικά πλαίσια που έχουν οριστεί και από τον πελάτη αλλά και από την εκάστοτε εταιρία παραγωγής λογισμικού.

Κάποιες βασικές τεχνικές που χρησιμοποιούνται για τη συλλογή των απαιτήσεων είναι:

- Έρευνα στην αγορά και συγκεκριμένα τι προσφέρει ο ανταγωνισμός.
- Συνεχής επικοινωνία πελάτη – αναλυτή με συνεχόμενη ανατροφοδότηση.
- Παρουσιάσεις στον πελάτη αρχείων με το πώς θα μοιάζει το τελικό προϊόν.

Συμπερασματικά, όπως ήδη αναφέρθηκε, παρατηρούμε πως η ανάλυση και η συλλογή απαιτήσεων θέλει ιδιαίτερη προσοχή από τα ενδιαφερόμενα μέλη. Σε περίπτωση που η ανάλυση έχει γίνει σωστά, τότε όλες οι εμπλεκόμενες ομάδες θα έχουν ξεκάθαρη εικόνα του τι πρέπει να κάνουν για την επίτευξη του έργου. Σε αντίθετη περίπτωση, υπάρχουν κενά μεταξύ των ομάδων, οπότε και το τελικό προϊόν δεν θα είναι το επιθυμητό.

Σχεδίαση (Design)

Στη φάση της σχεδίασης, σχεδιάζεται το σύστημα τεχνικά σύμφωνα με τις απαιτήσεις που έχουν δημιουργηθεί από την προηγούμενη φάση. Οι απαιτήσεις του πελάτη μετατρέπονται σε τεχνικές απαιτήσεις και σχεδιάζεται η αρχιτεκτονική του συστήματος, περιγράφονται οι βάσεις δεδομένων και σχεδιάζεται ποια προγράμματα θα γραφτούν και τι πρέπει να κάνουν. Αναλυτικότερα σε αυτή τη φάση καθορίζονται όλα τα τεχνικά δεδομένα και εργαλεία που απαιτούνται για την υλοποίηση του έργου, όπως για παράδειγμα η αρχιτεκτονική της βάσης δεδομένων, οι διαδικασίες ασφάλειας αλλά ακόμα και φυσικές απαιτήσεις (π.χ. ένα τερματικό μηχάνημα). Σε αυτή τη φάση είναι σημαντικό να αναλυθούν όλοι οι πιθανοί κίνδυνοι που μπορεί να έχει ένα σύστημα (π.χ. σημεία στο κώδικα που είναι προσβάσιμα από όλους), όλες οι λειτουργικές προδιαγραφές (π.χ. σε τι ζώνη ώρας (time zone) θα λειτουργεί το σύστημα) καθώς και μη λειτουργικές προδιαγραφές (π.χ. η απόδοση συστήματος).

Υλοποίηση (Implementation)

Σε αυτή τη φάση ξεκινάει η ανάπτυξη του κώδικα από την ομάδα ανάπτυξης λογισμικού. Οι προγραμματιστές υλοποιούν σύμφωνα με την ανάλυση και το σχεδιασμό που έχει γίνει στις προηγούμενες φάσεις. Σε αυτή τη φάση υπάρχει συνεχής επικοινωνία μεταξύ των προγραμματιστών και της ομάδας ανάλυσης έτσι ώστε να τηρηθούν και να υλοποιηθούν όλα όσα έχουν συμφωνηθεί με τον πελάτη. Κατά τη διάρκεια αυτής της φάσης δημιουργούνται και κάποιες μικρές παρουσιάσεις έτσι ώστε ο πελάτης να δει σε τι κατάσταση βρίσκεται το προϊόν και να ενημερώσει αν επιθυμεί αλλαγές ή όχι και με τι κόστος (χρηματικό αλλά και χρονικό) για τη συνολική πορεία του έργου. Αυτή η φάση συνήθως είναι η μεγαλύτερη σε διάρκεια στον κύκλο ζωής ανάπτυξης ενός προγράμματος. Με την ολοκλήρωση της φάσης αυτής, το προϊόν που έχει δημιουργηθεί παραδίδεται στην ομάδα ελέγχου.

Έλεγχος Λογισμικού (Testing)

Ο έλεγχος λογισμικού αποτελεί την προτελευταία φάση του κύκλου ζωής ανάπτυξης λογισμικού. Αποτελεί το θέμα της συγκεκριμένης διπλωματικής εργασίας και θα αναλυθεί εκτενώς σε επόμενα κεφάλαια. Μια πρώτη προσέγγιση είναι ότι η ομάδα προγραμματιστών εγκαθιστά το λογισμικό στο περιβάλλον ελέγχου (test environment). Στη συνέχεια η ομάδα ελέγχου ξεκινάει να κάνει δοκιμές στο σύστημα για να διασφαλίσει ότι η ποιότητα είναι η επιθυμητή και ότι τηρούνται όλα όσα έχουν συμφωνηθεί με το πελάτη. Σε περίπτωση σφαλμάτων, ο δοκιμαστής (tester)[1] είναι υποχρεωμένος να επικοινωνήσει το πρόβλημα στην ομάδα των προγραμματιστών με σκοπό την επίλυσή του. Στη φάση αυτή δημιουργούνται συχνά και νέες απαιτήσεις λόγω διαφορών μεταξύ της ανάλυσης και της υλοποίησης, οπότε πρέπει να γίνει εκ νέου ένα είδος ανάλυσης αλλά και ανάπτυξης κώδικα. Με την ολοκλήρωση της φάσης αυτής πιστοποιείται πως η εφαρμογή πληροί όλα τα συμφωνηθέντα και απομένει η ένταξη του συστήματος στο περιβάλλον του πελάτη για να το ελέγξει και αυτός και έπειτα να ξεκινήσει η χρήση του στην αγορά.

Εφαρμογή και Συντήρηση (Deployment and Maintenance)

Η τελευταία φάση του κύκλου ζωής αποτελείται από δύο υπό-φάσεις. Η πρώτη είναι η εγκατάσταση του λογισμικού σε ένα περιβάλλον που θα είναι μια «αντιγραφή» του περιβάλλοντος που θα βγει στην αγορά. Σε αυτό το περιβάλλον ο πελάτης θα κάνει τις δικές του δοκιμές για να αξιολογήσει και αυτός αν το προϊόν είναι έτοιμο να βγει στην αγορά (User Acceptance Testing – UAT). Στη συνέχεια, και αν όλα λειτουργούν όπως επιθυμεί, δίνει την έγκρισή του και το προϊόν εγκαθίσταται και στο τελικό παραγωγικό περιβάλλον. Η δεύτερη υπό-φάση ουσιαστικά αποτελεί τη συντήρηση του προϊόντος στο παραγωγικό περιβάλλον. Είναι σύνηθες, ειδικά τις πρώτες μέρες ενός καινούριου προϊόντος, να υπάρχουν κάποια σφάλματα οπότε οι προγραμματιστές σε συνεννόηση με τον πελάτη οφείλουν να τα φτιάξουν το συντομότερο δυνατό και να συντηρούν το περιβάλλον έτσι ώστε να είναι πάντα λειτουργικό. Στη φάση αυτή που αποτελεί και το τελικό στάδιο του κύκλου ζωής ανάπτυξης ενός προγράμματος μπορεί να ζητηθούν από τον πελάτη επιπλέον βελτιώσεις γιατί ο ανταγωνισμός είναι υψηλός ή γιατί αποφάσισε να βελτιώσει λειτουργίες που ήδη υπάρχουν στο σύστημα. Σε μια τέτοια περίπτωση επαναλαμβάνονται όλες οι φάσεις του κύκλου ζωής που αναφέρθηκαν παραπάνω για να καταλήξουμε στο τελικό βελτιωμένο προϊόν. Οι αλλαγές αυτές ονομάζονται “αιτήματα αλλαγής” (Change Requests - CR).

2.2 Μοντέλα Κύκλου Ζωής Λογισμικού

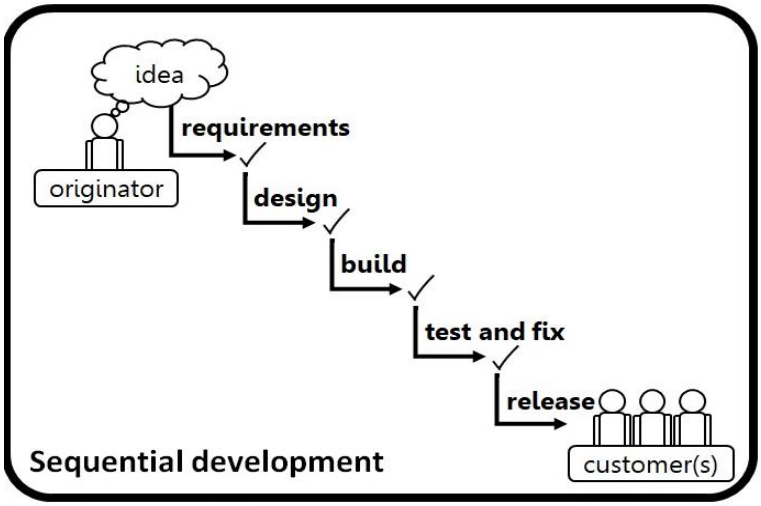
Σκοπός της υπό-ενότητας αυτής είναι ο ορισμός της έννοιας του μοντέλου κύκλου ζωής λογισμικού καθώς και η γνωριμία με τα πιο διαδεδομένα μοντέλα. Είναι σημαντικό να κατανοήσουμε ότι υπάρχουν διαφορετικά μοντέλα και σε κάθε μοντέλο ο τρόπος και κυρίως η

[1] Για την παρούσα εργασία αντί να χρησιμοποιούμε την ελληνική λέξη “δοκιμαστής” θα χρησιμοποιούμε την αγγλική λέξη “tester” που είναι ευρέως γνωστή και επιστημονικά αποδεκτή.

σειρά και ο χρόνος που γίνεται ο έλεγχος λογισμικού διαφέρει. Το τελευταίο θα αναλυθεί περαιτέρω σε επόμενο κεφάλαιο που θα αναλύσουμε τον έλεγχο λογισμικού .

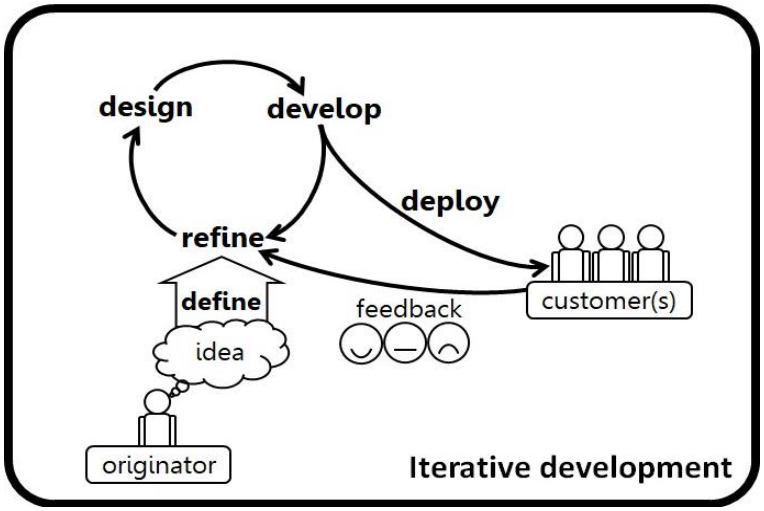
Ένα μοντέλο κύκλου ζωής λογισμικού περιγράφει τις φάσεις κύκλου ζωής λογισμικού όπως αναφέρθηκαν στην προηγουμένη υπό-ενότητα και τις ενέργειες που γίνονται σε κάθε φάση καθώς και τη σειρά που πρέπει να γίνουν. Ουσιαστικά το κάθε μοντέλο καθορίζει το πώς πρέπει να εκτελούνται οι διαδικασίες που έχουν προαναφερθεί, δηλαδή ποιες ενέργειες πρέπει να γίνουν, τι πρέπει να γίνει σε κάθε ενέργεια και ποιο είναι το τελικό παραδοτέο.

Σύμφωνα με το International Software Testing Qualifications Board (ISTQB) [1] υπάρχουν 2 κατηγορίες μοντέλων. Αυτά είναι τα ακολουθιακά μοντέλα και τα επαναληπτικά μοντέλα [5] [6]. Στα ακολουθιακά μοντέλα, όπως φαίνεται και στην εικόνα 2, κάθε φάση του κύκλου ζωής είναι διακριτή άρα και η ανάπτυξη του προγράμματος γίνεται σε συγκεκριμένες φάσεις. Έτσι υπάρχει αυστηρή σειρά στον τρόπο δημιουργίας ενός προγράμματος. Για παράδειγμα πρώτα γίνεται η ανάλυση, μετά η υλοποίηση και τέλος ο έλεγχος λογισμικού σε 3 διακριτές φάσεις. Για να ξεκινήσει μια φάση πρέπει να ολοκληρωθεί πλήρως η προηγούμενη. Τέτοια μοντέλα είναι το μοντέλο καταρράκτη και το μοντέλο V που θα αναλυθούν παρακάτω.



Εικόνα 2: Ακολουθιακά Μοντέλα [5]

Αντίθετα στα επαναληπτικά μοντέλα όπως φαίνεται και στην εικόνα 3 η ανάπτυξη του προγράμματος γίνεται σε μικρές φάσης. Σε κάθε τέτοια φάση όλες οι ομάδες (π.χ. αναλυτές, προγραμματιστές, testers, πελάτης) αλληλοεπιδρούν με σκοπό τη δημιουργία ενός μικρού παραδοτέου. Τέτοια μοντέλα είναι το μοντέλο Agile και το Σπειροειδές όπου θα αναλυθούν παρακάτω [5] [6].



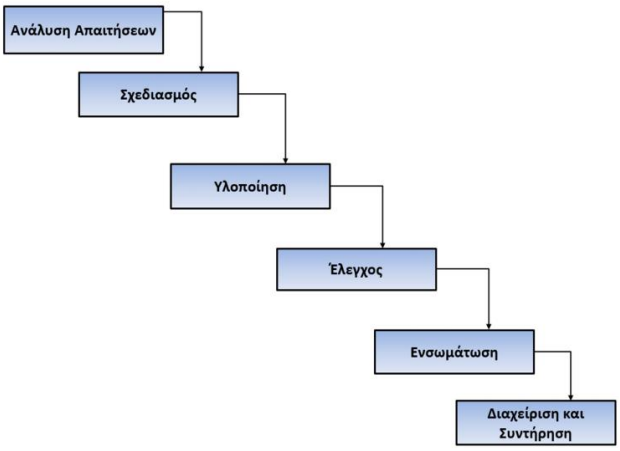
Εικόνα 3: Επαναληπτικά Μοντέλα [5]

2.2.1 Μοντέλο Καταρράκτη (Waterfall)

Ένα από τα πιο διαδεδομένα μοντέλα είναι το μοντέλο του καταρράκτη το οποίο όπως προαναφέρθηκε ανήκει στην κατηγορία των ακολουθιακών μοντέλων και περιλαμβάνει 6 διακριτές φάσεις [1] [7]. Η βασική ιδέα του μοντέλου είναι ότι κατά την ανάπτυξη της εφαρμογής, κάθε φάση είναι ανεξάρτητη και για να προχωρήσουμε από μία φάση σε άλλη πρέπει να έχει προηγηθεί η ολοκλήρωση της προηγούμενης φάσης. Με την ολοκλήρωση της κάθε φάσης υπάρχουν και τα παραδοτέα που μεταβιβάζονται στην αμέσως επόμενη φάση.

Οι διακριτές φάσεις του μοντέλου όπως φαίνονται και στην εικόνα 4 είναι:

- **Ανάλυση Απαιτήσεων:** Στη φάση αυτή συλλέγονται οι απαιτήσεις του πελάτη και αναλύονται όλες οι λειτουργίες του συστήματος.
- **Σχεδιασμός:** Με την ολοκλήρωση της προηγούμενης φάσης ξεκινάει η σχεδίαση του συστήματος και καθορίζεται η αρχιτεκτονική του καθώς και το λογισμικό του συστήματος.
- **Υλοποίηση:** Οι προγραμματιστές παράγουν το λογισμικό σύμφωνα με το σχεδιασμό που έγινε στην προηγουμένη φάση.
- **Έλεγχος:** Η ομάδα ελέγχου παραλαμβάνει το λογισμικό και ελέγχει αν τηρούνται όλες οι προδιαγραφές που έχουν οριστεί από τον πελάτη και το τμήμα ανάλυσης και σχεδιασμού. Σε περίπτωση σφαλμάτων πρέπει με την βοήθεια και τη συνεργασία της ομάδα των προγραμματιστών να επιλυθούν και να επανελεγχθούν.
- **Ενσωμάτωση:** Με τον επιτυχή έλεγχο από την ομάδα ελέγχου η χρήση του λογισμικού γίνεται διαθέσιμη στον πελάτη για να το χρησιμοποιήσει.
- **Διαχείριση και Συντήρηση:** Στην τελευταία φάση γίνεται η συνεχής συντήρηση του λογισμικού ώστε να συνεχίσει να λειτουργεί χωρίς προβλήματα και σύμφωνα με τις προδιαγραφές που έχουν τεθεί.



Εικόνα 4: Μοντέλο Καταρράκτη [8]

Μελετώντας το μοντέλο μπορούμε να καταλάβουμε ότι έχει και πλεονεκτήματα και μειονεκτήματα. Παρατηρούμε λοιπόν ότι είναι αρκετά απλό και εύκολο στη χρήση του καθώς ουσιαστικά ακολουθεί τον κύκλο ζωής ενός λογισμικού με αυστηρή σειρά. Επίσης κάθε φάση με το να είναι διακριτή, προσφέρει ευκολία στη διαχείριση των παραδοτέων κυρίως σε μικρά έργα όπου οι απαιτήσεις έχουν μικρότερο εύρος και είναι συγκεκριμένες. Αντίθετα σε μεγάλα έργα που συνεχώς μεταβάλλονται το μοντέλο αυτό παρουσιάζει προβληματική συμπεριφορά καθώς οποιαδήποτε αλλαγή στις απαιτήσεις μπορεί να προκαλέσει σύγχυση στις ομάδες και πρόβλημα στο λογισμικό. Επιπρόσθετα υπάρχει μεγάλη αβεβαιότητα για το τελικό παραδοτέο καθώς είναι δύσκολο να υπάρξει ανατροφοδότηση μεταξύ των ομάδων.

2.2.2 V-Model

Ένα ακόμη σημαντικό μοντέλο είναι το V-Model το οποίο αποτελεί μια επέκταση του μοντέλου του καταρράκτη. Το μοντέλο αυτό παρουσιάζεται και ως το μοντέλο επαλήθευσης και επικύρωσης (Verification and Validation) [1] [9]. Στο μοντέλο αυτό, σε κάθε φάση της διαδικασίας παραγωγής λογισμικού υπάρχει μια άμεση σχέση φάσης ελέγχου. Όπως και το μοντέλο καταρράκτη έτσι και αυτό, αποτελεί ένα ακολουθιακό μοντέλο όπου κάθε φάση ξεκινά με την ολοκλήρωση της προηγούμενης.

Οι φάσεις επαλήθευσης (Verification Phases) είναι οι παρακάτω:

Ανάλυση Απαιτήσεων (Requirement Analysis): Αυτή είναι η πρώτη φάση του κύκλου ανάπτυξης λογισμικού. Σε αυτή τη φάση υπάρχει συνεχής επικοινωνία με τον πελάτη με στόχο να κατανοηθεί τι ακριβώς επιθυμεί ο πελάτης. Επιπρόσθετα γίνεται και ο σχεδιασμός δοκιμής ελέγχου (Acceptance Test Design) καθώς οι απαιτήσεις του πελάτη μπορούν να χρησιμοποιηθούν σαν δεδομένα για τη δοκιμή αποδοχής (Acceptance Testing).

Σχεδίαση Συστήματος (System Design): Με την ολοκλήρωση της προηγούμενης φάσης πρέπει να υπάρχουν λεπτομερώς οι απαιτήσεις του υπό-δημιουργία προϊόντος. Στη συνέχεια στην παρούσα φάση γίνεται η σχεδίαση του συστήματος που περιλαμβάνει τόσο το hardware του συστήματος όσο και την ανάλυση των απαιτήσεων του πελάτη σε τεχνικό επίπεδο. Αν

κάποια από τις απαιτήσεις δεν είναι εφικτό να υλοποιηθεί, ο πελάτης ενημερώνεται για το ζήτημα. Σύμφωνα με το σχεδιασμό του συστήματος δημιουργείται και το πλάνο ελέγχου συστήματος (System Test Plan). Αυτό έχει σαν αποτέλεσμα να υπάρχει περισσότερο χρόνος για έλεγχο στις μετέπειτα φάσεις καθώς έχει γίνει ήδη η προεργασία για το τι θα πρέπει να ελέγξει ο tester.

Αρχιτεκτονική Σχεδίαση (Architecture Design): Η φάση αυτή αφορά την αρχιτεκτονική σχεδίαση του συστήματος. Είναι γνωστή και ως φάση σχεδιασμού υψηλού επιπέδου (High Level Design – HL) και οι ομάδες που την εκτελούν προσπαθούν να βρουν τη σωστή τεχνική προσέγγιση για το υπό-δημιουργία σύστημα. Επιπρόσθετα οι ομάδες σε αυτήν τη φάση προσπαθούν να αναλύσουν τις απαιτήσεις σε μικρότερες ενότητες όπου κάθε ενότητα θα έχει διαφορετική λειτουργία. Σε αυτή τη φάση δημιουργείται και το σχέδιο δοκιμής της ενσωμάτωσης (Integration Test Design) όπου θα δοκιμαστεί η επικοινωνία της κάθε ενότητας με μια άλλη.

Σχεδίαση ενοτήτων (Module Design): Σε αυτή τη φάση που είναι γνωστή και ως φάση σχεδιασμού χαμηλού επιπέδου (Low Level Design – LLD) σχεδιάζεται η λογική της κάθε ενότητας του συστήματος. Είναι πολύ σημαντικό κάθε ενότητα να μπορεί να επικοινωνεί με άλλες ενότητες του συστήματος αλλιώς το τελικό παραδοτέο δεν θα είναι λειτουργικό. Στη φάση αυτή δημιουργείται και το σχέδιο ελέγχου μονάδας (Unit Test Design) μέσω του οποίου σχεδιάζεται το τι θα ελεγχθεί για κάθε ενότητα ξεχωριστά.

Φάση υλοποίησης (Coding Phase): Η υλοποίηση του έργου μέσω κώδικα πραγματοποιείται σε αυτή τη φάση. Η επιλογή της γλώσσας προγραμματισμού επιλέγεται σύμφωνα με τις απαιτήσεις του συστήματος αλλά και την αρχιτεκτονική σχεδίαση του συστήματος. Για να ολοκληρωθεί αυτή η φάση απαιτείται πολύ χρόνος και ο κώδικας ελέγχεται και αξιολογείται συνεχώς με στόχο την καλύτερη επίδοση και το μικρότερο δυνατό αριθμό σφαλμάτων.

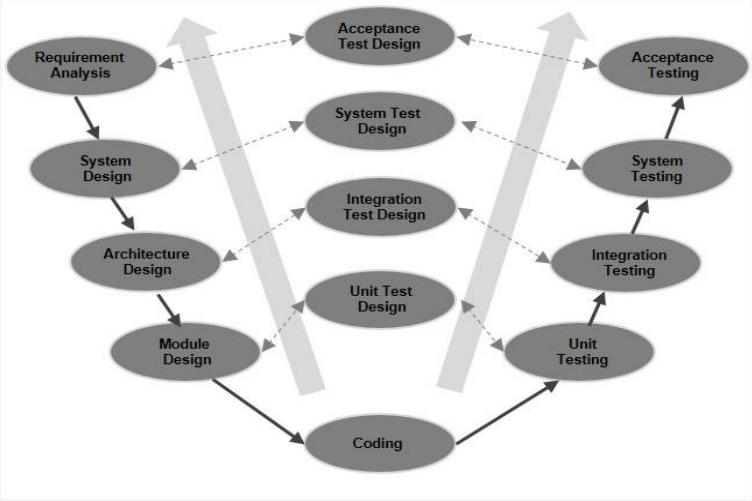
Αντίστοιχα υπάρχουν και οι φάσεις επικύρωσης (Validation Phases). Αυτές είναι :

Έλεγχος Μονάδων (Unit Testing): Σύμφωνα με το σχέδιο ελέγχου μονάδας που δημιουργήθηκε στη αντίστοιχη φάση σχεδιασμού ενότητας εκτελούνται οι έλεγχοι μονάδας. Αυτοί οι έλεγχοι γίνονται σε επίπεδο κώδικα και βοηθούν στο να μην υπάρχουν πολλά προβλήματα από την αρχή του έργου. Είναι σημαντικό να αναφερθεί ότι ο έλεγχος μονάδας δεν μπορεί να εξαλείψει τα προβλήματα του συστήματος.

Έλεγχος Ενσωμάτωσης (Integration Testing): Ο έλεγχος αυτός σχετίζεται με τη φάση της αρχιτεκτονικής σχεδίασης. Οι έλεγχοι αυτοί πραγματοποιούνται με σκοπό να ελεγχθεί η επικοινωνία και η συνύπαρξη των μονάδων του συστήματος.

Έλεγχος Συστήματος (System Testing): Όπως ήδη αναφέρθηκε, ο έλεγχος αυτός σχετίζεται με τη φάση της σχεδίασης του συστήματος. Οι έλεγχοι εκτελούνται με σκοπό να ελεγχθεί η πλήρης λειτουργικότητα του συστήματος.

Έλεγχος Αποδοχής (Acceptance Testing): Ο έλεγχος απόδοσης σχετίζεται με τη φάση ανάλυσης των απαιτήσεων και περιλαμβάνει τη δοκιμή του προϊόντας σε περιβάλλον χρήστη. Μέσω αυτού του ελέγχου μπορούν να βρεθούν και προβλήματα μη λειτουργικά όπως για παράδειγμα χαμηλή απόδοση του συστήματος σε πραγματικό περιβάλλον χρήστη.



Εικόνα 5: V-Model [10]

Συνοψίζοντας, όπως κάθε μοντέλο έτσι και αυτό έχει τα πλεονεκτήματα και τα μειονεκτήματα του. Αυτά είναι:

Πλεονεκτήματα

- Λειτουργεί πολύ καλά σε μικρά έργα όπου οι απαιτήσεις είναι λιγότερες άρα δεν υπάρχουν κενά μεταξύ της ανάλυσης και της υλοποίησης.
- Απλό στη χρήση του ακόμα και από ομάδες με μικρή εμπειρία.
- Κάθε φάση είναι διακριτή και αυτό έχει σαν αποτέλεσμα τα παραδοτέα να είναι συγκεκριμένα και ο κάθε εμπλεκόμενος σε κάθε φάση να γνωρίζεις ακριβώς τι ενέργειες πρέπει να κάνει χωρίς να αποκλίνει από το στόχο του.

Μειονεκτήματα

- Δύσκολο στη χρήση του για μεγάλα έργα όπου οι απαιτήσεις είναι περίπλοκες και αρκετές φορές δεν είναι σωστά καθορισμένες.
- Δυσκολία στην ανατροφοδότηση καθώς όταν το τελικό παραδοτέο παραδοθεί στην ομάδα ελέγχου είναι πολύ δύσκολο να πάει στην πρώτη φάση και να υπάρξει αλλαγή σε κάποια λειτουργικότητα και να επαναληφθούν όλες οι φάσεις του μοντέλου.

2.2.3 Σπειροειδές Μοντέλο

Στις προηγούμενες δύο υπό-ενότητες υπήρξε εκτενής αναφορά σε δύο ακολουθιακά μοντέλα. Στην υπό-ενότητα αυτή θα ασχοληθούμε με ένα επαναληπτικό μοντέλο που συνδυάζει τα χαρακτηριστικά των προηγούμενων μοντέλων (Καταρράκτη και V-Model) και αυτό είναι το σπειροειδές μοντέλο [1] [11]. Το μοντέλο αυτό αποτελείται από τέσσερις φάσεις όπου το υπό-υλοποίηση πρόγραμμα «περνάει» επαναληπτικά σε κάθε μια από αυτές. Κάθε επανάληψη σε μια σπείρα ουσιαστικά είναι μια φάση ανάπτυξης και το λογισμικό περνάει επανειλημμένα και διαδοχικά από αυτές τις φάσεις. Για να το εξηγήσουμε καλύτερα αυτό, σε αντίθεση με το μοντέλο καταρράκτη που με την ολοκλήρωση μιας φάσης δε μπορούμε να επιστρέψουμε πίσω, στο συγκεκριμένο μοντέλο όπως και σε όλα τα επαναληπτικά μοντέλα μπορούμε να

επανέλθουμε σε μια φάση ακόμα και αν αυτή έχει ολοκληρωθεί. Επιπρόσθετα με την ολοκλήρωση όλων των φάσεων δεν σημαίνει ότι έχει ολοκληρωθεί όλο το προϊόν αλλά ένα μέρος αυτού. Επαναληπτικά ξαναγίνονται όλες οι φάσεις μέχρι και τη δημιουργία του τελικού παραδοτέου που περιλαμβάνει όλες τις απαιτήσεις του πελάτη.

Οι τέσσερις φάσεις του σπειροειδές μοντέλου είναι:

Φάση Σχεδίασης (Planning): Η φάση αυτή περιλαμβάνει τη συλλογή των απαιτήσεων από τον πελάτη και τη λεπτομερή καταγραφή τους έτσι ώστε να μπορούν να χρησιμοποιηθούν από τις υπόλοιπες ομάδες. Σε αυτή τη φάση γίνεται και η εκτίμηση του κόστους του έργου και των πόρων που θα χρειαστούν.

Φάση Ανάλυσης Κινδύνου (Risk Analysis): Στη φάση αυτή αναλύονται όλοι οι πιθανοί κίνδυνοι που μπορεί να έχει το σύστημα και γίνεται προσπάθεια εύρεσης λύσεων έτσι ώστε το σύστημα να πληροί όλους τους περιορισμούς.

Φάση Προγραμματισμού και Ελέγχου (Engineering): Η φάση αυτή όπως φαίνεται και από την ονομασία της, περιλαμβάνει την ανάπτυξη του προϊόντος από την ομάδα των προγραμματιστών και τον έλεγχο του προϊόντος από την ομάδα ελέγχου. Στη συνέχεια το προϊόν μεταφέρεται στο περιβάλλον του πελάτη.

Φάση Αξιολόγησης (Evaluation): Ο πελάτης αξιολογεί το προϊόν που του παραδόθηκε. Όπως ήδη αναφέρθηκε, το προϊόν μπορεί να μην περιλαμβάνει όλες τις απαιτήσεις του πελάτη αλλά ένα μέρος αυτών. Σε επόμενη επανάληψη των προαναφερθέντων φάσεων θα του παραδοθεί κάτι καινούριο και αυτό θα συνεχιστεί μέχρι να πάρει το τελικό παραδοτέο προϊόν.

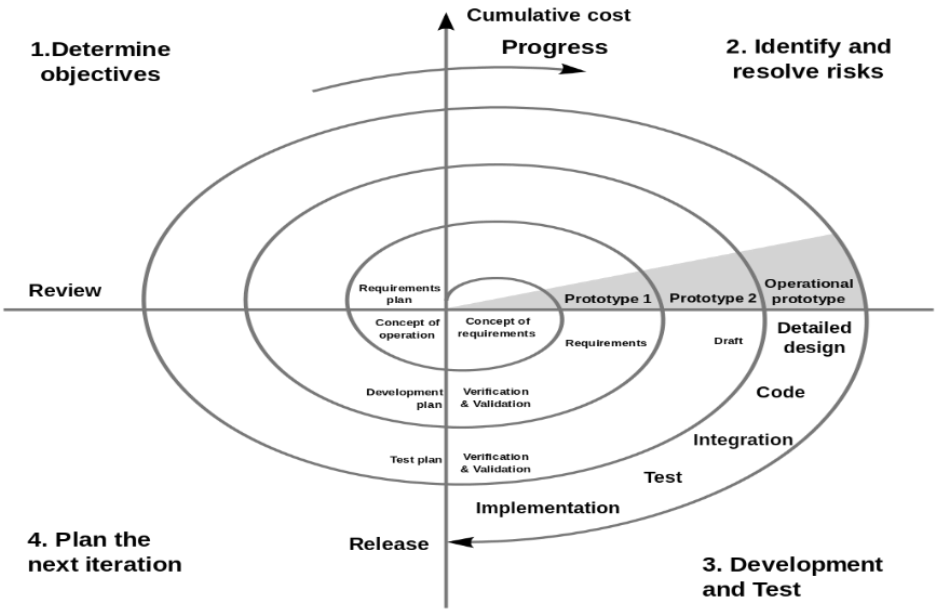
Όπως και τα ακολουθιακά μοντέλα έτσι και το μοντέλο αυτό έχει τα δικά του πλεονεκτήματα και μειονεκτήματα. Αυτά είναι:

<u>Πλεονεκτήματα</u>

- Ιδανικό για έργα υψηλού κινδύνου όπου οι ανάγκες του πελάτη μπορεί να αλλάξουν.
- Αλλαγές στη λειτουργικότητα καθώς και προσθήκη νέων απαιτήσεων είναι εύκολο να επιτευχθούν λόγω των συχνών επαναλήψεων.
- Οι κίνδυνοι διαχειρίζονται πιο εύκολα λόγω του συνεχούς ελέγχου λογισμικού.
- Ο πελάτης βλέπει το υπό-ανάπτυξη προϊόν συχνά οπότε υπάρχει συνεχής ανατροφοδότηση και το τελικό προϊόν είναι συχνά το επιθυμητό.

<u>Μειονεκτήματα</u>

- Υψηλό κόστος γιατί αρκετές φορές οι επαναλήψεις είναι πολλές μέχρι την ολοκλήρωση του προϊόντος άρα απαιτείται περισσότερος χρόνος και περισσότεροι πόροι.
- Περίπλοκο μοντέλο ειδικά σε μικρά έργα όπου οι απαιτήσεις είναι λίγες και σωστά προκαθορισμένες.
- Για τη σωστή λειτουργεία του απαιτείται να ακολουθούνται οι διαδικασίες του μοντέλου αυστηρά. Πολύ εύκολα μπορεί να δημιουργηθούν προβλήματα αν δεν τηρηθεί πιστά η εφαρμογή του.



Εικόνα 6: Σπειροειδές Μοντέλο [12]

2.2.4 Agile

Το τελευταίο μοντέλο που θα αναλυθεί στα πλαίσια αυτής της εργασίας είναι το μοντέλο Agile [1] [13] [14]. Το συγκεκριμένο μοντέλο είναι από τα πιο διαδεδομένα τα τελευταία χρόνια και όλα και περισσότερες εταιρίες το χρησιμοποιούν για να βελτιώσουν τη διαδικασίας παραγωγής λογισμικού. Αποτελεί ένα επαναληπτικό μοντέλο με έμφαση στην ευελιξία. Η ανάπτυξη του λογισμικού γίνεται σταδιακά και το έργο χωρίζεται σε μικρά τμήματα. Για παράδειγμα στο μοντέλο Scrum που ανήκει στη μεγάλη ομάδα των μοντέλων Agile, κάθε επανάληψη ή Sprint, όπως επιστημονικά ονομάζεται, διαρκεί συνήθως από μία έως τρεις βδομάδες και σε αυτήν συμμετέχουν όλες οι ομάδες, όπως για παράδειγμα οι αναλυτές, οι σχεδιαστές, οι προγραμματιστές και οι testers με σκοπό τη δημιουργία κάποιας λειτουργικότητας του συστήματος. Στο τέλος κάθε επανάληψης, το παραδοτέο που έχει ετοιμαστεί μπορεί να προβληθεί στον πελάτη και σε όλους τους εμπλεκόμενους φορείς. Στο μοντέλο Agile υπάρχουν, εκτός των κλασσικών ρόλων όλων των μοντέλων (π.χ. προγραμματιστής, αναλυτής, tester), και δύο ακόμη ρόλοι. Ο ιδιοκτήτης του προϊόντος (Product Owner) που είναι υπεύθυνος για την πορεία του έργου και ο Scrum Master που εξασφαλίζει ότι οι όλες οι εμπλεκόμενες τεχνολογικές ομάδες είναι αποδοτικές, παραγωγικές και ακολουθούν όλες τις διαδικασίες που έχουν συμφωνηθεί στην αρχή του έργου.

Συνήθως το μοντέλο Agile αποτελείται από 5 φάσεις. Αυτές είναι:

Φάση των απαιτήσεων: Στη φάση αυτή αναλύονται οι απαιτήσεις του πελάτη και προσδιορίζεται τι ακριβώς θα υλοποιηθεί. Είναι απαραίτητο να γίνει σωστή και λεπτομερής καταγραφή των απαιτήσεων. Μετά την καταγραφή των απαιτήσεων ο πελάτης εξετάζει τις απαιτήσεις αν πληρούν αυτά που επιθυμεί ο ίδιος να υλοποιηθούν.

Φάση σχεδιασμού: Υπάρχουν 2 προσεγγίσεις σχετικά με τη φάση σχεδιασμού. Η μία είναι η αρχιτεκτονική σχεδίαση και η άλλη η σχεδίαση των γραφικών και γενικά της απεικόνισης της εφαρμογής. Στην αρχιτεκτονική σχεδίαση, στις πρώτες επαναλήψεις ορίζεται η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί, οι βιβλιοθήκες καθώς επίσης και οι βάσεις δεδομένων και η αρχιτεκτονική τους. Στις επόμενες επαναλήψεις που ήδη έχει ξεκινήσει και η φάση προγραμματισμού σχεδιάζεται η δομή της κάθε νέας λειτουργικότητας που θα υλοποιηθεί. Στη σχεδίαση της γραφικής απεικόνισης του συστήματος, οι σχεδιαστές δημιουργούν πρόχειρα αρχεία με το πώς θα φαίνεται η εφαρμογή έτσι ώστε ο προγραμματιστής να ξέρει τι θα υλοποιήσει. Σε επόμενες επαναλήψεις γίνεται προσπάθεια βελτίωσης του συστήματος στον τομέα της διεπαφής χρήστη και δημιουργούνται και τα νέα πρόχειρα αρχεία για τις νέες λειτουργίες του συστήματος.

Φάση προγραμματισμού: Η φάση αυτή, όπως και στα προηγούμενα μοντέλα που αναλύθηκαν, αποτελεί την πιο χρονοβόρα φάση ενός έργου, καθώς εδώ πραγματοποιείται η υλοποίηση του έργου μέσω κώδικα από την ομάδα των προγραμματιστών. Σε κάθε επανάληψη είτε δημιουργούνται νέες λειτουργικότητες, είτε βελτιώνονται υπάρχουσες, είτε λύνονται προβλήματα που έχουν αναφερθεί από την ομάδα ελέγχου.

Φάση ελέγχου λογισμικού: Στη φάση αυτή πραγματοποιούνται όλοι οι απαραίτητοι έλεγχοι από την ομάδα των testers, έτσι ώστε να ελεγχτεί αν οι απαιτήσεις που έχουν δοθεί έχουν υλοποιηθεί σωστά και το πρόγραμμα δεν έχει λάθη. Ειδικότερα στο μοντέλο Agile σε κάθε επανάληψη η πολυπλοκότητα ελέγχου μπορεί και να αυξάνεται καθώς εκτός από τον έλεγχο μιας συγκεκριμένης λειτουργικότητας που μπορεί να παραδοθεί , πρέπει να ελεγχθούν και όλες οι υπόλοιπες λειτουργικότητες που είχαν παραδοθεί σε προηγούμενες επαναλήψεις καθώς και η επικοινωνία μεταξύ παλιών και νέων λειτουργιών.

Φάση εγκατάστασης: Στη φάση αυτή εγκαθίσταται η εφαρμογή στο περιβάλλον του πελάτη είτε για να τη δει στα πλαίσια ενός πρόχειρου παραδοτέου (demo) είτε για να την ελέγξει και ο ίδιος (User Acceptance Testing – UAT) και στη συνέχεια να ανατροφοδοτήσει τις ομάδες με τα σχόλιά του και τυχών προβλήματα που ίσως έχει βρει. Σε κάθε επανάληψη μπορεί να γίνει και νέα εγκατάσταση στο περιβάλλον του πελάτη με πιο ανανεωμένη έκδοση που μπορεί να περιλαμβάνει επίλυση τυχόν προβλημάτων που έχουν προκύψει καθώς και νέα λειτουργικότητα που έχει δημιουργηθεί από την ομάδα των προγραμματιστών και έχει περάσει επιτυχώς και από τους ελέγχους της ομάδας των testers.

Φάση ανασκόπησης: Η τελευταία φάση του μοντέλου Agile αποτελεί ένα είδος αυτοκριτικής και συζήτησης μεταξύ των ομάδων και όλων των εμπλεκόμενων. Μέσω κάποιων συναντήσεων οι ομάδες συζητούν την πορεία του έργου καθώς και οτιδήποτε τους έχει απασχολήσει στην υλοποίησή του. Ο ιδιοκτήτης του προϊόντος (Product Owner) λαμβάνει υπόψη του όλους τους προβληματισμούς που υπάρχουν καθώς επίσης και τυχόν βελτιώσεις στον τρόπο λειτουργίας μεταξύ των ομάδων έτσι ώστε σε επόμενες επαναλήψεις να επιτευχθεί καλύτερο αποτέλεσμα με λιγότερα προβλήματα είτε στην υλοποίηση του προγράμματος είτε και στην επικοινωνία μεταξύ των ομάδων.

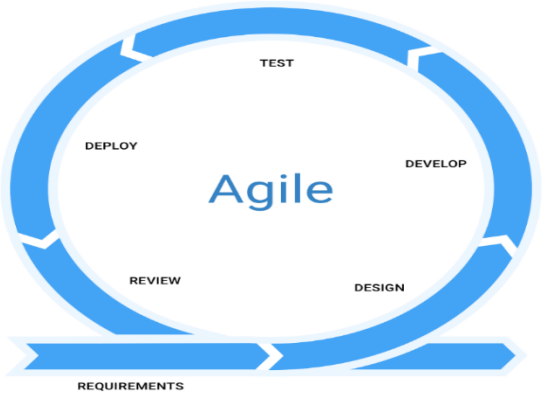
<u>Πλεονεκτήματα</u>

- Προωθεί την ομαδική εργασία. Οι ομάδες αλληλοεπιδρούν συνεχώς μεταξύ τους.

- Ο πελάτης είναι συνεχώς ικανοποιημένος λόγω της συνεχούς παροχής λογισμικού και μπορεί να ελέγχει ευκολότερα την πρόοδο του έργου.
- Οι αλλαγές στις απαιτήσεις μπορούν να επιτευχθούν πιο εύκολα σε σχέση με τα ακολουθιακά μοντέλα.
- Προσφέρει ευελιξία.
- Ταυτόχρονα μπορούν να δουλεύουν πολλές ομάδες οπότε εξοικονομείται χρόνος.

Μειονεκτήματα

- Λόγω των πολλών επαναλήψεων είναι δύσκολο να εκτιμηθεί ο χρόνος και το κόστος του έργου στην αρχή του έργου.
- Το τελικό παραδοτέο μπορεί να έχει πολλές διαφορές σε σχέση με αυτά που επιθυμεί ο πελάτης αν οι απαιτήσεις δεν είναι καλά καταγεγραμμένες. Γενικά στο συγκεκριμένο μοντέλο υπάρχει μεγάλη εξάρτηση από τον πελάτη.
- Η μεταφορά της τεχνολογίας σε νέα μέλη της ομάδας είναι δύσκολη καθώς υπάρχει έλλειψη τεκμηρίωσης.
- Λόγω της έλλειψης τεκμηρίωσής που ήδη αναφέρθηκε υπάρχει υψηλή ατομική εξάρτηση, δηλαδή κάποιο έμπειρο άτομο στο έργο μπορεί να επιφορτιστεί με πάρα πολλές υποχρεώσεις και αντίστοιχα αν αποδεσμευτεί από το έργο να μην υπάρχουν εύκολα άτομα που να συνεχίσουν την υλοποίησή του.



Εικόνα 7: Agile Model [15]

3. ΄ΕΛΕΓΧΟΣ ΛΟΓΙΣΜΙΚΟΥ - ΜΙΑ ΘΕΩΡΗΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ

Στο προηγούμενο κεφάλαιο αναλύθηκε ο κύκλος ζωής ενός λογισμικού καθώς και κάποια από τα πιο διαδεδομένα μοντέλα που χρησιμοποιούνται στην εποχή μας. Στο συγκεκριμένο κεφάλαιο θα αναλυθεί εκτενώς μια συγκεκριμένη φάση από τον κύκλο ζωής ενός λογισμικού και αυτή είναι ο έλεγχος λογισμικού που αποτελεί και το θέμα αυτής της διπλωματικής εργασίας.

3.1 Τι είναι ο έλεγχος λογισμικού

Στην εποχή που ζούμε παρατηρείται ραγδαία εξέλιξη της τεχνολογίας και αυτό έχει σαν αποτέλεσμα οι απαιτήσεις και ο ανταγωνισμός στα λογισμικά που δημιουργούνται να έχει αυξηθεί. Αυτό έχει σαν αποτέλεσμα την καλύτερη ποιότητα προϊόντων προς τον χρήστη αλλά και τη δυσκολία στην παραγωγή τους από τις τεχνικές ομάδες που αναλαμβάνουν το κάθε έργο. Επιπρόσθετα σε πολλούς τομείς (π.χ. υγεία) η ποιότητα του λογισμικού είναι απαραίτητο να είναι η ιδανική γιατί αν το τελικό προϊόν δεν είναι το επιθυμητό μπορεί να υπάρξουν συνέπειες ακόμα και στην ίδια τη ζωή του ανθρώπου. Όλα τα παραπάνω επιτυγχάνονται μέσω του ελέγχου λογισμικού.

Ουσιαστικά ο έλεγχος λογισμικού είναι μια έρευνα που διεξάγεται για να παρέχει στους ενδιαφερόμενους πληροφορίες σχετικά με την ποιότητα του υπό εξέταση προϊόντος ή υπηρεσίας. Επιπρόσθετα παρέχει μια αντικειμενική άποψη για το λογισμικό που επιτρέπει στην επιχείρηση να εκτιμήσει και να κατανοήσει τους κινδύνους του λογισμικού [16].

Για να κατανοηθεί καλύτερα η σημαντικότητα του ελέγχου λογισμικού, η οικονομία της Αμερικής χάνει κάθε χρόνο περίπου 60 δισεκατομμύρια δολάρια από ελαττωματικά λογισμικά. Κάποια γνωστά παραδείγματα ελαττωματικών λογισμικών και τι συνέπειες είχαν θα αναφερθούν παρακάτω [17].

- **Η ανατίναξη του πυραύλου Ariane 5 (1996):** Ο πύραυλος αυτός καταστράφηκε μετά από 36 δευτερόλεπτα στον αέρα και ο λόγος ήταν ότι προσπαθούσαν να αποθηκευτούν δεδομένα 64 bit σε φόρμα 16 bit. Η συνολική απώλεια εκτιμάται στα 8 δισεκατομμύρια δολάρια.
- **Η θεραπεία μέσω του Therac-25 (1985):** Για την αντιμετώπιση του καρκίνου είχε δημιουργηθεί ένα μηχάνημα πού αντί για 1 φορά ακτινοβολία μετέφερε στον άνθρωπο 100 φορές ακτινοβολία. Ο λόγος που συνέβη αυτό ήταν μια συνθήκη στον κώδικα που είχε οριστεί λάθος. Αυτό είχε σαν αποτέλεσμα εκατοντάδες ζωές να χαθούν.
- **Οι νικητές της Pepsi (1992):** Το 1992 στις Φιλιππίνες ξεκίνησε μια διαφήμιση που θα έδινε σε έναν τυχερό 37 χιλιάδες δολάρια αν έβρισκε ένα συγκεκριμένο αριθμό στο καπάκι του μπουκαλιού του συγκεκριμένου αναψυκτικού. Ο αριθμός 349 που ήταν ο τυχερός αριθμός βρέθηκε σε περίπου 500 χιλιάδες μπουκάλια. Η εταιρία δήλωσε ότι υπήρξε λάθος στο πρόγραμμα που επέλεξε τον αριθμό και δεν πλήρωσε τους συμμετέχοντες, ωστόσο υπήρξαν πάρα πολλές μηνύσεις και πάρα πολλές καταστροφές σε κτίρια της εταιρίας με αποτέλεσμα η εταιρία να χάσει πολλά εκατομμύρια ευρώ. Εκτός από αυτό, οι νικητές διαδήλωσαν στους δρόμους εξαγριωμένοι και λόγω αυτού πέθαναν 3 άτομα.

Συμπερασματικά όλα όσα αναφέρθηκαν δημιουργούν μια μεγάλη ανάγκη ώστε η ποιότητα του λογισμικού να είναι υψηλή και να μειωθούν όσο το δυνατόν γίνεται τα σφάλματα σε ένα λογισμικό. Για να επιτευχθεί αυτό απαιτείται σωστός και οργανωμένος έλεγχος του λογισμικού.

3.2 Οι 7 αρχές ελέγχου λογισμικού

Σύμφωνα με το International Software Testing Qualifications Board υπάρχουν 7 βασικές αρχές στον έλεγχο λογισμικού [1]. Οι συγκεκριμένες αρχές είναι πολύ διαδεδομένες στο χώρο του ελέγχου λογισμικού και κάθε άνθρωπος που έχει εργαστεί στο συγκεκριμένο τομέα αλλά και γενικά σε έναν κύκλο ζωής παραγωγής λογισμικού τις παρατηρεί καθημερινά στη δουλειά του. Αυτές παρουσιάζονται στην εικόνα 8 και είναι:

1. **<u>Ο έλεγχος λογισμικού δείχνει την παρουσία προβλημάτων και όχι την απουσία.</u>**

Ουσιαστικά μέσω του ελέγχου λογισμικού μπορεί να αποδειχτεί ότι το λογισμικό έχει προβλήματα και αυτά να επιλυθούν από τις ομάδες των προγραμματιστών και επίσης μπορεί να επιτευχθεί μείωση της πιθανότητάς προβλημάτων στο σύστημα αλλά δεν μπορεί να αποδειχτεί ότι δεν υπάρχουν άλλα προβλήματα στο σύστημα.

2. **<u>Ο πλήρης έλεγχος λογισμικού είναι αδύνατος.</u>**

Όπως αναφέρεται και στον τίτλο είναι αδύνατο να ελεγχθεί ένα σύστημα πλήρως. Με την έννοια πλήρης έλεγχος συστήματος εννοούμε να εξεταστούν όλες οι είσοδοι και οι συνθήκες που υπάρχουν στο σύστημα. Αυτό δεν μπορεί να επιτευχθεί κυρίως γιατί ο χρόνος που απαιτείται είναι τεράστιος και τα άτομα που χρειάζονται είναι πάρα πολλά. Για παράδειγμα μια απλή εφαρμογή μπορεί να χρειάζεται 600 χιλιάδες τεστ για να ελεγχθεί πλήρως. Αν θεωρήσουμε ότι κάθε τεστ διαρκεί 1 λεπτό τότε καταλαβαίνουμε ότι χρειάζονται περισσότερα από 2 χρόνια για να ελεγχθεί πλήρως χωρίς να υπολογίσουμε μια νέα έκδοση που μπορεί να απαιτεί άλλα τόσα τεστ για να ελεγχθεί. Το ερώτημα τώρα που δημιουργείται είναι τι μπορούν να κάνουν οι ομάδες ελέγχου λογισμικού από τη στιγμή που δεν μπορούν να ελέγχουν όλο το σύστημα. Η απάντηση σε αυτό το ερώτημα είναι ότι πρέπει να υπάρχουν προτεραιότητες στο τι θα ελεγχθεί στο σύστημα και πόσος χρόνος θα καταναλωθεί, ανάλογα και τη σημαντικότητα που έχει κάθε λειτουργικότητα.

3. **<u>Ο πρόωρος έλεγχος λογισμικού εξοικονομεί χρόνο και χρήμα.</u>**

Όσο πιο γρήγορα ξεκινάει ο έλεγχος λογισμικού τόσο πιο γρήγορα εντοπίζονται και τυχόν προβλήματα του συστήματος. Με τον όρο πρόβλημα στο σύστημα δεν εννοούμε μόνο τα προβλήματα στον κώδικα αλλά και προβλήματα σε άλλες φάσεις του κύκλου ζωής όπως για παράδειγμα στις προδιαγραφές. Για παράδειγμα αν εντοπιστεί ένα πρόβλημα στις προδιαγραφές και επιλυθεί γρήγορα τότε το πρόβλημα αυτό δεν θα επεκταθεί και στις υπόλοιπές φάσεις του κύκλου ζωής όπως για παράδειγμα στη φάση του προγραμματισμού. Αντίθετα αν αυτό το πρόβλημα εντοπιστεί αργά τότε θα πρέπει να επαναληφθούν όλες οι φάσεις του κύκλου ζωής και το κόστος θα είναι μεγάλο και χρονικά και οικονομικά.

4. **<u>Τα προβλήματα του λογισμικού συνήθως ομαδοποιούνται μαζί.</u>**

Έχει παρατηρηθεί ότι πολλά προβλήματα του λογισμικού συνήθως ομαδοποιούνται σε συγκεκριμένες λειτουργικότητες. Έτσι οι ομάδες ελέγχου λογισμικού οφείλουν να παρατηρούν που υπάρχουν τα περισσότερα προβλήματα και να προγραμματίζουν το χρόνο τους με σκοπό να ελέγξουν περισσότερο αυτές τις λειτουργικότητες σε σχέση με άλλες που δεν έχουν παρατηρηθεί πολλά προβλήματα σε βάθος χρόνου.

5. Το παράδοξο των φυτοφαρμάκων (Pesticide Paradox).

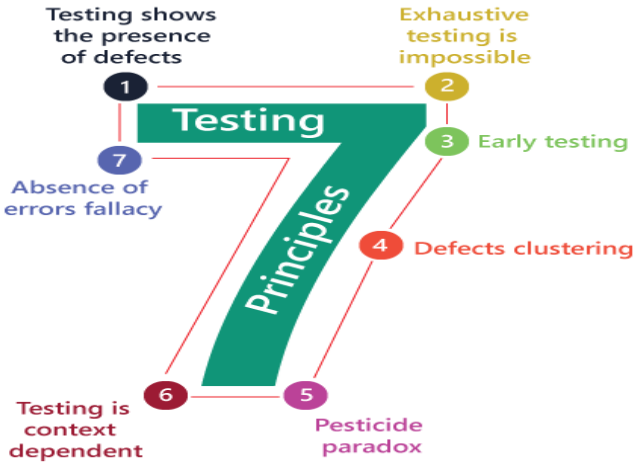
Ένα πρόβλημα σε ένα πρόγραμμα ονομάζεται έντομο (bug). Όπως και στα φυτά που αν τους ρίχνεις το ίδιο φυτοφάρμακο συνεχώς, ένα συγκεκριμένο έντομο μπορεί να μην εμφανιστεί πάλι αλλά μπορούν να εμφανιστούν διαφορετικά έντομα έτσι και στον τομέα του ελέγχου λογισμικού αν επιλαμβάνεται το ίδιο τεστ συνεχώς τότε κάποια στιγμή δεν θα υπάρχουν προβλήματα που να αφορούν το συγκεκριμένο τεστ αλλά μπορεί να έχουν δημιουργηθεί νέα προβλήματα που δεν καλύπτονται από το προαναφερθέν τεστ. Επιπρόσθετα, πολλές φορές οι προγραμματιστές μαθαίνουν τι είδους προβλήματα βρίσκουν οι testers με την πάροδο του χρόνου και προσπαθούν να μην έχουν τα προγράμματά τους προβλήματα σε αυτές τις περιοχές ελέγχου. Για την επίλυση όλων των προαναφερθέντων, οι ομάδες ελέγχουν λογισμικού πρέπει συνεχώς να ανανεώνουν τους ελέγχους που πραγματοποιούν, να προσθέτουν νέους ελέγχους αλλά και να αλλάζουν μεταξύ τους θέσεις έτσι ώστε κάθε tester να μην δουλεύει συνεχώς στην ίδια λειτουργικότητα αλλά και σε άλλες με σκοπό να βρεθούν περισσότερα προβλήματα στο λογισμικό από άτομα που μπορεί να σκέφτονται διαφορετικά.

6. Ο έλεγχος λογισμικού εξαρτάται από το περιβάλλον που πραγματοποιούνται οι έλεγχοι.

Υπάρχουν διαφορετικές προσεγγίσεις ως προς τον έλεγχο λογισμικού ανάλογα με τι θέλει να ελέγξει μια ομάδα καθώς επίσης και πόσα άτομα μπορεί να διαθέσει. Για παράδειγμα, διαφορετικά ελέγχεται μια απλή εφαρμογή κινητού σε σχέση με ένα σημαντικό ιατρικό όργανο. Επίσης διαφορετικά θα ελεγχθεί μια εφαρμογή όταν υπάρχουν διαθέσιμες 20 μέρες και 10 άτομα σε σχέση με το αν υπήρχαν διαθέσιμα 3 άτομα αλλά 50 μέρες.

7. Η απουσία των προβλημάτων είναι πλάνη.

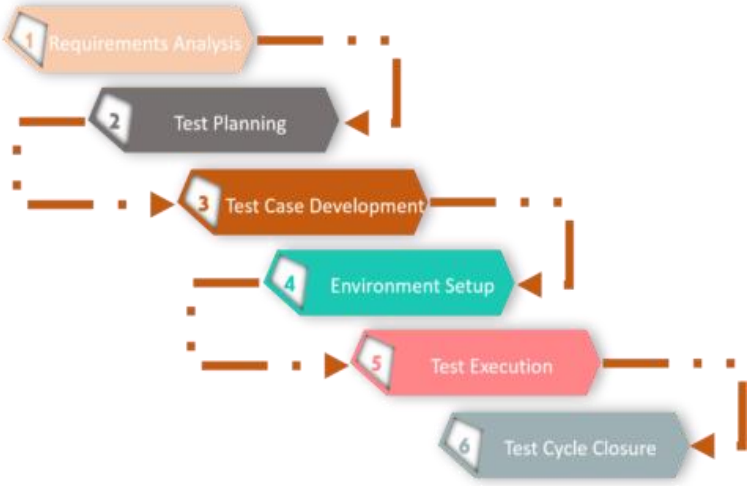
Η τελευταία αρχή ελέγχου λογισμικού έχει μια συσχέτιση και με τη δεύτερη που αναφέραμε. Δεν είναι δυνατόν οι ομάδες ελέγχου λογισμικού να τρέξουν όλους τους ελέγχους σε ένα σύστημα, να βρουν όλα τα πιθανά προβλήματα και ακόμα και αν αυτά φτιαχτούν, αυτόματα να θεωρηθεί ότι η εφαρμογή θα είναι σίγουρα επιτυχημένη. Είναι πολύ πιθανό όσο καλός έλεγχος λογισμικού και να γίνει, το τελικό παραδοτέο να έχει προβλήματα αλλά και ο πελάτης να το θεωρεί ελαττωματικό καθώς δεν πληροί τις ανάγκες και τις προσδοκίες του.



Εικόνα 8: Οι 7 αρχές ελέγχου λογισμικού [18]

3.3 Διαδικασία και φάσεις ελέγχου λογισμικού

Στο κεφάλαιο 2 αναλύσαμε τις φάσεις κύκλου ζωής λογισμικού. Στη συγκεκριμένα υπό-ενότητα του 3ου κεφαλαίου της παρούσας διπλωματικής εργασίας θα επικεντρωθούμε στον κύκλο ζωής μιας συγκεκριμένης φάσης και αυτή είναι ο έλεγχος λογισμικού. Ο κύκλος ζωής ελέγχου λογισμικού (Software Testing Life Cycle – STLC) συνήθως αποτελείται από 6 κύριες δραστηριότητες. Αυτές είναι: η ανάλυση των απαιτήσεων, ο προγραμματισμός των δοκιμών, η σχεδίαση τον δοκιμών, η εγκατάσταση στο περιβάλλον που θα γίνουν οι δοκιμές, η εκτέλεση των δοκιμών και η ολοκλήρωση των δοκιμών [1] [19]. Οι δραστηριότητες αυτές παρουσιάζονται στην εικόνα 9 και θα αναλυθούν συνοπτικά παρακάτω.



Εικόνα 9: Φάσεις Ελέγχου Λογισμικού [20]

Ανάλυση απαιτήσεων: Αποτελεί την πρώτη δραστηριότητα που εκτελείται στον κύκλο ζωής ελέγχου λογισμικού. Η ομάδα ελέγχου λογισμικού προσπαθεί να κατανοήσει τις απαιτήσεις που

έχουν τεθεί από τον πελάτη και έχουν καταγραφεί από την ομάδα των αναλυτών του συστήματος. Στη δραστηριότητα αυτή υπάρχει συχνά αλληλεπίδραση μεταξύ της ομάδας ελέγχου λογισμικού και της ομάδας των αναλυτών με στόχο να κατανοηθούν πλήρως όλες οι απαιτήσεις του πελάτη. Έχει παρατηρηθεί ότι όσο καλύτερα κατανοηθούν οι απαιτήσεις από την ομάδα ελέγχου τόσο καλύτερη ποιότητα θα παραδοθεί στον πελάτη μετά την ολοκλήρωση όλων των φάσεων του κύκλου ζωής λογισμικού. Πολλές φορές σε αυτή τη δραστηριότητα οι ομάδες ελέγχου λογισμικού παρατηρούν προβλήματα στις απαιτήσεις και τις αναφέρουν στην ομάδα των αναλυτών. Αυτό είναι ιδιαίτερο χρήσιμο όπως έχει ήδη αναφερθεί γιατί αν οι απαιτήσεις είναι σωστές τότε η εταιρία που παράγει το λογισμικό κερδίζει χρόνο και χρήματα και επίσης το τελικό παραδοτέο θα είναι το επιθυμητό.

Προγραμματισμός των δοκιμών: Η δραστηριότητα αυτή περιλαμβάνει τον προγραμματισμό του χρόνου που θα χρειαστεί η ομάδα ελέγχου, τα άτομα που θα χρειαστούν, τα εργαλεία, τον καταμερισμό ρόλων και γενικά τη στρατηγική που θα ακολουθήσει η ομάδα ελέγχου λογισμικού για τη σωστή περάτωση του έργου που έχει αναλάβει.

Σχεδίαση των δοκιμών: Λαμβάνοντας υπόψιν τις δύο προηγούμενες δραστηριότητες, στην παρούσα δραστηριότητα σχεδιάζονται όλοι οι έλεγχοι που θα εκτελεστούν στο σύστημα. Κάθε έλεγχος πρέπει να είναι αποτελεσματικός δηλαδή να μπορεί να εντοπίζει προβλήματα στο σύστημα, να είναι εξελίξιμος δηλαδή να μπορεί εύκολα να συντηρηθεί και να αλλαχθεί αλλά και να είναι οικονομικός στη χρήση του. Επίσης είναι σημαντικό κάθε έλεγχος να έχει μια συγκεκριμένη προτεραιότητα ως προς την εκτέλεσή του ανάλογα και με το τι ακριβώς ελέγχει γιατί όπως έχουμε ήδη αναφέρει ο πλήρης έλεγχος ενός συστήματος δεν είναι εφικτός οπότε πρέπει να ελεγχθούν πρώτα οι πιο σημαντικές μονάδες ενός συστήματος.

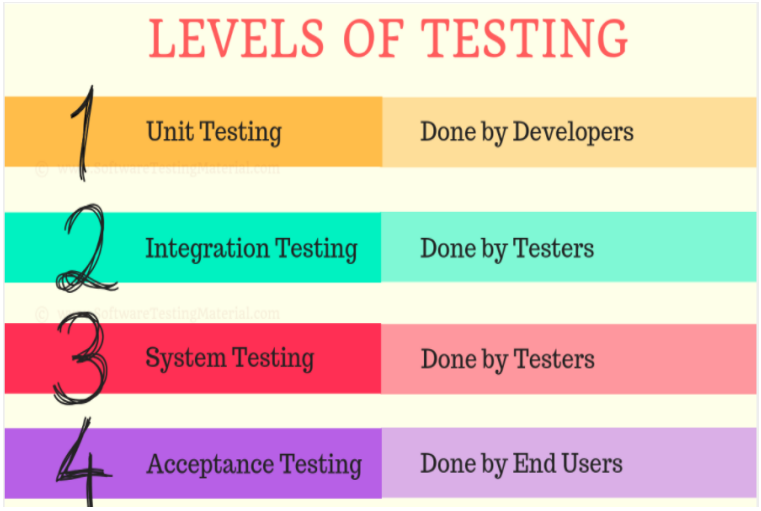
Εγκατάσταση στο περιβάλλον που θα γίνουν οι δοκιμές: Η συγκεκριμένη δραστηριότητα πολλές φορές εκτελείται από άλλες ομάδες και όχι από την ομάδα ελέγχου λογισμικού. Το ζητούμενο είναι να δημιουργηθεί ένα περιβάλλον για την ομάδα ελέγχου λογισμικού για να εκτελέσει τους ελέγχους της. Πάνω σε αυτό το περιβάλλον θα εγκατασταθεί το λογισμικό που έχουν ετοιμάσει οι προγραμματιστές και είναι έτοιμο για έλεγχο. Συνήθως η μόνη ενέργεια που εκτελεί η ομάδα ελέγχου λογισμικού σε αυτή τη δραστηριότητα είναι ένας γρήγορος έλεγχος έτσι ώστε να ελέγξει σε πρώτη φάση ότι το περιβάλλον είναι λειτουργικό (π.χ. ότι ανοίγει η εφαρμογή και μπορείς να περιηγηθείς στις βασικές τις σελίδες).

Εκτέλεση των δοκιμών: Αυτή η δραστηριότητα είναι η πιο μεγάλη και χρειάζεται τον περισσότερο χρόνο για να εκτελεστεί από τις ομάδες ελέγχου λογισμικού. Όλα τα σενάρια που αναλύθηκαν και σχεδιάστηκαν στις προηγούμενες φάσεις εκτελούνται με σκοπό να ελεγχθεί η ποιότητα του συστήματος σύμφωνα με τις απαιτήσεις που έχουν δοθεί από τον πελάτη. Όλα τα προβλήματα που εντοπίζονται, αναφέρονται στην ομάδα των προγραμματιστών και αυτή με τη σειρά τους τα επιλύουν και εγκαθιστούν νέες βελτιωμένες εκδόσεις στο περιβάλλον ελέγχου δοκιμών. Στη συνέχεια, η ομάδα ελέγχου λογισμικού ελέγχει ξανά αν όντως επιλύθηκαν αυτά τα προβλήματα και επιβεβαιώνει την επίλυσή τους. Αυτή η διαδικασία επαναλαμβάνεται συνεχώς ανάλογα βέβαια και με το διαθέσιμο χρόνο που υπάρχει μέχρι και την τελική παράδοση του έργου. Είναι συχνό φαινόμενο κάποια μικρά προβλήματα του συστήματος (π.χ. το χρώμα σε ένα κουμπί να μην είναι το επιθυμητό) να μην επιλύονται άμεσα γιατί δεν έχουν προτεραιότητα σε σχέση με άλλα σφάλματα του συστήματος (π.χ. το ίδιο κουμπί να μην είναι λειτουργικό).

Ολοκλήρωση των δοκιμών: Στην τελευταία φάση ελέγχονται αν όλα τα προβλήματα έχουν επιλυθεί και όσα δεν έχουν επιλυθεί καταγράφονται για να δημιουργηθεί η τελική αναφορά με όλη την πορεία του ελέγχου λογισμικού προς όλους τους εμπλεκόμενους. Με την ολοκλήρωση των δοκιμών, η ομάδα ελέγχου λογισμικού κάνει μια εσωτερική αξιολόγηση της συνολικής της πορείας σύμφωνα και με τα δεδομένα που έχει συλλέξει με σκοπό σε μελλοντικά έργα να είναι πιο αποτελεσματική και αποδοτική.

3.4 Επίπεδα ελέγχου λογισμικού

Μια ακόμα σημαντική θεωρητική αναφορά πρέπει να γίνει και στα επίπεδα ελέγχου λογισμικού. Τα επίπεδα ελέγχου λογισμικού ουσιαστικά αποτελούν ομαδοποιημένες δραστηριότητες ελέγχου που οργανώνονται και εκτελούνται μαζί με σκοπό να αποτραπεί η συνεχής επανάληψη των ίδιων ελέγχων μεταξύ των φάσεων του κύκλου ζωής ανάπτυξης λογισμικού [1]. Για κάθε επίπεδο ελέγχου συνήθως υπεύθυνη είναι μια συγκεκριμένη ομάδα και απαιτείται ένα κατάλληλο περιβάλλον για να εκτελεστούν όλοι οι απαραίτητοι έλεγχοι. Τα βασικά επίπεδα ελέγχου λογισμικού όπως και οι υπεύθυνες ομάδες που τα εκτελούν φαίνονται στην εικόνα 10 και είναι οι δοκιμές μονάδας, οι δοκιμές ενσωμάτωσης, οι δοκιμές του συστήματος και οι δοκιμές αποδοχής [1] [21] [22].



Εικόνα 10: Επίπεδα Ελέγχου Λογισμικού [23]

Δοκιμή μονάδας (Unit Testing): Γνωστό και ως δοκιμή συστατικών (component testing) ή δοκιμή ενοτήτων. Αποτελεί τον πρώτο έλεγχο που πρέπει να πραγματοποιηθεί στο σύστημα. Μέσω του ελέγχου αυτού εξετάζεται κάθε μονάδα του λογισμικού αν λειτουργεί σωστά και σύμφωνα με τις προδιαγραφές που έχουν δοθεί από την ομάδα των αναλυτών. Συνήθως οι έλεγχοι αυτοί γίνονται από την ομάδα των προγραμματιστών που έγραψαν τον κώδικα καθώς απαιτείται γνώση και πρόσβαση στον κώδικα του λογισμικού (κλάσεις, δομή κώδικα, βάσεις δεδομένων). Ο συγκεκριμένος έλεγχος είναι πολύ σημαντικός γιατί όσο νωρίτερα εντοπίζονται σφάλματα, τόσο περισσότερο μειώνεται το ρίσκο μιας προβληματικής εφαρμογής και όλο και περισσότερος χρόνος αλλά και χρήματα εξοικονομούνται.

Δοκιμή ενσωμάτωσης (Integration Testing): Οι συγκεκριμένες δοκιμές επικεντρώνονται στην επικοινωνία μεταξύ των διαφορετικών συστατικών του συστήματος με στόχο να εντοπιστούν όλα τα λειτουργικά και μη λειτουργικά προβλήματα που μπορεί να υπάρχουν σε αυτή την επικοινωνία. Στις δοκιμές αυτές εκτός από τα προβλήματα στην επικοινωνία μεταξύ των συστατικών του συστήματος, εντοπίζονται και προβλήματα στα ίδια τα συστατικά του συστήματος που ίσως δεν εντοπίστηκαν στις δοκιμές μονάδας που αναλύσαμε προηγουμένως. Υπάρχουν πολλές διαφορετικές προσεγγίσεις για το πώς μπορεί να γίνει ο έλεγχος αυτός. Δυο από τις πιο γνωστές είναι η από πάνω προς τα κάτω δοκιμή που πρώτα ελέγχονται τα υψηλού επιπέδου συστατικά του συστήματος και έπειτα τα συστατικά χαμηλότερου επιπέδου και η από κάτω προς τα πάνω δοκιμή που λειτουργεί με τον ακριβώς αντίθετο τρόπο, δηλαδή πρώτα ελέγχονται τα χαμηλότερου επιπέδου συστατικά του συστήματος και έπειτα τα υψηλότερα μέχρι να φτάσουμε στο πιο υψηλό.

Δοκιμή του συστήματος (System Testing): Όπως είναι εύκολα κατανοητό και από την ονομασία της δοκιμής αυτής, οι έλεγχοι επικεντρώνονται σε όλο το σύστημα (Software and Hardware) σαν μια οντότητα. Στις συγκεκριμένες δοκιμές εξετάζεται όλη η λειτουργία του συστήματος «end to end» δηλαδή ολοκληρωτικά ό,τι περιλαμβάνει το σύστημα. Οι δοκιμές αυτές αρκετά συχνά απαιτούνται να πραγματοποιηθούν λόγω των συμβολαίων που έχουν υπογραφεί μεταξύ των εμπλεκομένων εταιριών που αφορά το έργο.

Δοκιμή αποδοχής (Acceptance Testing): Αποτελεί την τελευταία δοκιμή πριν το λογισμικό παραδοθεί στον πελάτη και πρέπει μέσω αυτών των δοκιμών να ελεγχθεί και να διασφαλιστεί ότι το σύστημα που έχει αναπτυχθεί πληροί τις απαιτήσεις του πελάτη. Αναλυτικότερα, μπορεί να περιλαμβάνει έλεγχο σε λειτουργικότητες του συστήματος αλλά και σε πιο απλά πράγματα, όπως κάποιο ορθογραφικό λάθος ή κάποιο πρόβλημα στο σχήμα και το χρώμα των κουμπιών της εφαρμογής. Συχνά τα αποτελέσματα που προκύπτουν αποτελούν τη βάση για το αν θα πρέπει το σύστημα να παραδοθεί στον πελάτη ή όχι και με τι προβλήματα. Υπάρχουν αρκετοί διαφορετικοί τύποι τέτοιον δοκιμών. Αυτοί είναι:

- **User Acceptance Test (UAT):** Ελέγχεται ότι ο χρήστης θα μπορεί να χρησιμοποιήσει το σύστημα με όσο το δυνατόν μικρότερη δυσκολία, κόστος και χωρίς προβλήματα.
- **Operations Acceptance Test:** Ελέγχεται ότι οι διαχειριστές του συστήματος θα μπορούν να κρατήσουν ενεργό το σύστημα και να δουλεύει χωρίς προβλήματα στο παραγωγικό περιβάλλον.
- **Contract and Regulation Acceptance Test:** Ελέγχεται ότι όλα τα ζητήματα σχετικά με το συμβόλαιο που έχει υπογραφεί και τους νόμους που επικρατούν έχουν διασφαλιστεί και το σύστημα δεν θα έχει κάποιο είδος «παράνομης» χρήσης. Για παράδειγμα, οι νόμοι της πολιτείας της Washington δεν επιτρέπουν να παίζεις διαδικτυακό αθλητικό στοίχημα δίπλα σε μεγάλα μουσεία. Έτσι μια εφαρμογή αθλητικού στοιχήματος θα πρέπει να διασφαλίσει ότι όταν βρίσκεσαι σε τέτοιους χώρους δεν θα μπορείς να παίξεις κάποιο παιχνίδι .
- **Alpha and Beta Test:** Ελέγχεται ότι ο πελάτης μπορεί να χρησιμοποιήσει το σύστημα με όσο το δυνατόν μικρότερη δυσκολία, κόστος και προβλήματα. Τα προβλήματα που εντοπίζονται σε αυτή τη φάση σχετίζονται με τις συνθήκες και το περιβάλλον όπου το σύστημα θα χρησιμοποιηθεί.

3.5 Είδη ελέγχου λογισμικού

Στον έλεγχο λογισμικού πολλές δραστηριότητες ομαδοποιούνται μεταξύ τους στοχεύοντας σε συγκεκριμένα χαρακτηριστικά του συστήματος. Από αυτές τις ομαδοποιήσεις έχουν προκύψει τα είδη ελέγχου λογισμικού και αυτά είναι οι δοκιμές λειτουργίας, οι δοκιμές που δεν σχετίζονται με τη λειτουργία, οι δομικές δοκιμές και οι δοκιμές που σχετίζονται με τις αλλαγές του συστήματος [1] [24] [25].

Δοκιμές Λειτουργίας (Functional Testing): Απαντούν στο ερώτημα «Τι κάνει το σύστημα». Συνήθως το τι πρέπει να ελεγχθεί περιγράφεται στις απαιτήσεις του συστήματος και ο έλεγχος επικεντρώνεται στις λειτουργίες και τα χαρακτηριστικά του συστήματος καθώς επίσης και στη διαλειτουργικότητα με άλλα συστήματα. Επιπρόσθετα, κάποια άλλα βασικά χαρακτηριστικά που ελέγχονται είναι η ασφάλεια του συστήματος, η ακρίβεια, καθώς επίσης και η συμμόρφωση σε πρότυπα, συμβάσεις, νόμους και γενικά ό,τι αφορά κανόνες που σχετίζονται με τη λειτουργικότητα του συστήματος.

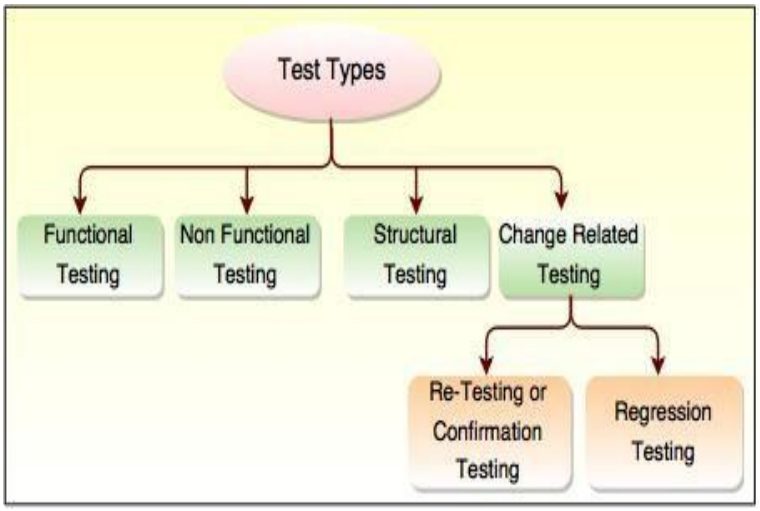
Δοκιμές που δεν σχετίζονται με τη λειτουργία (Non Functional Testing): Απαντούν στο ερώτημα «Πόσο καλά συμπεριφέρεται το σύστημα». Οι συγκεκριμένες δοκιμές είναι απαραίτητο να γίνονται νωρίς ωστόσο αυτό δεν περιορίζει τη χρήση τους και σε μετέπειτα στάδια. Τα πιο βασικά είδη ελέγχου που αφορούν τις δοκιμές που δεν σχετίζονται με τη λειτουργία είναι :

- Έλεγχος απόδοσης του συστήματος (Performance Testing): Μέσω αυτού του ελέγχου εξετάζεται πόσο καλά λειτουργεί το σύστημα σε σχέση με τη σταθερότητά του, την απόδοσή του, και τους πόρους που καταναλώνει.
- Έλεγχος φόρτωσης (Load Testing): Συχνά αυτός ο έλεγχος αναφέρεται και ως επεκτασιμότητα λογισμικού και μέσω αυτού εξετάζεται κατά πόσο το σύστημα μπορεί να λειτουργεί υπό συγκεκριμένο αριθμό δεδομένων ή χρηστών (Συνήθως ο έλεγχος αφορά μεγάλο όγκο δεδομένων ή μεγάλο αριθμό χρηστών).
- Έλεγχος κατά ακραίων καταστάσεων (Stress Testing): Στον έλεγχο αυτό το σύστημα δοκιμάζεται σε καταστάσεις που είναι πέρα από τη συνηθισμένη λειτουργία του. Έτσι εξετάζεται κατά πόσο μπορεί το σύστημα να ανταπεξέλθει σε τέτοιες καταστάσεις και να συνεχίζει να λειτουργεί ομαλά ή τι προβλήματα μπορεί να προκύψουν.
- Έλεγχος ευχρηστίας (Usability Testing): Στις δοκιμές αυτές εξετάζεται πόσο εύχρηστο είναι το σύστημα για τον χρήστη. Ουσιαστικά ελέγχεται κατά πόσο το σύστημα προκαλεί δυσκολίες στον χρήστη για να εκτελέσει κάποια ενέργεια, πόσο εύκολα μπορεί να τον ωθήσει σε κάποιο λάθος και γενικότερα πόσο ικανοποιητικό σε χρήση είναι το λογισμικό από τον τελικό πελάτη (Τελικός πελάτης είναι ο χρήστης).

Δομικές δοκιμές (Structural Testing): Οι δομικές δοκιμές εξετάζουν την εσωτερική δομή του συστήματος. Αυτές οι δοκιμές μπορούν να πραγματοποιηθούν σε οποιοδήποτε επίπεδο δοκιμών, αλλά κυρίως εφαρμόζονται στο επίπεδο «συστατικών» (component testing) και στο επίπεδο ολοκλήρωσης (integration testing). Μέσω αυτών των δοκιμών μετριέται η πληρότητα των ελέγχων, εξετάζοντας την κάλυψη συγκεκριμένων ομάδων στοιχείων της δομής του προγράμματος. Οι πιο βασικές τεχνικές που χρησιμοποιούνται σε αυτό το είδος ελέγχου είναι η τεχνική του λευκού κουτιού και τα μοντέλα ροής ελέγχου που θα αναλυθούν σε επόμενες ενότητες.

Δοκιμές που σχετίζονται με τις αλλαγές του συστήματος (Change Related Testing): Το τελευταίο είδος δοκιμών που θα αναλύσουμε μπορεί να περιλαμβάνει όλους τους παραπάνω τύπους που αναφέραμε (functional, non-functional, structural testing) και οι έλεγχοι εκτελούνται σε όλα τα επίπεδο ελέγχου λογισμικού. Συνήθως οι δοκιμές που σχετίζονται με τις αλλαγές του συστήματος χωρίζονται σε δύο υπό-κατηγορίες, τους ελέγχους επιβεβαίωσης (confirmation testing) και τους ελέγχους παλινδρόμησης (regression testing).

- Έλεγχος επιβεβαίωσης (confirmation testing): Στον έλεγχο αυτό, όταν ένα σφάλμα του συστήματος έχει επιλυθεί, τότε πρέπει ακολουθώντας ακριβώς τα ίδια βήματα που προκάλεσαν αυτό το σφάλμα να ελεγχθεί ότι το σύστημα τώρα λειτουργεί σωστά.
- Έλεγχος παλινδρόμησής (regression testing): Ο έλεγχος αυτός πραγματοποιείται μετά από μια μεγάλη αλλαγή στον κώδικα του συστήματος. Σκοπός αυτού του ελέγχου είναι να βρεθούν προβλήματα στο σύστημα που μπορεί να προέκυψαν από αυτή τη μεγάλη αλλαγή στον κώδικα. Αυτά τα προβλήματα δεν αφορούν μόνο το συγκεκριμένο κομμάτι του κώδικα που άλλαξε, αλλά μπορεί να είναι και προβλήματα που υπήρχαν στο σύστημα, διορθώθηκαν, και εμφανίστηκαν ξανά με τις καινούριες αλλαγές.



Εικόνα 11: Είδη Ελέγχου Λογισμικού [26]

3.6 Κατηγορίες και τεχνικές ελέγχου λογισμικού

Στις προηγούμενες υπό-ενότητες του κεφάλαιού αυτού έχει καταγραφεί αρκετή πληροφορία για το τι είναι έλεγχος λογισμικού, τις 7 βασικές αρχές του καθώς επίσης και ποιες είναι οι φάσεις, τα είδη, και τα επίπεδα ελέγχου λογισμικού. Είναι σημαντικό ωστόσο να αναφερθούμε και σε δυο βασικές κατηγορίες ελέγχου λογισμικού. Υπάρχει ένας βασικός διαχωρισμός μεταξύ των τεχνικών ελέγχου λογισμικού. Έτσι υπάρχει ο δυναμικός έλεγχος (dynamic testing) όπου το πρόγραμμα εκτελείται και αντίστοιχα ο στατικός έλεγχος (static testing) όπου το πρόγραμμα δεν εκτελείται.

3.6.1 Στατικός έλεγχος (Static Testing)

Ως στατικό έλεγχο ορίζουμε την τεχνική όπου ελέγχει για προβλήματα στο λογισμικό χωρίς να εκτελείται ο κώδικας [1] [25] [27] [28]. Ο έλεγχος αυτός γίνεται με σκοπό την αποφυγή σφαλμάτων κυρίως στο πρώιμο στάδιο ανάπτυξης του λογισμικού καθώς είναι πιο εύκολο να εντοπιστούν τυχόν προβλήματα και να επιλυθούν. Υπάρχουν δυο βασικοί τύποι τεχνικών στατικού ελέγχου, οι χειροκίνητες εξετάσεις που περιλαμβάνουν την ανασκόπηση και τα σχόλια (reviews) και η αυτοματοποιημένη ανάλυση που ουσιαστικά είναι η στατική ανάλυση (static analysis) που γίνεται χρησιμοποιώντας κατάλληλα εργαλεία.

Reviews

Ως ανασκόπηση στον στατικό έλεγχο θεωρούμε τη διαδικασία μέσω της οποίας γίνεται προσπάθεια εύρεσης πιθανών προβλημάτων στο σχεδιασμό ενός προγράμματος [1] [25] [27] [28]. Οι συμμετέχοντες στα review είναι συνήθως 5.

- **Ο συντονιστής** που πραγματοποιεί έλεγχο των συμμετεχόντων, παρακολουθεί την πορεία του review και προγραμματίζει τις συναντήσεις μεταξύ των εμπλεκόμενων.
- **Ο συγγραφέας** που αναλαμβάνει την ευθύνη να δημιουργήσει το υπό-review προϊόν αλλά και να διορθώσει τα προβλήματα που θα εντοπιστούν.
- **Ο γραμματέας** που κάνει την καταγραφή των προβλημάτων κατά τη διάρκεια ενός review.
- **Ο κριτής** που κάνει ανασκόπηση του προϊόντος που εξετάζεται και εντοπίζει πιθανά προβλήματα.
- **Ο υπεύθυνος του τμήματος** που αποφασίζει για την εκτέλεση ενός review, διαθέτει τους ανάλογους πόρους για το εκάστοτε review και ουσιαστικά έχει τον πρώτο και τον τελευταίο λόγο στην οργάνωση της συγκεκριμένης διαδικασίας.

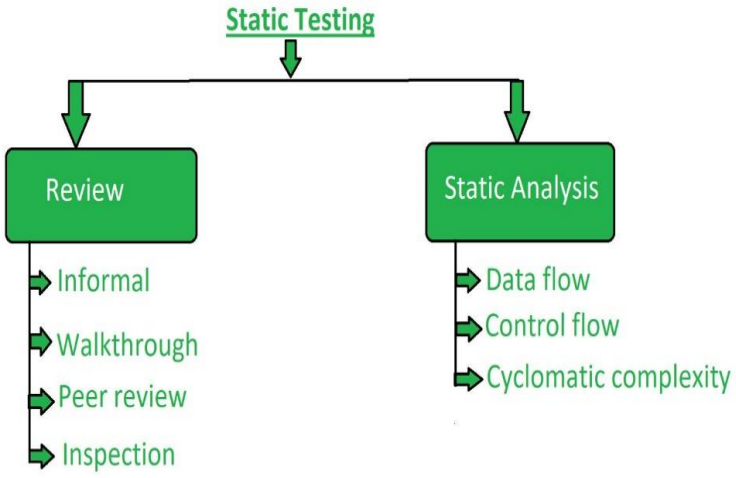
Τα reviews με την σειρά τους μπορούν να κατηγοριοποιηθούν σε 4 νέες κατηγορίες ανάλογα και το πόσο επίσημα ή μη επίσημα είναι. Αυτές είναι: Informal Review, Walkthrough, Technical Review και Inspection.

- **Informal Review:** Στόχος αυτού του review είναι να εντοπιστούν πιθανά προβλήματα στο σύστημα. Αποτελεί, όπως είναι εύκολα κατανοητό από την ονομασία του, ένα μη επίσημο τρόπο review και δεν βασίζεται σε επίσημα έγραφα και διαδικασίες. Αποτελεί πολύ συχνή διαδικασία στο μοντέλο Agile και συνήθως εκτελείται μεταξύ των συναδέλφων που συμμετέχουν στη δημιουργία και στον έλεγχο ενός λογισμικού.
- **Walkthrough:** Στόχος αυτού του review είναι να εντοπιστούν πιθανά προβλήματα στο σύστημα αλλά και να βελτιωθεί το λογισμικό αναζητώντας εναλλακτικές λύσεις. Στο συγκεκριμένο τύπο review ο γραμματέας είναι υποχρεωτικό να υπάρχει.
- **Technical Review:** Στόχος αυτού του review είναι να εντοπιστούν πιθανά προβλήματα στο σύστημα, να υπάρξει αυτοπεποίθηση για το προϊόν που παράγεται, να αναπτυχθούν νέες ιδέες και γενικά να υπάρξει μελλοντική βελτίωση του προϊόντος αναζητώντας εναλλακτικές ιδέες και λύσεις. Για τον συγκεκριμένο τύπο review απαιτείται προετοιμασία πριν από κάθε συνάντηση μεταξύ των reviewers και ο κάθε reviewer θα πρέπει να είναι γνώστης του αντικειμένου. Όπως και στο walkthrough, ο ρόλος του γραμματέα είναι υποχρεωτικό να υπάρχει.

- **Inspection :** Αποτελεί τον πιο επίσημο τύπο review και έχει ως σκοπό να εντοπιστούν πιθανά προβλήματα στο σύστημα και να αξιολογηθεί η ποιότητα του συστήματος. Μέσω αυτού του review δίνεται κίνητρο στους εμπλεκόμενους ρόλους να βελτιώσουν την ποιότητα και τις διαδικασίες που χρησιμοποιούν για τη δημιουργία ενός προϊόντος. Το συγκεκριμένο review ακολουθεί συγκεκριμένες διαδικασίες και όλοι οι ρόλοι είναι υποχρεωτικό να υπάρχουν. Επίσης, συγκεκριμένα κριτήρια εισόδου και εξόδου χρησιμοποιούνται. Τέλος εκτός από τα πιθανά προβλήματα που εντοπίζονται και καταγράφονται, συλλέγονται και στατιστικές μετρήσεις με σκοπό να βελτιωθούν οι διαδικασίες στο μέλλον.

Static Analysis

Ως στατική ανάλυση ορίζουμε τη διαδικασία μέσω της οποίας αναλύονται τα τεχνικά «artifacts» του λογισμικού χωρίς την εκτέλεσή τους [1] [25] [27] [28]. Συνήθως η στατική ανάλυση εκτελείται πριν τα επίσημα review στο σύστημα. Υπάρχουν τρεις βασικοί τύποι στατιστικής ανάλυσης, η ανάλυση ροής δεδομένων που εξετάζει πιθανές ανωμαλίες στο σύστημα στα κομμάτια όπου ο κώδικας χρησιμοποιεί δεδομένα και μεταβλητές, η ανάλυση ροής ελέγχου που εξετάζει τη ροή που έχει ο κώδικας και μπορεί να εντοπίσει προβλήματα όπως βρόγχους που εκτελούνται επ' άπειρον ή κώδικα που το πρόγραμμα δεν φτάνει ποτέ να τον εκτελέσει και η κυκλωματική πολυπλοκότητα (cyclomatic complexity) που εξετάζει την πολυπλοκότητα του προγράμματος. Τα πλεονεκτήματα της στατικής ανάλυσης είναι ότι εντοπίζονται νωρίς τα προβλήματα του συστήματος πριν ακόμα εκτελεστούν, παρατηρούνται νωρίς «ύποπτες» λειτουργίες στον κώδικα που μπορεί να προκαλέσουν προβλήματα στο μέλλον, εντοπίζει προβλήματα που είναι δύσκολο να εντοπιστούν στον δυναμικό έλεγχο που θα αναλύσουμε παρακάτω και γενικά βελτιώνει τη συντήρηση του κώδικα.

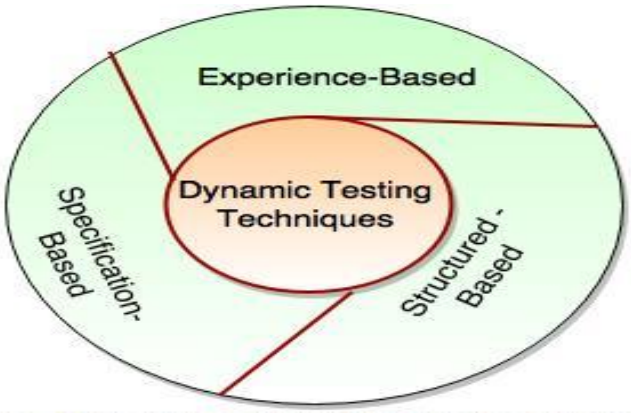


Εικόνα 12: Τεχνικές Στατικού Ελέγχου [29]

3.6.2 Δυναμικός έλεγχος (Dynamic Testing)

Ως δυναμικό έλεγχο ορίζουμε την τεχνική όπου ελέγχει για προβλήματα στο λογισμικό εκτελώντας τον πηγαίο κώδικα [1] [25] [27] [28]. Μέσω του δυναμικού ελέγχου μπορούν να

εντοπιστούν προβλήματα που δεν εντοπίζονται εύκολα μέσω του στατικού ελέγχου που αναφέραμε παραπάνω. Επίσης, επειδή εκτελείται ο κώδικας και γενικά το λογισμικό από την αρχή μέχρι το τέλος διασφαλίζεται καλύτερη ποιότητα στο τελικό παραδοτέο. Ο δυναμικός έλεγχος χωρίζεται σε 3 κατηγορίες τεχνικών ελέγχου, τις τεχνικές του μαύρου κουτιού (Black Box Techniques), τις τεχνικές του άσπρου κουτιού (White Box Techniques) και τις τεχνικές που βασίζονται στην εμπειρία του δοκιμαστή (Experience Based Techniques) [30].



Εικόνα 13: Τεχνικές Δυναμικού Ελέγχου [31]

Τεχνική Μαύρου Κουτιού (Black Box Technique)

Γνωστή και ως τεχνική που βασίζεται στις προδιαγραφές, επειδή τα δεδομένα και οι έλεγχοι που εκτελούνται βασίζονται στις προδιαγραφές του συστήματος και στα “Business User Stories” που έχουν καταγραφεί από την ομάδα των αναλυτών. Η συγκεκριμένη τεχνική επικεντρώνεται στο δίπολο είσοδος-έξοδος του συστήματος. Ουσιαστικά ελέγχει τι δεδομένα «εισέρχονται» στο σύστημα και ποιο είναι το αποτέλεσμα που παράγεται. Υπάρχουν πολλές τεχνικές που βασίζονται στην τεχνική αυτή. Κάποιες από τις βασικές είναι οι παρακάτω.

- **Κατανομή ισοδυναμίας (equivalence partitioning):** Η βασική ιδέα αυτής της τεχνικής είναι να χωρίζονται τα δεδομένα σε κλάσεις (partitions) υποθέτοντας ότι το σύστημα για κάθε κλάση έχει ίδια συμπεριφορά. Έτσι για κάθε κλάση υπάρχουν οι αποδεκτές τιμές που θα πρέπει το σύστημα να ανταποκρίνεται και οι μη αποδεκτές που το σύστημα θα πρέπει να βγάζει κάποιο μήνυμα λάθους. Έτσι αν μια τυχαία αποδεκτή τιμή δουλεύει σωστά, τότε υποθέτουμε ότι όλες οι τιμές που είναι αποδεχτές από αυτή την κλάση θα δουλεύουν και αυτές σωστά. Για παράδειγμα αν θέλουμε να ελέγξουμε ότι ένας κωδικός δέχεται από 5 έως 10 χαρακτήρες τότε αρκεί να ελέγξουμε μια τυχαία ακολουθία χαρακτήρων μέσα σε αυτό το εύρος (π.χ. 7).
- **Ανάλυση οριακής τιμής (boundary value analysis):** Η συγκεκριμένη τεχνική αποτελεί μια επέκταση της κατανομής ισοδυναμίας και μέσω αυτής ελέγχονται οι ακραίες τιμές μιας κλάσης. Ουσιαστικά ο έλεγχος επικεντρώνεται στη μικρότερη και στη μεγαλύτερη δυνατή τιμή που μπορεί να λάβει το σύστημα σαν είσοδο καθώς επίσης και τιμές που είναι κοντά στις επιτρεπτές τιμές. Έτσι στο προηγούμενο παράδειγμα με τον κωδικό αν εφαρμόζαμε αυτή την τεχνική θα ελέγχαμε κωδικούς με 5 και 10 χαρακτήρες (μικρότερη

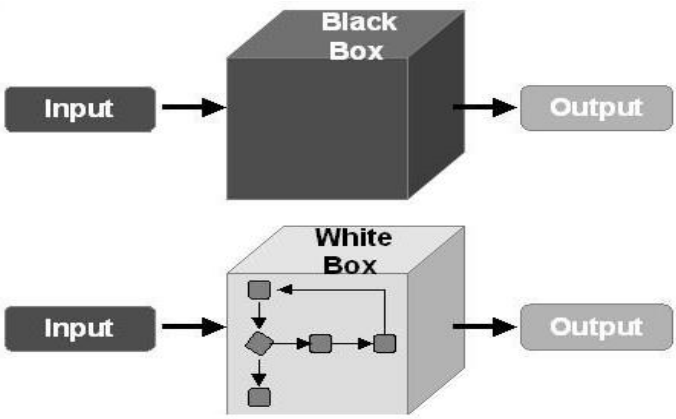
και μεγαλύτερη τιμή) καθώς επίσης και κωδικούς με 4 και 11 χαρακτήρες (κοντινές ακραίες τιμές).

- **Δοκιμή πίνακα αποφάσεων (decision table testing):** Συνδυαστική τεχνική που είναι ιδιαίτερα χρήσιμη σε συστήματα όπου ανάλογα την κάθε κατάσταση μπορεί να υπάρχει διαφορετικό αποτέλεσμα. Πρώτο βήμα αυτής της τεχνικής είναι ο εντοπισμός όλων των εισόδων και εξόδων που μπορεί να έχει το σύστημα και στη συνέχεια η καταγραφή τους σε ένα πίνακα. Έπειτα ανάλογα το συνδυασμό, ελέγχεται ποιοι συνδυασμοί μπορούν να πραγματοποιηθούν και ποιοι όχι και καταγράφονται με True ή False.
- **Δοκιμή μετάβασης κατάστασης (state transition testing):** Μέσω αυτής της τεχνικής εξετάζεται κάθε κατάσταση του λογισμικού και τα πιθανά αποτελέσματα που παράγονται ανάλογα με την επιλογή που θα κάνει ο χρήστης. Χρησιμοποιείται κυρίως σε λογισμικά με περιορισμένο αριθμό καταστάσεων που είναι προκαθορισμένα.
- **Δοκιμή περιπτώσεων χρήστη (use case testing):** Αποτελεί μια τεχνική που βοηθά στον εντοπισμό δοκιμαστικών περιπτώσεων που καλύπτουν ολόκληρο το σύστημα, από δραστηριότητα σε δραστηριότητα, από την αρχή έως το τέλος. Είναι μια περιγραφή μιας συγκεκριμένης χρήσης του συστήματος από έναν χρήστη. Επειδή ακριβώς ακολουθείται η πορεία ενός χρήστη στο σύστημα, είναι μια πολύ καλή τεχνική για να εντοπίζονται προβλήματα που έχουν εφαρμογή στον «πραγματικό κόσμο» του συστήματος.

Τεχνική Άσπρου Κουτιού (White Box Technique)

Γνωστή και ως τεχνική που βασίζεται στη δομή του συστήματος επειδή οι έλεγχοι που εκτελούνται αφορούν την εσωτερική δομή του συστήματος και τον κώδικα του λογισμικού. Οι έλεγχοι που πραγματοποιούνται καθώς και τα δεδομένα που χρησιμοποιούνται προέρχονται από την αρχιτεκτονική του συστήματος, τον σχεδιασμό του συστήματος καθώς επίσης και τον πηγαίο κώδικα του λογισμικού. Όπως και στην τεχνική του μαύρου κουτιού έτσι και εδώ υπάρχουν αρκετές τεχνικές που βασίζονται στην εσωτερική δομή του συστήματος. Δύο από τις πιο βασικές είναι οι παρακάτω.

- **Δοκιμή και κάλυψη δηλώσεων (Statement testing and coverage):** Αποτελεί μια δοκιμή που οι έλεγχοι σχεδιάζονται με σκοπό την εκτέλεση καταστάσεων του συστήματος ανά στοιχείο (component).
- **Δοκιμή και κάλυψη αποφάσεων (Decision testing and coverage):** Αποτελεί μια εξέλιξη της δοκιμής και κάλυψης δηλώσεων καθώς μέσω αυτής ελέγχονται οι αποφάσεις και τα πιθανά αποτελέσματα στον κώδικα. Για παράδειγμα σε μια FOR LOOP θα εξεταστεί τι γίνεται αν ο κώδικας εισέρθει στην επανάληψη αυτή ή όχι και ποια αποτελέσματα προκύπτουν.



Εικόνα 14: Black Box VS White Box [32]

Τεχνική που βασίζεται στην εμπειρία του tester (Experience Based Technique)

Η τελευταία μεγάλη κατηγορία τεχνικών ελέγχου λογισμικού που αποτελεί μέρος του δυναμικού ελέγχου είναι η τεχνική που βασίζεται στην εμπειρία του tester. Αναλυτικότερα, η τεχνική αυτή βασίζεται στις γνώσεις, στην εμπειρία, στη φαντασία και στη διαίσθηση του tester και είναι προτεινόμενο να χρησιμοποιείται συμπληρωματικά σε πιο «επίσημες τεχνικές» όπως αυτές που ήδη έχουμε αναφέρει παραπάνω καθώς δεν είναι εύκολο να καλυφθούν πολλές περιοχές ελέγχου χρησιμοποιώντας απλά την εμπειρία κάποιου. Αν και δεν είναι εύκολο να καλυφθούν πολλές περιοχές ελέγχου, αυτή η τεχνική μπορεί να φανεί ιδιαίτερα χρήσιμη στον εντοπισμό προβλημάτων που δεν μπορούν εύκολα να εντοπιστούν με τις τεχνικές του μαύρου και του άσπρου κουτιού. Όπως και στις 2 προηγούμενες τεχνικές έτσι και εδώ υπάρχουν αρκετές τεχνικές που βασίζονται στην εμπειρία του tester. Οι πιο βασικές είναι:

- **Πρόβλεψη λαθών (error guessing):** Τεχνική που βασίζεται στις γνώσεις και την εμπειρία του tester. Αναλυτικότερα ο έμπειρος tester γνωρίζοντας πως λειτουργούσε το σύστημα στο παρελθόν, τι λάθη συνήθως κάνουν οι προγραμματιστές και προβλήματα από άλλα λογισμικά, μπορεί να ελέγξει το σύστημα και να εντοπίσει πιθανά προβλήματα του. Μέσω αυτής της τεχνικής δεν υπάρχουν κανόνες και ο tester είναι χρήσιμο να ελέγχει σενάρια που δεν έχουν καμία επιχειρηματική λογική ωστόσο μπορούν να προκαλέσουν πρόβλημα στο σύστημα. Ένα απλό παράδειγμα είναι ο tester να πάει να τραβήξει λεφτά από ένα τραπεζικό σύστημα και σαν ποσό να επιλέξει αρνητικό ή μηδέν.
- **Διερευνητικές δοκιμές (exploratory testing)**: Η τεχνική αυτή αποτελεί μια άτυπη τεχνική ελέγχου λογισμικού και συνήθως χρησιμοποιείται όταν υπάρχει πίεση χρόνου ή όταν οι προδιαγραφές είναι λίγες. Μέσω αυτής ο tester απλά περιηγείται στο σύστημα εκτελώντας βασικές λειτουργίες του συστήματος και ελέγχει αν όλα δουλεύουν σωστά. Πολλές εταιρίες χρησιμοποιούν αυτή την τεχνική ως ένα είδους εκπαίδευσης προς τους νέους testers.
- **Δοκιμές βάσει λίστας ελέγχου (checklist based testing):** Η συγκεκριμένη τεχνική όπως μπορεί εύκολα να κατανοηθεί και από την ονομασία της βασίζεται σε λίστες ελέγχου που έχουν δημιουργηθεί από τους testers. Έτσι ο tester εκτελεί ελέγχους στο σύστημα ανάλογα με τη λίστα και ότι σενάρια αυτή περιλαμβάνει.

4. ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΣ ΕΛΕΓΧΟΣ ΛΟΓΙΣΜΙΚΟΥ

Στο κεφάλαιο αυτό το οποίο αποτελεί την τελευταία θεωρητική ενότητα αυτής της εργασίας θα αναλυθεί η έννοια του αυτοματοποιημένου ελέγχου λογισμικού, τα πλεονεκτήματα και τα μειονεκτήματα που υπάρχουν σε σχέση με το μη αυτοματοποιημένο έλεγχο (manual testing), και τέλος θα παρουσιαστούν κάποια εργαλεία αυτοματοποιημένου ελέγχου λογισμικού.

4.1 Τι είναι ο αυτοματοποιημένος έλεγχος λογισμικού

Όπως ήδη έχει αναφερθεί σε προηγούμενες ενότητες, τα λογισμικά που παράγονται την εποχή μας έχουν μεγάλη πολυπλοκότητα, ωστόσο η τελική ποιότητα πρέπει να είναι όσο το δυνατόν καλύτερη γίνεται. Για το λόγο αυτό προέκυψε μια νέα ανάγκη για νέους τρόπους ελέγχου λογισμικού διαφορετικούς από τον παραδοσιακό «χειρωνακτικό» τρόπο. Έτσι προέκυψε η ανάγκη του αυτοματοποιημένου ελέγχου λογισμικού [33]. Πριν όμως αναλύσουμε αυτή την έννοια και για να προσπαθήσουμε να την κάνουμε ακόμα πιο κατανοητή, θα αναλύσουμε πρώτα τι είναι ο μη αυτοματοποιημένος έλεγχος λογισμικού ή manual testing.

Ως manual testing ορίζουμε κάθε δοκιμή ελέγχου που πραγματοποιείται χειρωνακτικά από έναν tester με σκοπό την εύρεση ελαττωμάτων στο σύστημα [34]. Ο tester πρέπει να τρέξει πολλαπλά σενάρια για να ελέγχει αν η ποιότητα του συστήματος είναι η επιθυμητή σύμφωνα με τις προδιαγραφές του συστήματος και τις απαιτήσεις του πελάτη. Ουσιαστικά ο manual tester έχει το ρόλο του τελικού χρήστη στο σύστημα. Για παράδειγμα, σε μια απλή φόρμα για είσοδο σε μια ιστοσελίδα, ο tester θα ελέγξει αν μπορεί με συγκεκριμένους κωδικούς να εισέλθει στο σύστημα και αντίστοιχα με μη αποδεκτούς κωδικούς αν δεν μπορεί να εισέλθει στο σύστημα.

Ωστόσο, τα σύγχρονα λογισμικά έχουν τεράστια πολυπλοκότητα και οι έλεγχοι δεν είναι τόσο απλοί [34]. Για παράδειγμα σε ένα σύστημα καταγραφής κλινικών δοκιμών για τη δημιουργία μιας κλινικής δοκιμής μπορεί να χρειάζεται να συμπληρωθούν από το χρήστη δεκάδες πεδία. Σε ένα τέτοιο σενάριο ο έλεγχος λογισμικού απαιτεί από το manual tester πολλές κινήσεις που είναι χρονοβόρες και ειδικά όταν αυτές πρέπει να επαναλαμβάνονται συχνά τότε αυτόματα καταλαβαίνουμε ότι ο «χειρωνακτικός» έλεγχος δεν είναι ο ιδανικός. Για το λόγο αυτό δημιουργήθηκαν και δημιουργούνται ειδικά λογισμικά που επιτρέπουν τον αυτοματοποιημένο έλεγχο σε ένα λογισμικό. Μέσω αυτών των λογισμικών δημιουργούνται αλληλουχίες αυτοματοποιημένων κινήσεων, ή αλλιώς scripts όπως συνήθως ονομάζονται, με σκοπό να ελεγχθεί η πραγματική ποιότητα του λογισμικού σε σχέση με την αναμενόμενη. Τα αποτελέσματα αυτών των ελέγχων καταγράφονται και μπορεί μετά η ομάδα ελέγχου να δει ποια σενάρια πέρασαν και ποια σενάρια απέτυχαν. Τα σενάρια που πέρασαν συνήθως καταγράφονται ως “pass” στην τελική αναφορά του tester, ενώ τα σενάρια που απέτυχαν ως “fail”. Αν δημιουργηθούν τα scripts μετά είναι πολύ εύκολο να εκτελούνται οποιαδήποτε στιγμή απαιτηθεί, οπότε επιτυγχάνεται ένας πολύ πιο γρήγορος και επαναλαμβανόμενος έλεγχος σε ένα λογισμικό.

Φυσικά προκύπτουν πολλά ερωτήματα όπως για παράδειγμα «τι» πρέπει να αυτοματοποιηθεί και «αν» χρειάζεται να αυτοματοποιηθεί [33] [34]. Η απάντηση σε αυτά τα ερωτήματα εξαρτάται

από το προϊόν που είναι υπό εξέταση καθώς και τη στρατηγική που θα λάβει η διοίκηση της κάθε εταιρίας. Για παράδειγμα, σε ένα συμβόλαιο μικρής διάρκειας για μια απλή εφαρμογή ο αυτοματοποιημένος έλεγχος πιθανότατα δεν χρειάζεται και δεν θα βοηθήσει στη διαδικασία ελέγχου λογισμικού. Αντίθετα σε ένα μεγάλο έργο μεγάλης διάρκειας και μεγάλης πολυπλοκότητας, ο αυτοματοποιημένος έλεγχος θεωρείται πλέον απαραίτητος και αποτελεί αναπόσπαστο κομμάτι της διαδικασίας ελέγχου λογισμικού. Τέλος, το πιο σημαντικό ερώτημα που προκύπτει είναι αν ο αυτοματοποιημένος έλεγχος μπορεί να καλύψει και να εξαλείψει τον «χειρωνακτικό» έλεγχο. Η απάντηση σε αυτό το ερώτημα είναι πως κανείς δεν μπορεί να πει με βεβαιότητα ότι αυτό μπορεί να επιτευχθεί. Το ανθρώπινο μάτι πάντα μπορεί να εντοπίσει κάτι που ένα script δεν θα καταφέρει να εντοπίσει στο δικό του έλεγχο. Ωστόσο ένας σωστά δομημένος και υλοποιημένος αυτοματοποιημένος έλεγχος μόνο θετικά οφέλη μπορεί να προσφέρει σε μια εταιρία και να βελτιώσει τη συνολική ποιότητα του υπο-ελέγχου λογισμικού.

4.2 Πλεονεκτήματα και μειονεκτήματα

Στη συγκεκριμένα υπό-ενότητα του κεφαλαίου αυτού παρουσιάζονται αναλυτικά τα πλεονεκτήματα και τα μειονεκτήματα του αυτοματοποιημένου ελέγχου έτσι ώστε ο αναγνώστης να μπορεί να δημιουργήσει μια συνολική εικόνα του τι μπορεί και τι δεν μπορεί να προσφέρει ο έλεγχος αυτός [33] [34] [35].

Πλεονεκτήματα

- **Μείωση του χρόνου ελέγχου:** Οι αυτοματοποιημένοι έλεγχοι μπορεί να εκτελεστούν σε σύντομο χρονικό διάστημα και να κάνουν πολλαπλές ενέργειες που ένας manual tester θα χρειαζόταν ώρες ή και μέρες για να πραγματοποιήσει.
- **Αύξηση της κάλυψης των δοκιμών:** Οι αυτοματοποιημένοι έλεγχοι εκτελούν πολύπλοκα και χρονοβόρα σενάρια σε σύντομο χρονικά διάστημα που μπορεί κατά τη διάρκεια περιόδων που δεν υπάρχει ο απαιτούμενος χρόνος, αυτά τα σενάρια να αποφεύγονται από την ομάδα των manual testers και να μην εκτελούνται. Έτσι με το να υπάρχουν οι αυτοματοποιημένοι έλεγχοι αυτόματα καλύπτονται και περισσότερα σενάρια δοκιμών που πριν δεν θα εκτελούνταν λόγω πίεσης χρόνου ή πολυπλοκότητας.
- **Πραγματοποιούνται έλεγχοι που δεν μπορούν να πραγματοποιηθούν με άλλο τρόπο:** Μέσω των αυτοματοποιημένων ελέγχων μπορούν να εκτελεστούν και έλεγχοι επίδοσης του συστήματος (performance testing). Για παράδειγμα, ένα script μπορεί να συνδέει σε ένα σύστημα 500 χρήστες ταυτόχρονα. Αυτό δεν είναι εφικτό με το χειρωνακτικό έλεγχο. Επιπρόσθετα, μέσω των εργαλείων του αυτοματοποιημένου ελέγχου λογισμικού μπορούν να ελεγχθούν παραπάνω από ένα προγράμματα περιήγησης με σχετικά απλές εντολές στον κώδικα. Για παράδειγμα, φανταστείτε μια εφαρμογή που έχει 500 σενάρια ελέγχου που πρέπει να τρέξουν εκτός από το Google Chrome και σε Internet Explorer, Mozilla Firefox και Opera. Αυτό απαιτεί πρακτικά 2000 σενάρια ελέγχου για ένα μόνο έλεγχο μιας έκδοσης. Άρα σε κάθε νέα έκδοση, και σύμφωνα με τον έλεγχο παλινδρόμησης που έχουμε ήδη αναφέρει θα πρέπει αυτά τα 2000 σενάρια ελέγχου να ξανά εκτελεστούν. Είναι αυτονόητο ότι σε μια τέτοια περίπτωση το manual testing δεν θα είναι αποδοτικό και δεν θα μπορέσει να καλύψει όλα τα σενάρια. Αντίθετα ο αυτοματοποιημένος έλεγχος λογισμικού μπορεί να

ανταπεξέλθει σε αυτές τις απαιτήσεις και να εκτελέσει σε σύντομο χρονικό διάστημα τα σενάρια ελέγχου που απαιτούνται.

- **Αμεσότητα προς την ομάδα των προγραμματιστών:** Με το να υπάρχουν έλεγχοι αυτοματοποιημένοι, μπορεί η ομάδα των προγραμματιστών οποιαδήποτε στιγμή να ζητήσει να ελεγχθούν κάποια βασικά σενάρια σε κάποια έκδοση του συστήματος πριν αυτή η έκδοση εγκατασταθεί στο περιβάλλον της ομάδας του ελέγχου λογισμικού. Αυτό παρατηρείται συχνά όταν εκτελούνται οι αυτοματοποιημένοι έλεγχοι στο περιβάλλον των προγραμματιστών για να βεβαιωθούν ότι βασικές λειτουργίες της εφαρμογής δεν έχουν χαλάσει μετά από νέες βελτιώσεις του συστήματος. Έτσι μπορούν να εντοπιστούν γρήγορα βασικά προβλήματα της εφαρμογής και να κερδηθεί χρόνος για την επίλυση τους.
- **Αύξηση του ηθικού της ομάδας:** Με την ύπαρξη αυτοματοποιημένων ελέγχων όλες οι ομάδες νιώθουν πιο ασφαλείς για την ποιότητα του συστήματος. Επιπρόσθετα δίνεται χρόνος στους manual testers να ασχοληθούν με σενάρια που είτε είναι ακραία είτε δεν μπορούν να αυτοματοποιηθούν και έτσι αυξάνεται ακόμα περισσότερο η ποιότητα του συστήματος.
- **Επαναληπτικός έλεγχος οποιαδήποτε ώρα χρειαστεί:** Το τελευταίο πλεονέκτημα που θα αναφερθεί στην ενότητα αυτή είναι ότι οι αυτοματοποιημένοι έλεγχοι λογισμικού μπορούν να τρέξουν οποιαδήποτε ώρα απαιτηθεί αρκεί να έχουν προγραμματιστεί. Για παράδειγμα μπορεί να χρειάζεται να ελεγχθεί ένα σενάριο στην αλλαγή της μέρας (π.χ. 23:59 μ.μ.). Αυτό αν έπρεπε να γίνει χειροκίνητα θα χρειαζόταν ένας άνθρωπος να εργαστεί εκείνη την ώρα με ότι συνέπειες αυτό μπορεί να έχει για την εταιρία αλλά και τον ίδιο τον εργαζόμενο. Αντίθετα αν έχει προγραμματιστεί ο αυτοματοποιημένος έλεγχος, τότε αυτός θα τρέξει εκείνη την ώρα και την επόμενη μέρα θα μπορεί η ομάδα ελέγχου να δει τα αποτελέσματα μέσω των αποτελεσμάτων που θα έχουν καταγραφεί.

<u>Μειονεκτήματα</u>

- **Απαιτεί άτομα με εμπειρία και δεξιότητες:** Ο αυτοματοποιημένος έλεγχος δεν είναι μια απλή διαδικασία και χρειάζεται άτομα με κατάλληλες γνώσεις και δεξιότητες που θα μπορούν να δημιουργήσουν τα αυτοματοποιημένα σενάρια. Η έλλειψη των απαραίτητων γνώσεων μπορεί να δημιουργήσει λανθασμένα αποτελέσματα και αυτό να δημιουργήσει μεγάλο πρόβλημα στην τελική ποιότητα του συστήματος.
- **Απαιτείται χρόνος για τη δημιουργία των ελέγχων:** Η δημιουργία των αυτοματοποιημένων ελέγχων είναι μια χρονοβόρα διαδικασία. Η ομάδα που θα αναλάβει αυτό το σκοπό πρέπει να δει τι εργαλεία θα χρησιμοποιήσει, να εκπαιδεύσει την ομάδα πάνω σε αυτά, να κατανοήσει πλήρως το σύστημα που θα εκτελεστούν οι έλεγχοι αυτοί και τέλος να προγραμματίσει τα scripts που θα αυτοματοποιούν τους επιθυμητούς ελέγχους.
- **Απαιτείται συντήρηση:** Όπως ήδη αναφέρθηκε, η δημιουργία των scripts είναι μια χρονοβόρα διαδικασία. Ωστόσο ακόμα και μετά τη δημιουργία τους, μια μικρή αλλαγή στο λογισμικό που ελέγχεται μπορεί να προκαλέσει μια αλληλουχία προβλημάτων στα scripts. Έτσι είναι απαραίτητο να γίνει συντήρηση και ανανέωση των scripts έτσι ώστε να ακολουθούν την υλοποίηση του λογισμικού που ελέγχεται. Κάθε αλλαγή στο λογισμικό μικρή ή μεγάλη επηρεάζει άμεσα ή έμμεσα τον αυτοματοποιημένο έλεγχο και

αυτό έχει σαν αποτέλεσμα να πρέπει να ελεγχθούν και να αλλαχθούν σενάρια που με τις νέες αλλαγές πλέον δεν λειτουργούν.

- **Δεν εντοπίζει όλα τα ακραία σενάρια:** Η δημιουργία ενός script έχει ένα συγκεκριμένο πεδίο εφαρμογής και εξετάζει κάποια συγκεκριμένα βήματα. Αντίθετα ο manual tester πραγματοποιώντας το ίδιο σενάριο χειρωνακτικά, μπορεί να εντοπίσει και άλλα προβλήματα που ένα συγκεκριμένος και επαναλαμβανόμενος έλεγχος δεν θα εντοπίσει.

4.3 Εργαλεία αυτοματοποιημένου ελέγχου λογισμικού

Τα εργαλεία αυτοματοποιημένου ελέγχου λογισμικού στην εποχή μας είναι πολλά και αυτό δείχνει πως ο αυτόματος έλεγχος τείνει να γίνει απαραίτητος σε όλες τις εταιρίες. Υπάρχουν εργαλεία που κα΄ποιος πρέπει να αγοράσει για να τα χρησιμοποιήσει, ωστόσο στα πλαίσια αυτής της εργασίας ερευνήθηκαν εργαλεία ανοιχτού λογισμικού (open source tools) που είναι δωρεάν και ελεύθερα να χρησιμοποιηθούν από όλους.

Κάθε εργαλείο μπορεί να ελέγχει κάτι διαφορετικό σε μια εφαρμογή. Υπάρχουν εργαλεία για αυτοματοποιημένους ελέγχους μονάδας (Unit Testing), εργαλεία για αυτοματοποιημένους ελέγχους αποδοχής (Acceptance Testing) αλλά και εργαλεία που ελέγχουν την απόδοση του συστήματος (Performance testing) [33].

4.3.1 Εργαλεία για αυτοματοποιημένους ελέγχους μονάδας

Τα συγκεκριμένα εργαλεία δεν θα μας απασχολήσουν σε αυτή την εργασία καθώς δεν θα γίνει κάποιος έλεγχος μονάδας ωστόσο θα αναφερθούν τα πιο διαδεδομένα για να υπάρχει μια πλήρη καταγραφή των πιο διαδεδομένων εργαλείων ανοιχτού λογισμικού που υπάρχουν αυτή τη στιγμή στην αγορά.

Nunit [36]

Ένα από τα πιο διαδεδομένα εργαλεία ανοιχτού κώδικα και είναι γραμμένο σε C#. Υποστηρίζει όλες τις γλώσσες .NET και μέσω αυτού μπορούν να τρέξουν παράλληλες δοκιμές. Ανήκει στην εταιρία Microsoft.

TestNG [37]

Η πλήρη ονομασία του είναι Test Next Generation. Υποστηρίζει όλες τις γλώσσες .NET και τη γλώσσα Java. Αποτελεί ένα προηγμένο εργαλείο δοκιμών που η βάση του είναι το Nunit που αναφέρθηκε παραπάνω. Είναι πολύ εύχρηστο καθώς υποστηρίζει πολλά πρόσθετα εργαλεία όπως το Eclipse, το Maven και το IDEA.

Mockito [38]

Το συγκεκριμένο εργαλείο έχει γραφτεί σε γλώσσα Java και ανήκει στο MIT (Massachusetts Institute of Technology). Έχει ως στόχο να απλοποιήσει την ανάπτυξη ενός ελέγχου και μπορεί να χρησιμοποιηθεί μαζί με τα άλλα δύο εργαλεία που προαναφέρθηκαν.

4.3.2 Εργαλεία για αυτοματοποιημένους ελέγχους αποδοχής

Αυτά τα εργαλεία θα μας απασχολήσουν περισσότερο καθώς μέσω αυτών θα προσπαθήσουμε στο επόμενο κεφάλαιο να εκτελέσουμε αυτοματοποιημένους ελέγχους λογισμικού.

Selenium [39]

Το Selenium είναι το πιο διαδεδομένο εργαλείο ανοιχτού κώδικα αυτή τη στιγμή στην αγορά. Προσφέρει πολλές επιλογές όπως για παράδειγμα το Selenium IDE που υπάρχει σαν plug-in σε όλους σχεδόν τους web browsers και μέσω αυτού πολύ εύκολα κάποιος χρήστης χωρίς να γνωρίζει κάποια γλώσσα προγραμματισμού μπορεί να αυτοματοποιήσει ενέργειες που πραγματοποιούνται σε γραφικά περιβάλλοντα λογισμικών. Εκτός από αυτό όμως υπάρχει και το Selenium Web driver που είναι ένα πιο ολοκληρωμένο εργαλείο για τη δημιουργία δοκιμών, απαιτεί γνώση προγραμματισμού και μπορεί να χρησιμοποιηθεί με τις περισσότερες γλώσσες (π.χ. Java, Python, Ruby, C#).

Ένα απλό παράδειγμα σε γλώσσα Java που ανοίγει την ιστοσελίδα του Πανεπιστημίου Πειραιώς στο πρόγραμμα περιήγησης Google Chrome είναι το παρακάτω:

```
public class Third {


   public static void main(String[] args) {


        // Εδώ δείχνουμε το Path που βρίσκεται ο WebDriver  του Google Chrome στον υπολογιστή μας.
      System.setProperty("webdriver.chrome.driver", "D:\\ChromeDriver\\chromedriver.exe");


        // Εδώ δημιουργούμε ένα αντικείμενο της κλάσης  ChromeDriver
      WebDriver driver=new ChromeDriver();
```

Robot Framework [40]

Το συγκεκριμένο εργαλείο αναπτύχθηκε και υποστηρίζεται από την εταιρία Nokia Networks. Αποτελεί ένα ιδανικό εργαλείο για τη μετάβαση ενός manual tester σε automation tester, καθώς ακολουθεί την τεχνική λέξεων-κλειδιών που είναι αρκετά κατανοητή. Υπάρχουν πολλές διαθέσιμες βιβλιοθήκες ελέγχου που επεκτείνουν τις δυνατότητες αυτού του εργαλείου. Αυτές οι βιβλιοθήκες είναι γραμμένες σε Python ή Java και μια από τις πιο διαδεδομένες είναι η SeleniumLibrary που συσχετίζεται με το Selenium WebDriver που αναφέρθηκε παραπάνω. Τέλος το συγκεκριμένο εργαλείο υποστηρίζεται και από πολλά Integrated Development Environments (IDE), όπως το Robot Integrated Development Environment (RIDE) ή το PyCharm. Μέσω των IDE παρέχεται στο χρήστη μεγάλη διευκόλυνση στον τρόπο γραφής των

περιπτώσεων ελέγχων καθώς υπάρχει η αυτόματη συμπλήρωση κώδικα και ο έλεγχος του συντακτικού όταν κάποιο προγραμματιστικό λάθος γίνει. Παρακάτω παρουσιάζεται το παράδειγμα που εκτελέσαμε και στο εργαλείο Selenium για να παρατηρηθεί η απλότητα του εργαλείου αυτού σε σχέση με το προηγούμενο.

```
*** Settings ***
Library  SeleniumLibrary
*** Keywords ***
Begin Web Test
    open browser  https://www.unipi.gr/  gc
    maximize browser window
End Web Test
    close browser
```

Cucumber [41]

Αποτελεί ένα εργαλείο που είναι γραμμένο σε γλώσσα Ruby. Επιτρέπει στο χρήση να γράφει περιπτώσεις ελέγχου που εύκολα ο οποιασδήποτε άνθρωπος ανεξάρτητα των γνώσεων που έχει θα μπορεί να κατανοήσει. Ο τρόπος αυτός γραφής ακολουθεί το μοντέλο «Gherkin» που ουσιαστικά χρησιμοποιεί τις λέξεις κλειδιά «Given» «When» «Then». Ένα απλό παράδειγμα ελέγχου σεναρίου για να δείξουμε πόσο απλός είναι ο τρόπος γραφής είναι το παρακάτω:

Feature: Google Searching

Σαν μεταπτυχιακός φοιτητής θέλω να βρω πληροφορίες για τον «έλεγχο λογισμικού» μέσω του Google.

Scenario: Simple Google search

Given Βρίσκομαι σε ένα πρόγραμμα περιήγησης και έχω συνδεθεί στο google.com

When η φράση "έλεγχος λογισμικού" αναζητηθεί

Then τα αποτελέσματα για "έλεγχος λογισμικού" θα εμφανιστούν

4.3.3 Εργαλεία που ελέγχουν την απόδοση του συστήματος

Όπως και τα εργαλεία ελέγχου μονάδας, έτσι και τα εργαλεία απόδοσης του συστήματος δεν θα μας απασχολήσουν σε αυτή την εργασία. Καλό είναι όμως ο αναγνώστης να έχει μια σφαιρική εικόνα όλων των εργαλείων οπότε και σε αυτή την υπό-ενότητα θα αναφερθεί το πιο διαδεδομένο εργαλείο ελέγχου απόδοσης ενός συστήματος.

JMeter [42]

Το συγκεκριμένο εργαλείο έχει γραφτεί σε γλώσσα Java και αποτελεί ίσως το πιο διαδεδομένο εργαλείο ελέγχου και μέτρησης της απόδοσης ενός συστήματος. Με το εργαλείο αυτό μπορούν να προσομοιαστούν μεγάλου και διαφορετικού φορτίου ενέργειες σε κάποιον διακομιστή (server) και να αναλυθεί η απόδοση του συστήματος σύμφωνα με τα αποτελέσματα που θα προκύψουν. Είναι σημαντικό να αναφερθεί ότι το συγκεκριμένο εργαλείο εκτός από εφαρμογές διαδικτύου υποστηρίζει και άλλα πρωτόκολλα όπως HTTP, SOAP, LDAP, MongoDB, TCP κ.α.

4.4 Σύγκριση και αξιολόγηση εργαλείων αυτοματοποιημένου ελέγχου αποδοχής

Όπως ήδη αναφέρθηκε, τα εργαλεία που θα μας απασχολήσουν σε αυτή την εργασία είναι τα εργαλεία αυτοματοποιημένου ελέγχου αποδοχής. Τα 3 εργαλεία που αναλύθηκαν παραπάνω είναι το Selenium, το Robot Framework και το Cucumber. Καταρχάς και τα 3 εργαλεία είναι εργαλεία ανοιχτού κώδικα οπότε μπορεί να τα χρησιμοποιήσει όποιος επιθυμεί. Επιπρόσθετα και τα 3 εργαλεία είναι σχετικά απλά στη χρήση τους και μπορεί κάποιος χρήστης που κατέχει βασικές γνώσεις στον αυτοματοποιημένο έλεγχο λογισμικού να τα χρησιμοποιήσει για να αυτοματοποιήσει τα σενάρια ελέγχου που επιθυμεί. Ένα ακόμα κοινό στοιχείο των παραπάνω εργαλείων είναι ότι παρέχουν στο χρήστη τη δυνατότητα να χρησιμοποιήσει τη γλώσσα προγραμματισμού Python για τους ελέγχους του, που στα πλαίσια αυτής της εργασίας είναι η γλώσσα που επιθυμεί ο συγγραφέας να χρησιμοποιήσει. Μια διαφορά που έχουν τα παραπάνω εργαλεία είναι ότι το Selenium προσφέρει τη δυνατότητα «Record-Playback» μέσω του Selenium WebDriver που δίνει τη δυνατότητα στον χρήστη να καταγράφει τα βήματα που ακολουθεί σε ένα γραφικό περιβάλλον και αυτά να αναπαράγονται πάλι όποτε το επιθυμεί, ενώ τα άλλα 2 εργαλεία δεν προσφέρουν αυτή τη δυνατότητα . Επίσης άλλη μια διαφορά είναι ότι το Robot Framework προσφέρει εύκολα στο χρήστη υψηλής ποιότητας αναφορές (reports) σε σχέση με τα άλλα δύο εργαλεία. Τέλος, σχετικά με την απόδοση του κάθε εργαλείου, έχει παρατηρηθεί ότι όλα τα εργαλεία είναι σχετικά γρήγορα ωστόσο αρκετές φορές το Selenium είναι πιο αργό στην εκτέλεση των σεναρίων σε σχέση με τα άλλα δύο εργαλεία κυρίως λόγω των επιλογών που δίνει στο χρήστη που είναι σαφώς πιο πολλές.

Λαμβάνοντας υπόψη όλα τα παραπάνω και στα πλαίσια της παρούσας διπλωματικής εργασίας επιλέχθηκαν σαν πρώτες επιλογές το Selenium και το Robot Framework αφήνοντας σαν τρίτη επιλογή το Cucumber καθώς αν και απλό στη χρήση του δεν είναι τόσο διαδεδομένο. Στη συνέχεια και μαθαίνοντας τα 2 εργαλεία μέσω διαφόρων εφαρμογών και εκτελώντας κάποια απλά σενάρια ελέγχου σαν παραδείγματα, παρατηρήθηκε ότι το Robot Framework ήταν πιο απλό στη χρήση του με τη γλώσσα Python σε σχέση με το Selenium καθώς επίσης και οι αναφορές του ήταν ιδιαίτερα ελκυστικές στο μάτι του αναγνώστη και εύκολες στην εξαγωγή τους. Ειδικά το τελευταίο αποτέλεσε και τον βασικό παράγοντα επιλογής του Robot Framework αντί του Selenium στην εκτέλεση των σεναρίων ελέγχουν στην παρούσα διπλωματική εργασία όπως θα αναλυθεί και στο κεφάλαιο 5 καθώς οι αναφορές αποτελούν σημαντικό κομμάτι της εργασίας αυτής.

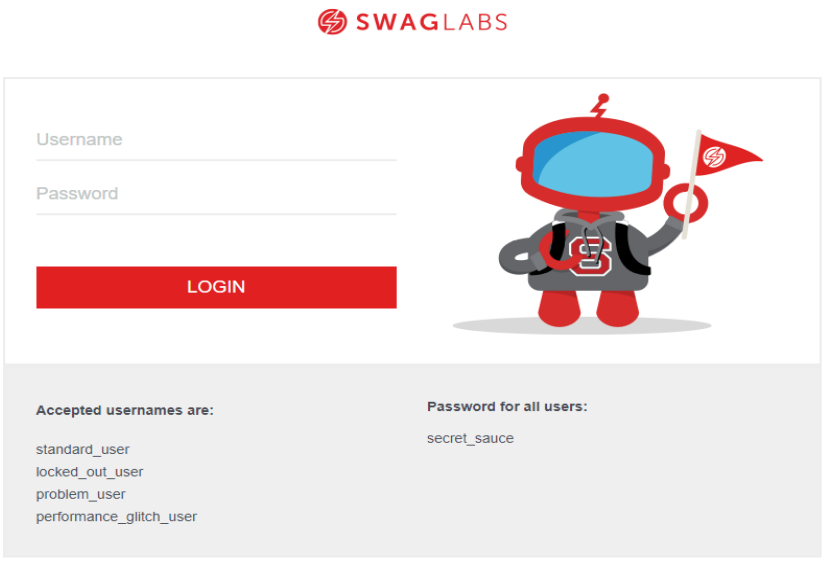
5. ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΣ ΈΛΕΓΧΟΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΗΝ ΕΦΑΡΜΟΓΗ «SauceDemo»

Στο προτελευταίο κεφάλαιο της παρούσας Διπλωματικής Εργασίας θα γίνει μια σύντομη περιγραφή της ιστοσελίδας που επιλέχθηκε να ελεγχθεί, στη συνέχεια θα δημιουργηθούν τα σενάρια ελέγχου και τέλος τα σενάρια αυτά θα αυτοματοποιηθούν και θα «εκτελεστούν». Τα αποτελέσματα που θα προκύψουν θα καταγραφούν και θα αναλυθούν.

5.1 Περιγραφή της εφαρμογής «SauceDemo»

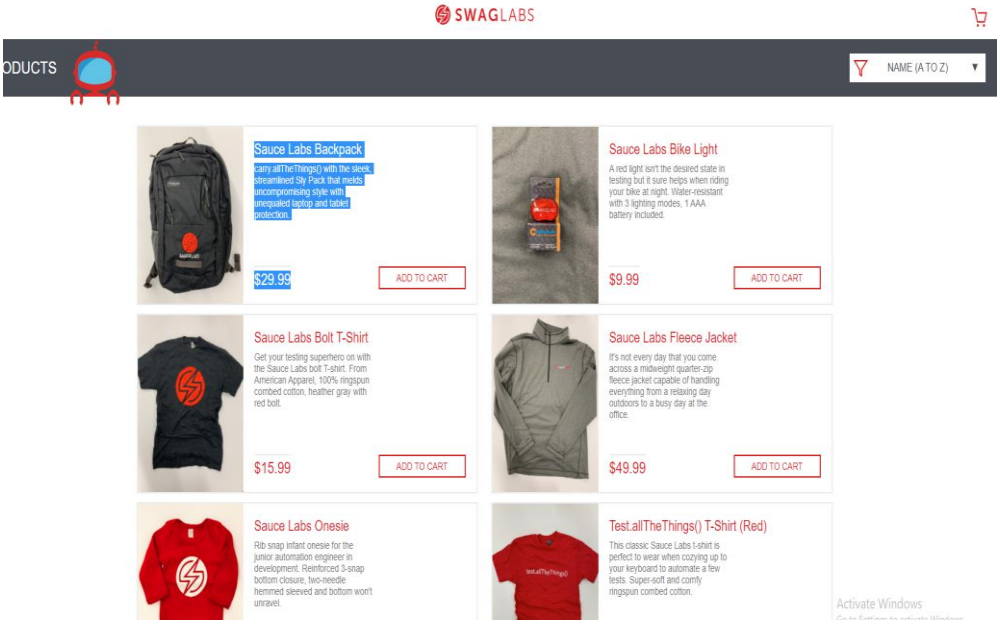
Η εφαρμογή SauceDemo (https://www.saucedemo.com/) αποτελεί μια εφαρμογή διαδικτύου ιδανική για αυτοματοποιημένους ελέγχους λογισμικού. Η εφαρμογή αυτή ανήκει στην εταιρία SauceLabs η οποία αποτελεί μια από τις πιο γνωστές εταιρίες παροχής εργαλείων αυτοματοποιημένου ελέγχου λογισμικού. Τα δικαιώματα αυτής της εφαρμογής ανήκουν αποκλειστικά στην εταιρία SauceLabs και η χρήση της στην συγκεκριμένη εργασία έγινε σε ερευνητικό πλαίσιο χωρίς να υπάρχει σκοπός εκμετάλλευσής της για κερδοσκοπικούς λόγους. Στη συνέχεια θα γίνει μια επεξήγηση της εφαρμογής και των βασικών τις λειτουργιών [43].

Αρχικά, μέσω του προαναφερθέντος URL ο χρήστης βρίσκεται σε μια φόρμα εισόδου που μπορεί να συνδεθεί με διαφορετικούς χρήστες και κάθε χρήστης έχει διαφορετική συμπεριφορά στο σύστημα. Έτσι υπάρχει ο χρήστης ***standard_user*** που προσφέρει όλες τις δυνατότητες του συστήματος χωρίς κανένα πρόβλημα, ο ***locked_out_user*** που αποτελεί έναν κλειδωμένο χρήστη που δεν έχει δικαίωμα να συνδεθεί στο σύστημα και ο ***problem_user*** που είναι ένας χρήστης που θα συνδεθεί στο σύστημα ωστόσο είναι «προβληματικός» και πολλές λειτουργίες του συστήματος δεν θα λειτουργούν. Ουσιαστικά αυτός ο χρήστης θα μας προσομοιώσει μια αληθινή κατάσταση που μπορεί σε μια έκδοση του συστήματος να λειτουργούν όλα άψογα ωστόσο στην αμέσως επόμενη πολλές λειτουργίες του συστήματος να έχουν «σπάσει» και να μην λειτουργούν σωστά. Εδώ είναι που ο αυτοματοποιημένος έλεγχος θα μας φανεί ιδιαίτερα χρήσιμος και σημαντικός και θα μας κερδίσει πολύτιμο χρόνο. Υπάρχει και ένας ακόμα χρήστης, ο ***performance_glitch_user***, που στα πλαίσια της παρούσας διπλωματικής εργασίας δεν θα μας απασχολήσει. Η αρχική σελίδα του συστήματος καθώς και όλοι οι πιθανοί χρήστες φαίνονται στην παρακάτω εικόνα.



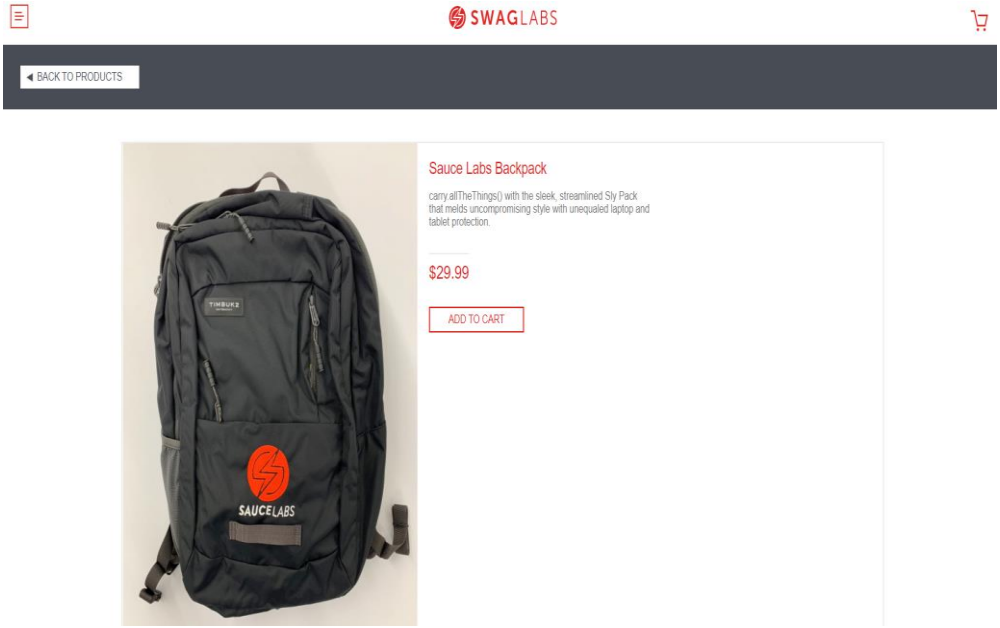
Εικόνα 15: Login Form [43]

Στην περίπτωση που συνδεθούμε με τον *standard_user* γίνεται ανακατεύθυνση σε μια σελίδα με προϊόντα όπου ο χρήστης μπορεί να δει τα προϊόντα, την περιγραφή τους, την τιμή τους και να επιλέξει πια από αυτά επιθυμεί να βάλει στο καλάθι του (Εικόνα 16). Υπάρχουν και άλλες δυνατότητες σε αυτή τη σελίδα, όπως το να ταξινομήσει τα προϊόντα ανάλογα το όνομά τους ή την τιμή τους, να μεταβεί στα μέσα κοινωνικής δικτύωσης της εταιρίας μέσω των αντίστοιχων κουμπιών και φυσικά να βγει από το σύστημα.



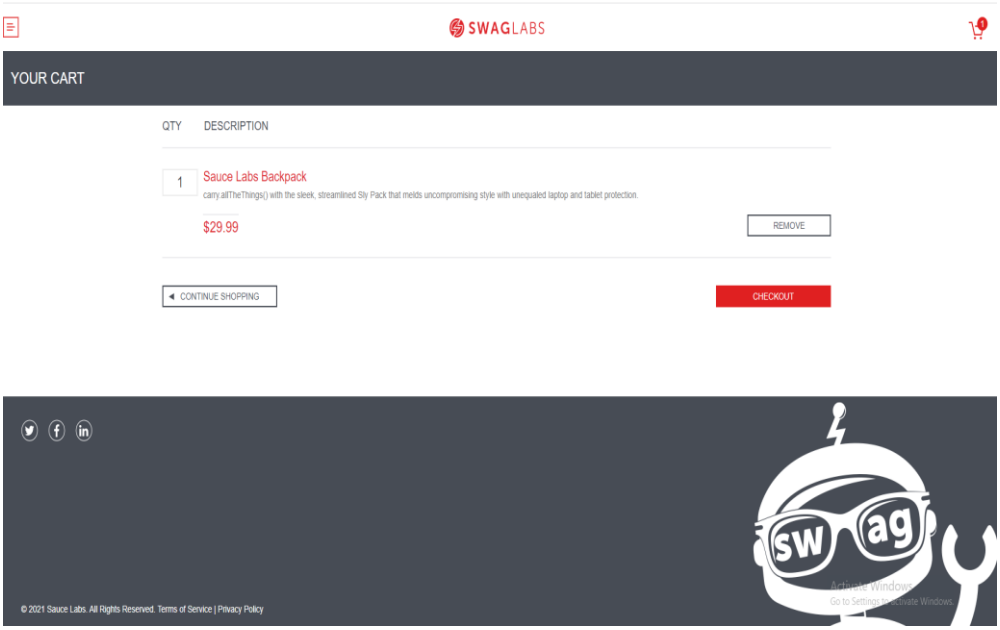
Εικόνα 16: Product page [43]

Ο χρήστης μπορεί επίσης να πατήσει πάνω στο προϊόν που επιθυμεί και μια νέα σελίδα θα ανοίξει που παρουσιάζει την τιμή του συγκεκριμένου προϊόντος και τα στοιχεία του (Εικόνα 17).



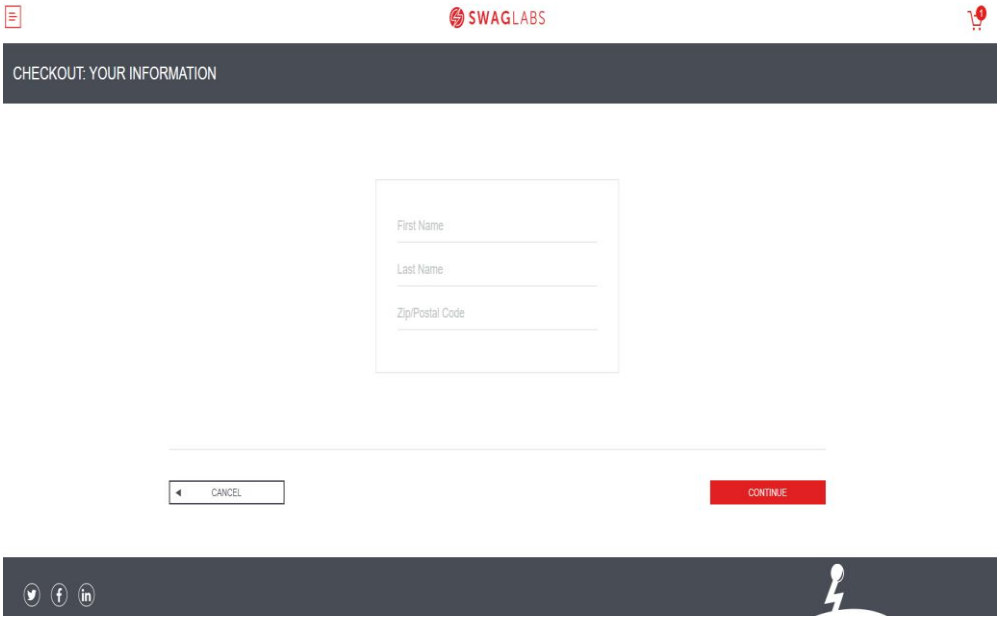
Εικόνα 17: Specific product page [43]

Στη συνέχεια και εφόσον ο χρήστης έχει προσθέσει προϊόντα στο καλάθι του μπορεί να επιλέξει το καλάθι (πάνω δεξιά) και θα μεταβεί σε μια νέα σελίδα που μπορεί να δει τα προϊόντα που έχει επιλέξει, την ποσότητα και την τιμή τους. Μπορεί να επιλέξει αν θέλει να τα βγάλει από το καλάθι ή να συνεχίσει την παραγγελία του (Εικόνα 18).



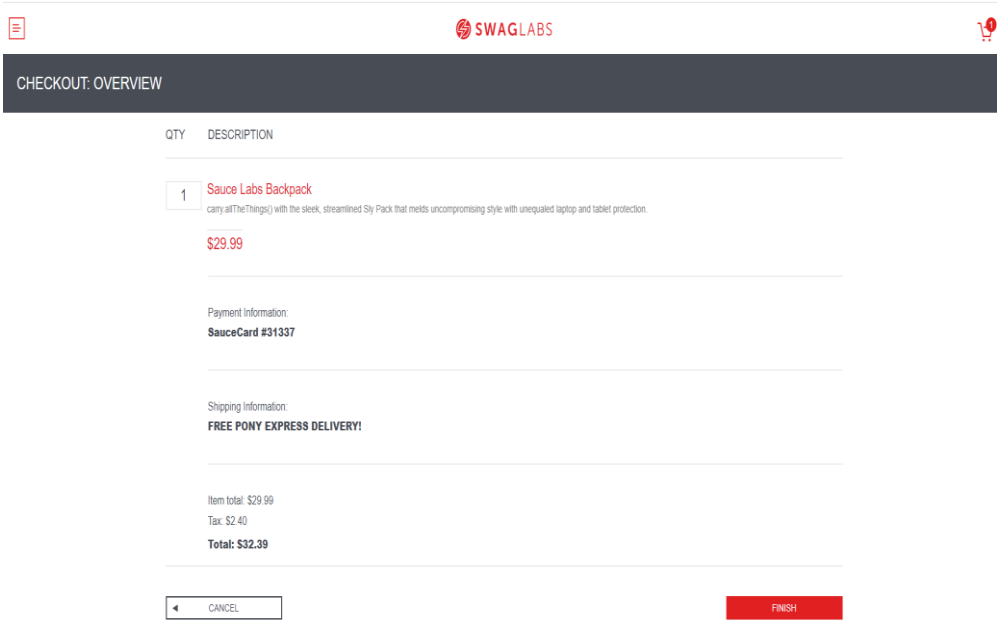
Εικόνα 18: Cart page [43]

Επιλέγοντας το κουμπί «CHECKOUT», ο χρήστης πρέπει να εισάγει τα στοιχεία του για να προχωρήσει στην τελική φάση της παραγγελίας του (Εικόνα 19).



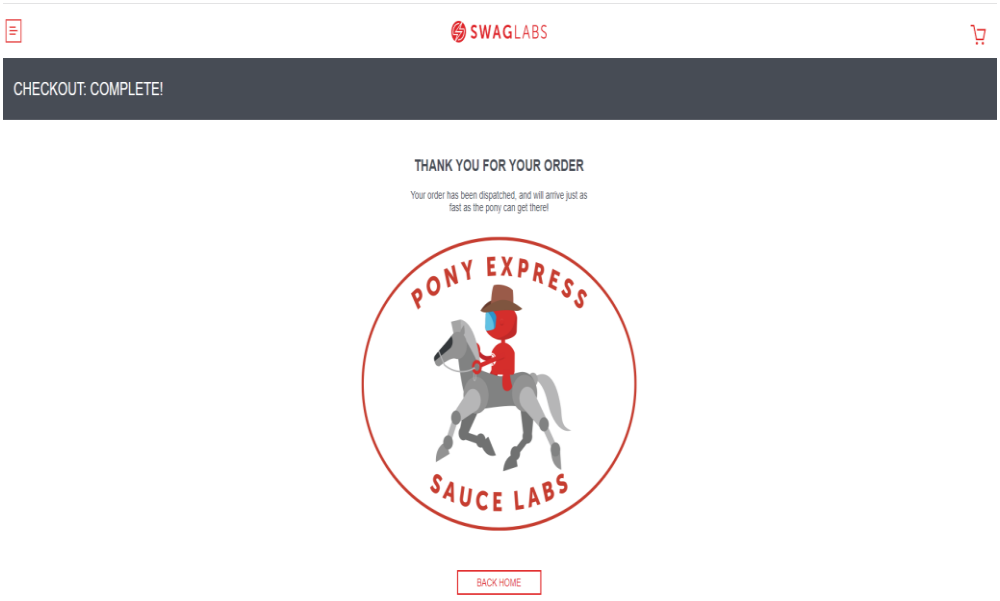
Εικόνα 19: Checkout-Your information page [43]

Με τη συμπλήρωση των στοιχείων και πατώντας το κουμπί «CONTINUE», ο χρήστης είναι έτοιμος να ολοκληρώσει την παραγγελία του. Στην τελευταία σελίδα ο χρήστης βλέπει όλες τις πληροφορίες της παραγγελίας του (προϊόν, πληροφορίες πληρωμής, κόστος, φορολογία κ.α.). Πατώντας «FINISH» μπορεί να ολοκληρώσει την παραγγελία του (Εικόνα 20).



Εικόνα 20: Checkout-Overview page [43]

Με την ολοκλήρωση της παραγγελίας του, ο χρήστης ενημερώνεται από το σύστημα με ένα ευχαριστήριο μήνυμα για την επιτυχή παραγγελία και μπορεί εφόσον το επιθυμεί να επιστρέψει στην αρχική σελίδα της εφαρμογής (Εικόνα 21).



Εικόνα 21: Checkout-Complete page [43]

Αν είχαμε συνδεθεί με τον «προβληματικό» χρήστη, τότε πολλές από τις παραπάνω ενέργειες δεν θα ήταν εφικτές και θα υπήρχαν πολλά προβλήματα (bugs) στο σύστημα. Τα προβλήματα αυτά θα τα δούμε αναλυτικότερα μέσω του αυτοματοποιημένου ελέγχου λογισμικού.

5.2 Επιλογή μεθόδου ελέγχου και δημιουργία σεναρίων ελέγχου

Για τον έλεγχο της εφαρμογής πρέπει πρώτα να αποφασίσουμε τη μέθοδο που θα χρησιμοποιήσουμε. Από τη στιγμή που δεν γνωρίζουμε τίποτα για τον κώδικα της εφαρμογής και οι μόνες μας γνώσεις είναι οι απαιτήσεις του συστήματος, τότε η πιο κατάλληλη μέθοδος σύμφωνα και με όσα αναλύσαμε στο κεφάλαιο 3 είναι η μέθοδος του μαύρου κουτιού.

Για να χρησιμοποιήσουμε την παραπάνω μέθοδο ως tester οφείλουμε να καταλάβουμε την εφαρμογή, τις απαιτήσεις της και τι ακριβώς πρέπει να ελέγξουμε. Στη συνέχεια πρέπει να δημιουργήσουμε τα σενάρια ελέγχου (test cases) που θα «τρέξουν» στο σύστημα. Τα σενάρια αυτά πρέπει να καλύπτουν όλο το εύρος της εφαρμογής. Επίσης τα σενάρια αυτά πρέπει να καλύπτουν και θετικά αλλά και αρνητικά αποτελέσματα του συστήματος. Για παράδειγμα, όταν ελέγχουμε την είσοδο σε ένα σύστημα δεν αρκεί να ελέγξουμε ότι χρησιμοποιώντας τα σωστά στοιχεία θα συνδεθούμε επιτυχώς στο σύστημα αλλά πρέπει να ελέγξουμε και το σενάριο που δεν χρησιμοποιούμε σωστά στοιχεία και τότε το σύστημα δεν πρέπει να μας αφήνει να συνδεόμαστε σε αυτό. Με την ολοκλήρωση της δημιουργίας των σεναρίων ελέγχου, ο tester μπορεί να ξεκινήσει τον έλεγχο της εφαρμογής αλληλοεπιδρώντας με αυτήν σαν τελικός χρήστης αν είναι manual tester ή με τη χρήση κατάλληλων εργαλείων αν είναι automation tester [44].

Δημιουργία test cases

Για τη δημιουργία κάθε test case δημιουργήσαμε έναν δισδιάστατο πίνακα με τα παρακάτω πεδία:

Αναγνωριστικό Δοκιμής (Test ID): Ένα αναγνωριστικό για κάθε test case που δημιουργούμε. Η αρίθμηση είναι κατά αύξουσα σειρά, οπότε το πρώτο σενάριο θα έχει την τιμή TC01, το δεύτερο την τιμή TC02 και έτσι θα συνεχιστεί μέχρι να ολοκληρώσουμε όλα τα σενάρια ελέγχου. Αυτό το αναγνωριστικό είναι υποχρεωτικό και θα μας βοηθήσει όταν θα «τρέξουμε» τους ελέγχους και θέλουμε να δώσουμε αποτελέσματα να μπορούμε να δηλώσουμε ποιο σενάριο ελέγχου πέρασε επιτυχώς τον έλεγχο και ποιο απέτυχε.

Τίτλος Δοκιμής (Test Title): Εδώ για κάθε σενάριο ελέγχου θα δηλώνουμε μια σύντομη περιγραφή του τι θα ελέγξουμε. Και αυτό το πεδίο είναι υποχρεωτικό.

Ετικέτες (Labels): Μη υποχρεωτικό πεδίο. Μέσω των ετικετών θα δηλώνουμε όποτε το επιθυμούμε ένα χαρακτηριστικό που μπορεί να έχει το σενάριο ελέγχου που φτιάχνουμε. Για παράδειγμα, κάποια σενάρια ελέγχου μπορεί να είναι πολύ σημαντικά οπότε να θέλουμε να τους βάλουμε μια ετικέτα που να δηλώνει αυτή τη σημαντικότητα (π.χ. Critical_Scenario). Έτσι αν έχουμε πολλά τέτοια σενάρια μπορούμε εύκολα να τα ομαδοποιήσουμε με αυτή την ετικέτα και να τα εκτελέσουμε όλα μαζί πρώτα, και στη συνέχεια να εκτελέσουμε δευτερεύοντα σενάρια. Στα πλαίσια αυτής της εργασίας χρησιμοποιήσαμε 3 ετικέτες. Αυτές είναι:

- **Smoke_Test:** Η ετικέτα αυτή ουσιαστικά θα μπει στα πολύ βασικά σενάρια ελέγχου που θα εξετάζουν τις βασικές λειτουργίες του συστήματος. Αν κάποιο

από αυτά τα σενάρια αποτύχει, τότε το σύστημα σίγουρα δεν μπορεί να θεωρηθεί λειτουργικό.

- **Regression:** Η ετικέτα αυτή θα χρησιμοποιηθεί στα περισσότερα σενάρια ελέγχου. Κάθε έλεγχος που θα έχει αυτή την ετικέτα θα εκτελείται σε κάθε έκδοση του συστήματος και αποτελεί τον κύριο όγκο των ελέγχων που πρέπει να πραγματοποιηθούν για να ελεγχθεί η εφαρμογή
- **Low_Priority:** Η ετικέτα αυτή θα χρησιμοποιηθεί σε σενάρια ελέγχου μικρής σημασίας που η προτεραιότητά τους είναι μικρή. Τα σενάρια αυτά μπορούν να εκτελεστούν τελευταία ή και να μην εκτελεστούν καθόλου καθώς έχουν μικρή επίπτωση στη λειτουργία του συστήματος.

Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps & Expected Results): Εδώ θα γίνει η αναλυτική περιγραφή του σεναρίου ελέγχου που θα τρέξουμε βήμα βήμα και ποια είναι τα επιθυμητά αποτελέσματα. Αυτό θα επιτευχθεί αναλύοντας τι «κινήσεις» πρέπει να κάνει ο tester για να ελέγξει κάθε βήμα και ποιο είναι το επιθυμητό αποτέλεσμα κάθε κίνησης. Είναι εύκολα κατανοητό πως και αυτό το πεδίο είναι υποχρεωτικό.

Δεδομένα Δοκιμής (Test Data): Το συγκεκριμένο πεδίο είναι προαιρετικό και θα το χρησιμοποιούμε όταν θέλουμε να βάλουμε συγκεκριμένα δεδομένα σε ένα σενάριο ελέγχου.

Παρακάτω παρουσιάζονται όλα τα σενάρια ελέγχου που δημιουργήσαμε.

Πίνακας 1: TC01-Είσοδος στο σύστημα με έγκυρο χρήστη

Αναγνωριστικό Δοκιμής (Test ID)	TC01
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα με έγκυρο χρήστη
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 2: TC02-Είσοδος στο σύστημα με μη έγκυρο username

Αναγνωριστικό Δοκιμής (Test ID)	TC02
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα με μη έγκυρο username χρήστη
Ετικέτες (Labels)	Regression

Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί μη έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης δεν συνδέεται στο λογαριασμό του και ένα μήνυμα λάθους εμφανίζεται στην οθόνη *(Epic sadface: Username and password do not match any user in this service*).
Δεδομένα Δοκιμής (Test Data)	Username: abc123 Password: secret_sauce

Πίνακας 3: TC03-Είσοδος στο σύστημα με μη έγκυρο password

Αναγνωριστικό Δοκιμής (Test ID)	TC03
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα με μη έγκυρο password χρήστη
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί μη έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης δεν συνδέεται στο λογαριασμό του και ένα μήνυμα λάθους εμφανίζεται στην οθόνη (*Epic sadface: Username and password do not match any user in this service*).
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: abc123

Πίνακας 4: TC04-Είσοδος στο σύστημα με κλειδωμένο χρήστη

Αναγνωριστικό Δοκιμής (Test ID)	TC04
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα με κλειδωμένο χρήστη
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί στοιχεία από έναν κλειδωμένο χρήστη και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του.

	Αποτέλεσμα: Ο χρήστης δεν συνδέεται στο λογαριασμό του επειδή είναι κλειδωμένος και ένα μήνυμα λάθους εμφανίζεται στην οθόνη(*Epic sadface: Sorry, this user has been locked out*.).
Δεδομένα Δοκιμής (Test Data)	Username: locked_out_user Password: secret_sauce

Πίνακας 5: TC05-Είσοδος στο σύστημα χωρίς τη χρήση credentials

Αναγνωριστικό Δοκιμής (Test ID)	TC05
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα χωρίς τη χρήση username και password
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης χωρίς να πληκτρολογήσει username και password πατάει το κουμπί «LOGIN» *Αποτέλεσμα:* Ο χρήστης δεν συνδέεται σε κάποιο λογαριασμό και ένα μήνυμα λάθους εμφανίζεται στην οθόνη (*Epic sadface: Username is required*).
Δεδομένα Δοκιμής (Test Data)	Username: - (Empty) Password: - (Empty)

Πίνακας 6: TC06-Είσοδος στο σύστημα χωρίς τη χρήση password

Αναγνωριστικό Δοκιμής (Test ID)	TC06
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα χωρίς τη χρήση password
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί username και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης δεν συνδέεται στο λογαριασμό του επειδή δεν έχει εισάγει κωδικό και ένα μήνυμα λάθους εμφανίζεται στην οθόνη (*Epic sadface: Password is required*).
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: - (Empty)

Πίνακας 7: TC07-Είσοδος στο σύστημα χωρίς τη χρήση username

Αναγνωριστικό Δοκιμής (Test ID)	TC07
Τίτλος Δοκιμής (Test Title)	Είσοδος στο σύστημα χωρίς τη χρήση username
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί password και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης δεν συνδέεται στο λογαριασμό του επειδή δεν έχει εισάγει username και ένα μήνυμα λάθους εμφανίζεται στην οθόνη (*Epic sadface: Username is required*).
Δεδομένα Δοκιμής (Test Data)	Username: - (Empty) Password: secret_sauce

Πίνακας 8: TC08-Ανακατεύθυνση στα Social Media της εφαρμογής

Αναγνωριστικό Δοκιμής (Test ID)	TC08
Τίτλος Δοκιμής (Test Title)	Ανακατεύθυνση στα Social Media της εφαρμογής
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης πατάει στην εικόνα με το σήμα του «Twitter». *Αποτέλεσμα:* Μια νέα καρτέλα περιηγητή ανοίγει και ο χρήστης μπορεί να δει το Twitter της εφαρμογής. 4. Ο χρήστης πατάει στην εικόνα με το σήμα του «Facebook». *Αποτέλεσμα:* Μια νέα καρτέλα περιηγητή ανοίγει και ο χρήστης μπορεί να δει το Facebook της εφαρμογής. 3. Ο χρήστης πατάει στην εικόνα με το σήμα του «LinkedIn».

	Αποτέλεσμα: Μια νέα καρτέλα περιηγητή ανοίγει και ο χρήστης μπορεί να δει το LinkedIn της εφαρμογής.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 9: TC09-Έξοδος από την εφαρμογή μέσω του πλαϊνού μενού

Αναγνωριστικό Δοκιμής (Test ID)	TC09
Τίτλος Δοκιμής (Test Title)	Έξοδος από την εφαρμογή μέσω του πλαϊνού μενού
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης πατάει στο κουμπί πάνω αριστερά με τις τρεις γραμμές και επιλέγει να κάνει «LOGOUT» από το σύστημα. *Αποτέλεσμα:* Ο χρήστης βγήκε από την εφαρμογή και η αρχική σελίδα που ζητάει στοιχεία για να συνδεθεί κάποιος χρήστης εμφανίζεται.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 10: TC10-Χρήση του κουμπιού «About»

Αναγνωριστικό Δοκιμής (Test ID)	TC10
Τίτλος Δοκιμής (Test Title)	Χρήση του κουμπιού «ABOUT»
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας.

	3. Ο χρήστης πατάει στο κουμπί πάνω αριστερά με τις τρεις γραμμές και επιλέγει να πατήσει την επιλογή «ABOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνθηκε στη σελίδα «https://saucelabs.com/».
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 11: TC11-Χρήση της επιλογής ταξινόμησης

Αναγνωριστικό Δοκιμής (Test ID)	TC11
Τίτλος Δοκιμής (Test Title)	Χρήση της επιλογής ταξινόμησης
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης πατάει στο κουμπί πάνω δεξιά που αφορά την ταξινόμηση και επιλέγει να κάνει ταξινόμηση των προϊόντων κατά αύξουσα σειρά σύμφωνα με το όνομα. *Αποτέλεσμα:* Τα προϊόντα παρουσιάζονται κατά αύξουσα σειρά σύμφωνα με το όνομα του κάθε προϊόντος. 4. Ο χρήστης πατάει στο κουμπί πάνω δεξιά που αφορά την ταξινόμηση και επιλέγει να κάνει ταξινόμηση των προϊόντων κατά φθίνουσα σειρά σύμφωνα με το όνομα. *Αποτέλεσμα:* Τα προϊόντα παρουσιάζονται κατά φθίνουσα σειρά σύμφωνα με το όνομα του κάθε προϊόντος. 5. Ο χρήστης πατάει στο κουμπί πάνω δεξιά που αφορά την ταξινόμηση και επιλέγει να κάνει ταξινόμηση των προϊόντων κατά αύξουσα σειρά σύμφωνα με την τιμή τους.

	Αποτέλεσμα: Τα προϊόντα παρουσιάζονται κατά αύξουσα σειρά σύμφωνα με την τιμή τους. 6. Ο χρήστης πατάει στο κουμπί πάνω δεξιά που αφορά την ταξινόμηση και επιλέγει να κάνει ταξινόμηση των προϊόντων κατά φθίνουσα σειρά σύμφωνα με την τιμή τους *Αποτέλεσμα:* Τα προϊόντα παρουσιάζονται κατά φθίνουσα σειρά σύμφωνα με την τιμή τους.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 12: TC12-Έλεγχος όλων των διαθέσιμων προϊόντων

Αναγνωριστικό Δοκιμής (Test ID)	TC12
Τίτλος Δοκιμής (Test Title)	Έλεγχος όλων των διαθέσιμων προϊόντων
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης ελέγχει ότι τα διαθέσιμα προϊόντα σύμφωνα με τις προδιαγραφές του συστήματος πρέπει να είναι 6. *Αποτέλεσμα:* Τα διαθέσιμα προϊόντα είναι 6.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 13: TC13-Προσθήκη προϊόντος στο καλάθι του χρήστη

Αναγνωριστικό Δοκιμής (Test ID)	TC13
Τίτλος Δοκιμής (Test Title)	Προσθήκη προϊόντος στο καλάθι του χρήστη
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς.

	2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 14: TC14-Αφαίρεση προϊόντος από το καλάθι ενός χρήστη

Αναγνωριστικό Δοκιμής (Test ID)	TC14
Τίτλος Δοκιμής (Test Title)	Αφαίρεση προϊόντος από το καλάθι ενός χρήστη
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης επιλέγει το προϊόν που έχει προσθέσει και πατάει το κουμπί «REMOVE». *Αποτέλεσμα:* Το προϊόν αφαιρείται από το καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι ο αριθμός των προϊόντων έχει μειωθεί κατά ένα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 15: TC15-Εμφάνιση σελίδας καλαθιού χρήστη

Αναγνωριστικό Δοκιμής (Test ID)	TC15
Τίτλος Δοκιμής (Test Title)	Εμφάνιση σελίδας καλαθιού χρήστη
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 16: TC16-Αφαίρεση προϊόντος από το καλάθι ενός χρήστη(cart page)

Αναγνωριστικό Δοκιμής (Test ID)	TC16
Τίτλος Δοκιμής (Test Title)	Αφαίρεση προϊόντος από το καλάθι ενός χρήστη ενώ αυτός βρίσκεται στη σελίδα καλαθιού (cart page)
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας.

	3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στην σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης επιλέγει το προϊόν που έχει προσθέσει και πατάει το κουμπί «REMOVE» *Αποτέλεσμα:* Το προϊόν αφαιρείται από το καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι ο αριθμός των προϊόντων έχει μειωθεί κατά ένα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 17: TC17-Επιστροφή στην σελίδα με τα προϊόντα ενώ ο χρήστης βρίσκεται στο cart page

Αναγνωριστικό Δοκιμής (Test ID)	TC17
Τίτλος Δοκιμής (Test Title)	Επιστροφή στη σελίδα με όλα τα διαθέσιμα προϊόντα ενώ ο χρήστης βρίσκεται στη σελίδα καλαθιού (cart page)
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 4. Ο χρήστης πατάει το κουμπί «CONTINUE SHOPPING».

	Αποτέλεσμα: Ο χρήστης επιστρέφει στη σελίδα με όλα τα διαθέσιμα προϊόντα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 18: TC18-Έλεγχος πνευματικών δικαιωμάτων

Αναγνωριστικό Δοκιμής (Test ID)	TC18
Τίτλος Δοκιμής (Test Title)	Έλεγχος πνευματικών δικαιωμάτων στην κεντρική σελίδα της εφαρμογής
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης ελέγχει αν υπάρχει το κείμενο με τα απαραίτητα πνευματικά δικαιώματα στο υποσέλιδο της σελίδας. *Αποτέλεσμα:* Το απαραίτητο κείμενο «© 2021 Sauce Labs. All Rights Reserved. Terms of Service | Privacy Policy» είναι διαθέσιμο.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 19: TC19-Έλεγχος ανακατεύθυνσης στη σελίδα «CHECKOUT: YOUR INFORMATION»

Αναγνωριστικό Δοκιμής (Test ID)	TC19
Τίτλος Δοκιμής (Test Title)	Έλεγχος ανακατεύθυνσης στη σελίδα «CHECKOUT: YOUR INFORMATION»
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του.

	Αποτέλεσμα: Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στην σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION».
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 20: TC20-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Έγκυρα στοιχεία παραγγελίας

Αναγνωριστικό Δοκιμής (Test ID)	TC20
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Έγκυρα στοιχεία παραγγελίας
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά).

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στην σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT» *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στην σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name», «Last Name», «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: OVERVIEW».
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 21: TC21-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας κενά

Αναγνωριστικό Δοκιμής (Test ID)	TC21
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας κενά
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά).

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης πατάει στο κουμπί «CONTINUE» χωρίς να εισάγει κανένα στοιχείο. *Αποτέλεσμα:* Ένα μήνυμα λάθους εμφανίζεται στο χρήστη «Error: First Name is required» και η παραγγελία δεν συνεχίζεται.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 22: TC22-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς όνομα

Αναγνωριστικό Δοκιμής (Test ID)	TC22
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς όνομα
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ . *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά) *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT».

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «Last Name» «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ένα μήνυμα λάθους εμφανίζεται στο χρήστη «Error: First Name is required» και η παραγγελία δεν συνεχίζεται.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 23: TC23-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς επίθετο

Αναγνωριστικό Δοκιμής (Test ID)	TC23
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς επίθετο
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT».

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name» «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ένα μήνυμα λάθους εμφανίζεται στο χρήστη «Error: Last Name is required» και η παραγγελία δεν συνεχίζεται.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 24: TC24-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς ταχυδρομικό κώδικα

Αναγνωριστικό Δοκιμής (Test ID)	TC24
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Στοιχεία παραγγελίας χωρίς ταχυδρομικό κώδικα
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT»

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name» «Last Name». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ένα μήνυμα λάθους εμφανίζεται στο χρήστη «Error: Postal Code is required» και η παραγγελία δεν συνεχίζεται.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 25: TC25-Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Κουμπί επιστροφής στην προηγούμενη σελίδα

Αναγνωριστικό Δοκιμής (Test ID)	TC25
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: YOUR INFORMATION» - Κουμπί επιστροφής στην προηγούμενη σελίδα.
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά) *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT».

	Αποτέλεσμα: Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης πατάει το κουμπί «CANCEL». *Αποτέλεσμα:* Ο χρήστης επιστρέφει στην προηγούμενη σελίδα δηλαδή την σελίδα καλαθιού (Cart page).
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 26: TC26-Έλεγχος της σελίδας «CHECKOUT: OVERVIEW» . Σωστά διαθέσιμη όλη η πληροφορία

Αναγνωριστικό Δοκιμής (Test ID)	TC26
Τίτλος Δοκιμής (Test Title)	Έλεγχος της σελίδας «CHECKOUT: OVERVIEW» . Σωστά διαθέσιμη όλη η πληροφορία
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name», «Last Name», «Zip/Postal Code».

	Αποτέλεσμα: Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: OVERVIEW» 8. Ο χρήστης ελέγχει ότι όλα τα προϊόντα που έχει επιλέξει έχουν σωστά δεδομένα. *Αποτέλεσμα:* Όλα τα προϊόντα εμφανίζονται με σωστή περιγραφή, σωστή τιμή και σωστή ποσότητα. 9. Ο χρήστης ελέγχει τις πληροφορίες πληρωμής. *Αποτέλεσμα:* Οι πληροφορίες πληρωμής σύμφωνα με τις προδιαγραφές του συστήματος είναι πάντα οι «SauceCard #31337». 10. Ο χρήστης ελέγχει τις πληροφορίες αποστολής. *Αποτέλεσμα:* Οι πληροφορίες αποστολής σύμφωνα με τις προδιαγραφές του συστήματος είναι πάντα οι «FREE PONY EXPRESS DELIVERY!».
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 27: TC27-Σελίδα «CHECKOUT: OVERVIEW». Ολοκλήρωση παραγγελίας

Αναγνωριστικό Δοκιμής (Test ID)	TC27
Τίτλος Δοκιμής (Test Title)	Σελίδα «CHECKOUT: OVERVIEW». Ολοκλήρωση παραγγελίας
Ετικέτες (Labels)	Smoke_Test, Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει.

	4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name», «Last Name» «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: OVERVIEW». 8. Ο χρήστης ελέγχει όλα τα προϊόντα που έχει επιλέξει. *Αποτέλεσμα:* Όλα τα προϊόντα εμφανίζονται με σωστή περιγραφή, σωστή τιμή και σωστή ποσότητα. 9.Ο χρήστης πατάει στο κουμπί «FINISH». *Αποτέλεσμα:* Η παραγγελία ολοκληρώθηκε επιτυχώς και ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: COMPLETE!». 10. Ο χρήστης ελέγχει τα μηνύματα που εμφανίζονται στη σελίδα ολοκλήρωσης παραγγελίας. *Αποτέλεσμα:* Σύμφωνα με τις προδιαγραφές του συστήματος δύο μηνύματα πρέπει να εμφανίζονται a. THANK YOU FOR YOUR ORDER b. Your order has been dispatched, and will arrive just as fast as the pony can get there!
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 28: TC28-Σελίδα «CHECKOUT: OVERVIEW» . Ακύρωση παραγγελίας

Αναγνωριστικό Δοκιμής (Test ID)	TC28
Τίτλος Δοκιμής (Test Title)	Σελίδα «CHECKOUT: OVERVIEW» . Ακύρωση παραγγελίας
Ετικέτες (Labels)	Regression

Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART». *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name», «Last Name» «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: OVERVIEW». 8. Ο χρήστης ελέγχει όλα τα προϊόντα που έχει επιλέξει. *Αποτέλεσμα:* Όλα τα προϊόντα εμφανίζονται με σωστή περιγραφή, σωστή τιμή και σωστή ποσότητα. 9.Ο χρήστης πατάει στο κουμπί «CANCEL». *Αποτέλεσμα:* Η παραγγελία ακυρώθηκε και ο χρήστης ανακατευθύνεται στην αρχική σελίδα της εφαρμογής με όλα τα διαθέσιμα προϊόντα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user

	Password: secret_sauce

Πίνακας 29: TC29-Σελίδα «CHECKOUT: COMPLETE!». Έλεγχος κουμπιού επιστροφής στην αρχική σελίδα

Αναγνωριστικό Δοκιμής (Test ID)	TC29
Τίτλος Δοκιμής (Test Title)	Σελίδα «CHECKOUT: COMPLETE!» . Έλεγχος κουμπιού επιστροφής στην αρχική σελίδα
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART»/ *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης πατάει στο καλάθι με τα προϊόντα (κουμπί πάνω δεξιά). *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα με όλα τα προϊόντα που έχει βάλει στο καλάθι του. 5. Ο χρήστης πατάει το κουμπί «CHECKOUT». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: YOUR INFORMATION». 6. Ο χρήστης εισάγει τα στοιχεία του στα πεδία «First Name», «Last Name» «Zip/Postal Code». *Αποτέλεσμα:* Τα στοιχεία καταγράφηκαν και μπορεί ο χρήστης να τα δει. 7. Ο χρήστης πατάει στο κουμπί «CONTINUE». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: OVERVIEW». 8. Ο χρήστης ελέγχει όλα τα προϊόντα που έχει επιλέξει

	Αποτέλεσμα: Όλα τα προϊόντα εμφανίζονται με σωστή περιγραφή, σωστή τιμή και σωστή ποσότητα. 9.Ο χρήστης πατάει στο κουμπί «FINISH». *Αποτέλεσμα:* Η παραγγελία ολοκληρώθηκε επιτυχώς και ο χρήστης ανακατευθύνεται στη σελίδα «CHECKOUT: COMPLETE!». 10. Ο χρήστης πατάει στο κουμπί «BACK HOME». *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στην αρχική σελίδα της εφαρμογής που περιλαμβάνει όλα τα διαθέσιμα προϊόντα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 30: TC30-Έλεγχος εικόνων προϊόντων

Αναγνωριστικό Δοκιμής (Test ID)	TC30
Τίτλος Δοκιμής (Test Title)	Έλεγχος εικόνων προϊόντων
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης ελέγχει τις εικόνες όλων των προϊόντων. *Αποτέλεσμα:* Όλες οι εικόνες πρέπει να είναι αυτές που έχουν συμφωνηθεί από τις προδιαγραφές της εφαρμογής.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 31: TC31-Προσθήκη και αφαίρεση όλων των προϊόντων

Αναγνωριστικό Δοκιμής (Test ID)	TC31
Τίτλος Δοκιμής (Test Title)	Προσθήκη και αφαίρεση όλων των προϊόντων
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/ .

	Αποτέλεσμα: Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει το κουμπί «ADD TO CART» *Αποτέλεσμα:* Το προϊόν προστίθεται στο καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι βλέπει τον αριθμό των προϊόντων που έχει προσθέσει. 4. Ο χρήστης επιλέγει το προϊόν που έχει προσθέσει και πατάει το κουμπί «REMOVE». *Αποτέλεσμα:* Το προϊόν αφαιρείται από το καλάθι του χρήστη και πάνω δεξιά στο σύμβολο με το καλάθι ο αριθμός των προϊόντων έχει μειωθεί κατά ένα. *Να επαναληφθεί για όλα τα προϊόντα στο ίδιο σενάριο ελέγχου.*
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 32: TC32-Άνοιγμα συγκεκριμένου προϊόντος και επιστροφή στην αρχική σελίδα

Αναγνωριστικό Δοκιμής (Test ID)	TC32
Τίτλος Δοκιμής (Test Title)	Άνοιγμα συγκεκριμένου προϊόντος και επιστροφή στην αρχική σελίδα
Ετικέτες (Labels)	Low_Priority
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης επιλέγει ένα προϊόν και πατάει στο τίτλο του. *Αποτέλεσμα:* Το συγκεκριμένο προϊόν εμφανίζεται σε μια άλλη σελίδα μόνο του με τα στοιχεία του.

	4. Ο χρήστης πατάει το κουμπί επιστροφής στην αρχική σελίδα. *Αποτέλεσμα:* Ο χρήστης ανακατευθύνεται στην αρχική σελίδα.
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 33: TC33-Έλεγχος τιμής προϊόντων μεταξύ σελίδας όλων των προϊόντων και της σελίδας συγκεκριμένου προϊόντος

Αναγνωριστικό Δοκιμής (Test ID)	TC33
Τίτλος Δοκιμής (Test Title)	Έλεγχος τιμής προϊόντων μεταξύ σελίδας όλων των προϊόντων και της σελίδας συγκεκριμένου προϊόντος
Ετικέτες (Labels)	Regression
Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης ελέγχει την τιμή του προϊόντος στην αρχική σελίδα σε σύγκριση με την τιμή του προϊόντος στη σελίδα του κάθε συγκεκριμένου προϊόντος. *Αποτέλεσμα:* Το συγκεκριμένο προϊόν έχει την ίδια τιμή και στις δύο σελίδες. *Να επαναληφθεί για όλα τα προϊόντα.*
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

Πίνακας 34: TC34-Έλεγχος κειμένου προϊόντων μεταξύ σελίδας όλων των προϊόντων και της σελίδας συγκεκριμένου προϊόντος

Αναγνωριστικό Δοκιμής (Test ID)	TC34
Τίτλος Δοκιμής (Test Title)	Έλεγχος κειμένου προϊόντων μεταξύ σελίδας όλων των προϊόντων και της σελίδας συγκεκριμένου προϊόντος
Ετικέτες (Labels)	Low_Priority

Περιγραφή Δοκιμής και Επιθυμητά Αποτελέσματα (Test Steps – Expected Results)	1. Ο χρήστης κάνει πλοήγηση στην ιστοσελίδα https://www.saucedemo.com/. *Αποτέλεσμα:* Η ιστοσελίδα ανοίγει επιτυχώς. 2. Ο χρήστης πληκτρολογεί έγκυρα στοιχεία και πατάει το κουμπί «LOGIN» για να συνδεθεί στο λογαριασμό του. *Αποτέλεσμα:* Ο χρήστης συνδέθηκε επιτυχώς στο λογαριασμό του και βλέπει τα διαθέσιμα προϊόντα της ιστοσελίδας. 3. Ο χρήστης ελέγχει το κείμενο του προϊόντος στην αρχική σελίδα σε σύγκριση με το κείμενο του προϊόντος στη σελίδα του κάθε συγκεκριμένου προϊόντος. *Αποτέλεσμα:* Το συγκεκριμένο προϊόν έχει το ίδιο κείμενο και στις δύο σελίδες. *Να επαναληφθεί για όλα τα προϊόντα.*
Δεδομένα Δοκιμής (Test Data)	Username: standard_user Password: secret_sauce

5.3 Αυτοματοποιημένος έλεγχος της εφαρμογής

Όπως είδαμε και στο προηγούμενο υπό-κεφάλαιο, τα σενάρια ελέγχου που δημιουργήσαμε μπορούν να εκτελεστούν «χειρωνακτικά» από έναν χρήστη ωστόσο αυτό απαιτεί χρόνο. Ο χρόνος αυτός μεγαλώνει και άλλο αν σκεφτούμε ότι αυτά τα σενάρια απαιτείται να εκτελεστούν μετά από κάθε νέα έκδοση του συστήματος. Για το λόγο αυτό και στα πλαίσια της συγκεκριμένης εργασίας θα προσπαθήσουμε να αυτοματοποιήσουμε τα σενάρια ελέγχου που δημιουργήσαμε και να δούμε σε τι ποσοστό καταφέραμε να το επιτύχουμε και να εξάγουμε χρήσιμα συμπεράσματα.

Το εργαλείο που επιλέξαμε για την εκτέλεση του αυτοματοποιημένου ελέγχου είναι το *Robot Framework* όπως παρουσιάστηκε στο κεφάλαιο 4 γιατί επιτρέπει να αυτοματοποιήσουμε τα σενάρια ελέγχου που δημιουργήσαμε βάση της τεχνικής λέξεων-κλειδιών (keywords)[2] που αποτελεί μια κατανοητή τεχνική και επιπρόσθετα προσφέρει τη δυνατότητα δημιουργίας αναφορών που θα μας βοηθήσουν άμεσα στο να δούμε τα αποτελέσματα του ελέγχου μας. Για την εγκατάσταση του συγκεκριμένου εργαλείου χρησιμοποιήσαμε τη γλώσσα προγραμματισμού *Python* [45] [47]. Επιπρόσθετα για να επιτύχουμε την αυτοματοποιημένη πλοήγηση στο γραφικό περιβάλλον της εφαρμογής μας θα χρησιμοποιήσουμε την εξωτερική βιβλιοθήκη *SeleniumLibrary* καθώς και την ενσωματωμένη βιβλιοθήκη *BuildIn* που μας προσφέρουν πλήθος λέξεων-κλειδιών για να επιτύχουμε το στόχο μας [40]. Τέλος επιλέξαμε το

[2] Για την παρούσα εργασία αντί να χρησιμοποιούμε τις ελληνικές λέξεις “λέξεις-κλειδιά” θα χρησιμοποιούμε την αγγλική λέξη “keywords” που είναι ευρέως γνωστή και επιστημονικά αποδεκτή.

PyCharm σαν IDE για να μας βοηθήσει σε περίπτωση λαθών στον κώδικα και φυσικά στην αυτόματη συμπλήρωση εντολών [46].

Όπως ήδη αναφέρθηκε τα keywords παίζουν βασικό ρόλο στο εργαλείο αυτό. Υπάρχουν έτοιμα keywords από τις βιβλιοθήκες που αναφέραμε (π.χ. *Click Button*) και χρησιμοποιώντας τα μπορούμε να φτιάξουμε τα δικά μας που ουσιαστικά θα αποτελούν μια αλληλουχία keywords δηλαδή αλληλουχία ενεργειών στο γραφικό περιβάλλον της εφαρμογής. Όλα τα keywords ορίζονται κάτω από τον τίτλο *****Keywords*****.

Έτσι τα πρώτα 2 keywords που δημιουργήσαμε που αποτελούν την αρχή και το τέλος κάθε σεναρίου ελέγχου είναι:

- *Begin Web Test* που ουσιαστικά ανοίγει την επιθυμητή ιστοσελίδα που θέλουμε στο πρόγραμμα περιήγησης που επιθυμούμε και μεγιστοποιεί το παράθυρο. Οι ενέργειες αυτές πραγματοποιούνται με τα ήδη υπάρχοντα keywords *open browser* και *maximize browser window*. Η αλληλουχία αυτών των keywords δημιουργεί το δικό μας keyword.
- *End Web Test* που ουσιαστικά κλείνει το πρόγραμμα περιήγησης. Εδώ χρησιμοποιήσαμε το ήδη υπάρχον keyword *close browser* που εκτελεί την επιθυμητή ενέργεια.

```
*** Keywords ***
Begin Web Test
    open browser  ${URL}  ${BROWSER}
    maximize browser window

End Web Test
    close browser
```

Τα δύο αυτά keywords επιθυμούμε να εκτελούνται σε κάθε σενάριο ελέγχου ανεξαρτήτως αποτελέσματος οπότε χρησιμοποιήσαμε τις εντολές *Test Setup* και *Test Teardown*.

```
Test Setup  Begin Web Test
Test Teardown  End Web Test
```

Επίσης, όπως φαίνεται και στο παρακάτω κομμάτι κώδικα, το πρόγραμμα περιήγησης, η ιστοσελίδα και τα username και password ενός χρήστη έχουν οριστεί σαν μεταβλητές τιμές κάτω από τον τίτλο *****Variables***** οπότε εύκολα αν θέλουμε να τα αλλάξουμε, αλλάζουμε την τιμή μόνο στη μεταβλητή και όχι σε όλα τα σενάρια ελέγχου. Αυτό θα μας βοηθήσει να ελέγξουμε άμεσα τα στοιχεία εισόδου και να συνδεθούμε με τον προβληματικό χρήστη απλά αλλάζοντας μια τιμή και όχι επαναλαμβάνοντας την ίδια διαδικασία για όλα τα σενάριά μας.

```
*** Variables ***
${BROWSER} =  gc
${URL} =  https://www.saucedemo.com/
${USERNAME} =  standard_user
${PASSWORD} =  secret_sauce
```

Στη συνέχεια και ακολουθώντας την πρακτική του μοντέλου αντικειμένου σελίδας (Page Object Model- POM) που ουσιαστικά αποτελεί ένα μοτίβο σχεδίασης ευρέως διαδεδομένο στον αυτοματοποιημένο έλεγχο δημιουργήσαμε τα keywords για όλα τα σενάρια ελέγχου [47] [48] [49]. Ο λόγος επιλογής του POM είναι ότι μειώνει την αντιγραφή του κώδικα, βελτιώνει τη συντήρηση των ελέγχων και γενικά κρατάει το αρχείο με τα σενάρια ελέγχου «καθαρό» δηλαδή

δεν δημιουργούνται στο συγκεκριμένο αρχείο τα keywords αλλά φτιάχνονται άλλα αρχεία όπου κάθε ένα από αυτά είναι για μια σελίδα της εφαρμογής και εκεί δημιουργούνται τα keywords που επιθυμούμε.

Έτσι δημιουργήθηκαν 10 αρχεία και η δομή τους παρουσιάζεται παρακάτω[3]:

- ***Swag_Labs.robot:*** Το βασικό αρχείο που περιλαμβάνει κάτω από τα ***** Test Cases ***** όλα τα σενάρια ελέγχου που δημιουργήσαμε. Μέσω των ***** Settings ***** επικοινωνεί με όλα τα υπόλοιπα αρχεία για να αντλήσει τα keywords που χρειάζεται.
- ***Swag_Lags_GUI.robot****:* Το δεύτερο πιο σημαντικό αρχείο καθώς σε αυτό διοχετεύονται όλα τα keywords από τα αρχεία σελίδων που θα παρουσιαστούν παρακάτω. Στο συγκεκριμένο αρχείο δημιουργείται το τελικό keyword που θα εκτελεί μια αλληλουχία ενεργειών για να ελέγξει το σενάριο που επιθυμούμε.
- ***CommonWeb.robot****:* Το αρχείο που περιλαμβάνει τα βασικά keywords ανοίγματος και κλεισίματος περιηγητή.
- ***Cart_Page.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα καλαθιού.
- ***Checkout_Complete.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα Checkout_Complete.
- ***Checkout_Information.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα Checkout_Information.
- ***Checkout_Overview.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα Checkout_Overview.
- ***Login_Page.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα εισόδου χρήστη.
- ***Product_Page.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν την σελίδα όλων τον προϊόντων.
- ***Specific_Product_Page.robot:*** Το αρχείο που περιλαμβάνει τα keywords που αφορούν τη σελίδα συγκεκριμένου προϊόντος.

Η σύνδεση μεταξύ των παραπάνω αρχείων ορίζεται κάτω από τον τίτλο ***** Settings ***** με την εντολή *Resource* και στη συνέχεια το μονοπάτι (path) του αρχείου που θέλουμε.

```
*** Settings ***
Documentation  This TestSuite is created for my Master Project and will check
SauceDemo website
Resource  ../Resources/CommonWeb.robot
Resource  ../Resources/Swag_Labs_GUI.robot
```

Στο σημείο αυτό θα παρουσιαστεί ένα από πολλά σενάρια ελέγχου που δημιουργήσαμε και αυτό είναι ο έλεγχος της τιμής ενός προϊόντος στη σελίδα όλων των προϊόντων σε σύγκριση με την τιμή του στη σελίδα του ίδιου του προϊόντος.

[3] Ο κώδικας της εφαρμογής έχει «ανέβει» στο GitHub και το URL για να εμφανιστεί μπορεί να βρεθεί στο Παράρτημα Ι.

Για τον έλεγχο αυτό δημιουργήσαμε στο αρχείο Swag_Labs_GUI.robot το keyword *Price checking product A* που με τη χρήση των ενσωματωμένων keyword *Get Text, Click Element, Should Be Equal* ελέγχει αυτό που επιθυμούμε.

```
Price checking product A
  ${APrice}=   Get Text  //*[@id="inventory_container"]/div/div[1]/div[2]/div[2]/div
  Click Element  //*[@id="item_4_title_link"]/div
  ${APrice1}=   Get Text  //*[@id="inventory_item_container"]/div/div/div[2]/div[3]
  Should Be Equal  ${APrice}  ${APrice1}
```

Στη συνέχεια, στο αρχείο Swag_Labs.robot κάτω από τον τίτλο ***** Test Cases ***** ορίσαμε το σενάριο ελέγχου *Price Verification A* που θα «καλέσει» το keyword που δημιουργήσαμε και θα εκτελέσει την αλληλουχία των ενεργειών. Εδώ για να μην υπάρξει σύγχυση καλούμε και ένα άλλο keyword (*Valid User Login*) που έχει δημιουργηθεί για την επιτυχή σύνδεση του χρήστη στην ιστοσελίδα. Εκτός των keywords, στα σενάρια ελέγχου ορίζουμε το **[Documentation]** που ουσιαστικά αποτελεί μια περιγραφή του τι θα ελέγξουμε και τα **[Tags]** που ουσιαστικά έχουν την ίδια λογική με τις ετικέτες που αναφέραμε στο προηγούμενο υπό-κεφάλαιο.

```
Price verification A
    [Documentation]  This is TC33A that will check if Price of product A is the
same in both product page and specific product page
    [Tags]  Regression
    Swag_Labs_GUI.Valid User Login
    Swag_Labs_GUI.Price checking product A
```

Ο παραπάνω έλεγχος μπορεί να τρέξει με την εντολή *robot -d results -t "Price verification A" tests* και τα αποτελέσματα αν επιλέξουμε να κάνουμε Login με τον προβληματικό χρήστη φαίνονται στην παρακάτω αναφορά.

Tests Log

Generated
20211208 14:04:09 UTC+02:00
20 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	1	0	1	0	00:00:08	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Regression	1	0	1	0	00:00:08	

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Tests	1	0	1	0	00:00:09	
Tests . Swag Labs	1	0	1	0	00:00:08	

Test Execution Log

SUITE Tests
Full Name: Tests
Source: D:\Users\user\Desktop\Master_Project\tests
Start / End / Elapsed: 20211208 14:04:01.276 / 20211208 14:04:09.781 / 00:00:08.505
Status: 1 test total, 0 passed, 1 failed, 0 skipped

SUITE Swag Labs
Full Name: Tests.Swag Labs
Documentation: This TestSuite is created for my Master Project and will check SauceDemo website
Source: D:\Users\user\Desktop\Master_Project\tests\Swag_Labs.robot
Start / End / Elapsed: 20211208 14:04:01.317 / 20211208 14:04:09.778 / 00:00:08.461
Status: 1 test total, 0 passed, 1 failed, 0 skipped

TEST Price verification A
Full Name: Tests.Swag Labs.Price verification A
Documentation: This is TC33A that will check if Price of product A is the same in both product page and specific product page
Tags: Regression
Start / End / Elapsed: 20211208 14:04:01.778 / 20211208 14:04:09.775 / 00:00:07.997
Status: FAIL
Message: $29.99 != $49.99

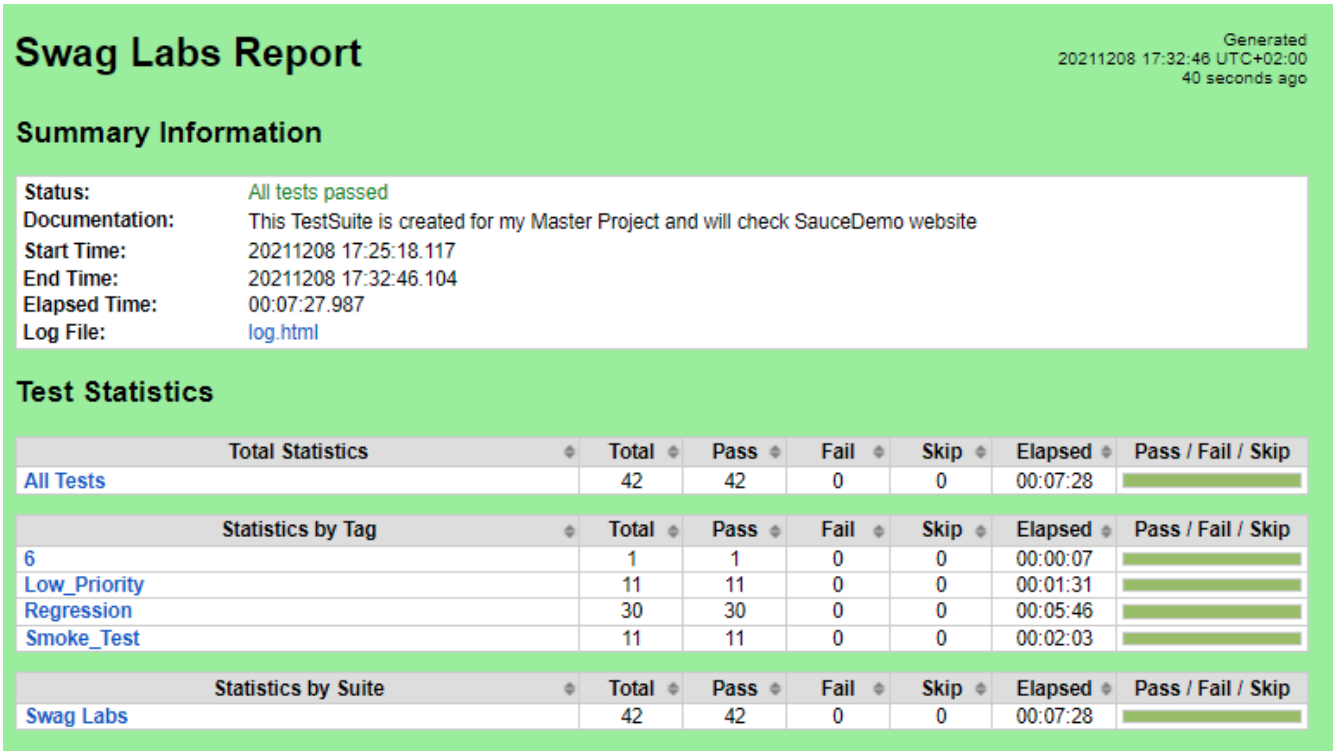
SETUP CommonWeb . Begin Web Test
KEYWORD Swag_Labs_GUI . Valid User Login
KEYWORD Swag_Labs_GUI . Price checking product A
Start / End / Elapsed: 20211208 14:04:07.309 / 20211208 14:04:07.484 / 00:00:00.175
KEYWORD ${APrice} = SeleniumLibrary . Get Text //*[@id="inventory_container"]/div/div[1]/div[2]/div[2]/div
KEYWORD SeleniumLibrary . Click Element //*[@id="item_4_title_link"]/div
KEYWORD ${APrice1} = SeleniumLibrary . Get Text //*[@id="inventory_item_container"]/div/div/div[2]/div[3]
KEYWORD BuiltIn . Should Be Equal ${APrice}, ${APrice1}
Documentation: Fails if the given objects are unequal.
Start / End / Elapsed: 20211208 14:04:07.483 / 20211208 14:04:07.484 / 00:00:00.001
14:04:07.484 FAIL $29.99 != $49.99

Εικόνα 22: Αποτελέσματα ελέγχου «Price verification A»

Έτσι παρατηρούμε ότι ο συγκεκριμένος έλεγχος με τον προβληματικό χρήστη δεν πέρασε (fail), γιατί η τιμή του προϊόντος στην αρχική σελίδα είναι άλλη σε σχέση με την τιμή του προϊόντος στη σελίδα του συγκεκριμένου προϊόντος (FAIL $29.99 != $49.99).

Αν θέλαμε να τρέξουμε σενάρια ελέγχου με συγκεκριμένα **Tags** θα χρησιμοποιούσαμε την εντολή *robot –include Low_Priority -d results tests/Swag_Labs.robot* και τέλος αν θέλαμε να τρέξουμε όλα τα σενάρια ελέγχου θα χρησιμοποιούσαμε την εντολή *robot -d results tests/Swag_Labs.robot*

«Τρέχοντας» την τελευταία εντολή όλα τα αυτοματοποιημένα σενάρια ελέγχου εκτελούνται και τα αποτελέσματα που προκύπτουν με τον προβληματικό και το μη-προβληματικό χρήστη παρουσιάζονται στις παρακάτω αναφορές[4] [5].



Swag Labs Report

Generated
20211208 17:32:46 UTC+02:00
40 seconds ago

Summary Information

Status:	All tests passed
Documentation:	This TestSuite is created for my Master Project and will check SauceDemo website
Start Time:	20211208 17:25:18.117
End Time:	20211208 17:32:46.104
Elapsed Time:	00:07:27.987
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	42	42	0	0	00:07:28	

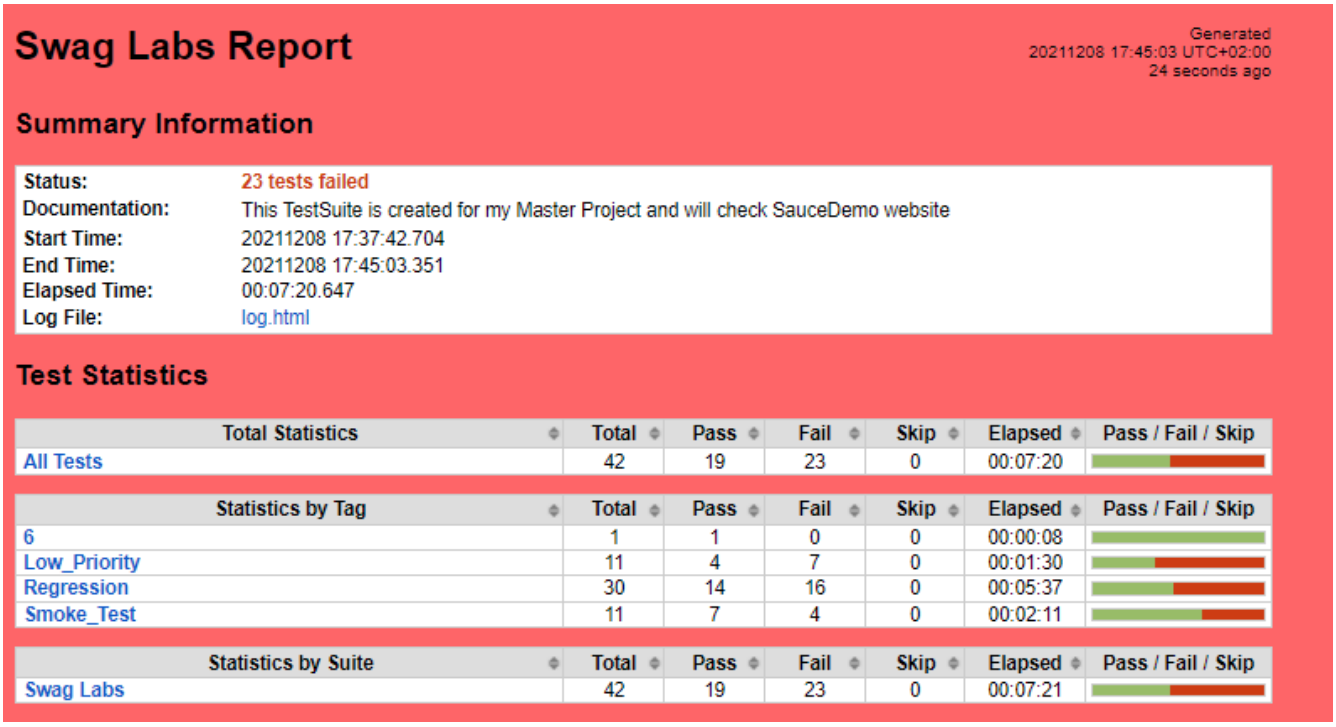
Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
6	1	1	0	0	00:00:07	
Low_Priority	11	11	0	0	00:01:31	
Regression	30	30	0	0	00:05:46	
Smoke_Test	11	11	0	0	00:02:03	

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Swag Labs	42	42	0	0	00:07:28	

Εικόνα 23: Αποτελέσματα όλων των σεναρίων ελέγχου με το μη-προβληματικό χρήστη

[4] Οι 2 αναφορές που θα παρουσιαστούν αποτυπώνουν τη γενική εικόνα των σεναρίων ελέγχου. Οι αναλυτικές αναφορές μπορούν να βρεθούν στο Παράρτημα ΙΙ της παρούσας διπλωματικής εργασίας.

[5] Οι αναφορές περιλαμβάνουν και ένα επιπλέον Tag (6). Αυτό συμβαίνει αυτόματα από το Robot Framework λόγω της εντολής “Pass Execution If ${Items}= 6 ” που χρησιμοποιήσαμε στο TC12.



Swag Labs Report

Generated
20211208 17:45:03 UTC+02:00
24 seconds ago

Summary Information

Status:	23 tests failed
Documentation:	This TestSuite is created for my Master Project and will check SauceDemo website
Start Time:	20211208 17:37:42.704
End Time:	20211208 17:45:03.351
Elapsed Time:	00:07:20.647
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	42	19	23	0	00:07:20	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
6	1	1	0	0	00:00:08	
Low_Priority	11	4	7	0	00:01:30	
Regression	30	14	16	0	00:05:37	
Smoke_Test	11	7	4	0	00:02:11	

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Swag Labs	42	19	23	0	00:07:21	

Εικόνα 24: Αποτελέσματα όλων των σεναρίων ελέγχου με τον προβληματικό χρήστη

6. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ

6.1 Συμπεράσματα [6]

Αρχικά, το πρώτο συμπέρασμα που προκύπτει είναι πόσο σημαντικός και απαραίτητος είναι ο «Έλεγχος Λογισμικού» την εποχή αυτή. Μέσω των διαδικασιών του, εξετάζεται η ποιότητα του προϊόντος αλλά ταυτόχρονα ελέγχεται και κατά πόσο το προϊόν ικανοποιεί και τις απαιτήσεις του πελάτη.

Στη συνέχεια και μέσω του αυτοματοποιημένου ελέγχου λογισμικού πετύχαμε να αυτοματοποιήσουμε 42 από τα 44 σενάρια που είχαν προκύψει μέσω της ανάλυσης και δημιουργίας σεναρίων ελέγχων. Αυτό το ποσοστό είναι αρκετά υψηλό και συμπεραίνουμε ότι ο αυτοματοποιημένος έλεγχος μπορεί να καλύψει σε μεγάλο βαθμό τον χειρωνακτικό. Ωστόσο παρατηρήσαμε προβλήματα στο σύστημα που δεν είχαμε καταγράψει στα σενάρια ελέγχου οπότε δεν είχαν αυτοματοποιηθεί. Άρα ο χειρωνακτικός έλεγχος κατά την άποψη του συγγραφέα θεωρείται ιδιαίτερα σημαντικός και δεν πρέπει να εξαλειφθεί.

Επιπρόσθετα, συγκρίνοντας τις 2 αναφορές παρατηρούμε ότι όλα τα σενάρια ελέγχου εκτελέστηκαν επιτυχώς με το μη-προβληματικό χρήστη, ενώ αντίθετα 23 σενάρια ελέγχου απέτυχαν με τον προβληματικό χρήστη. Μάλιστα 4 από αυτά είχαν την ετικέτα Smoke_Test που σημαίνει ότι είναι από τα πιο σημαντικά και αυτόματα σημαίνει ότι μέσω αυτού του χρήστη η εφαρμογή δεν είναι πλήρως λειτουργική. Αν υποθέσουμε ότι ο μη-προβληματικός χρήστης αποτελεί την 1η έκδοση του συστήματος και ο προβληματικός χρήστης την αμέσως επόμενη που εγκαταστάθηκε στο περιβάλλον ελέγχου λογισμικού τότε η δεύτερη έκδοση δεν πρέπει να προχωρήσει και να εγκατασταθεί στο περιβάλλον του πελάτη.

Ένα άλλο συμπέρασμα που προέκυψε είναι πόσο πολύς χρόνος μπορεί να κερδηθεί μέσω των αυτοματοποιημένων ελέγχων. Παρατηρώντας τις αναφορές βλέπουμε πως όλα τα σενάρια ελέγχου «εκτελέστηκαν» σε 7 λεπτά και 20 δευτερόλεπτα. Σε αντίθετη περίπτωση αν όλο αυτό έπρεπε να γίνει χειρωνακτικά θα χρειάζονταν πολύ παραπάνω χρόνος και ειδικά τα πιο περίπλοκα σενάρια ίσως δεν εκτελούνται καν αν η πίεση χρόνου ήταν υψηλή.

Το τελικό γενικό συμπέρασμα που προκύπτει με την ολοκλήρωση αυτής της εργασίας είναι πως ο έλεγχος λογισμικού αποτελεί αναπόσπαστο κομμάτι του κύκλου ζωής ανάπτυξης λογισμικού και πως με την κατάλληλη εκπαίδευση των testers και τη χρήση των κατάλληλων εργαλείων αυτοματοποιημένου ελέγχου μπορούν να επιτευχθούν τρομερά αποτελέσματα που θα βελτιώσουν την ποιότητα των λογισμικών παγκοσμίως αλλά και θα κερδηθεί χρόνος και χρήμα από τις εταιρίες. Όλο αυτό βέβαια για να επιτευχθεί, απαιτεί τη σωστή συνεργασία και συνύπαρξη δυο διαφορετικών ομάδων ελέγχου λογισμικού, των manual testers και των automation testers.

[6] Όλα τα συμπεράσματα της παρούσας μεταπτυχιακής διπλωματικής εργασίας εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθούν ότι αντιπροσωπεύουν τις επίσημες θέσεις των επιστημόνων πάνω στον τομέα του «Ελέγχου Λογισμικού»

6.2 Μελλοντικές βελτιώσεις

Το θέμα της εργασίας αναλύθηκε εκτενώς και θα μπορούσε να χρησιμοποιηθεί περεταίρω στην έρευνα κυρίως του αυτοματοποιημένου ελέγχου λογισμικού.

Κάποιες μελλοντικές βελτιώσεις που θα μπορούσαν να πραγματοποιηθούν είναι να γίνει προσπάθεια να αυτοματοποιηθούν όλα τα σενάρια ελέγχου που ήδη έχουν δημιουργηθεί καθώς επίσης και να γίνει δημιουργία καινούριων σεναρίων που θα καλύπτουν περισσότερες περιοχές του συστήματος.

Επιπρόσθετα, στα πλαίσια της χρήσης του εργαλείου Robot Framework σίγουρα μπορεί να βελτιωθεί ο τρόπος συγγραφής του κώδικα. Κάποια σενάρια ελέγχου επαναλαμβάνουν το ίδιο κομμάτι κώδικα οπότε αυτό ίσως μπορεί να συμπτυχθεί και να μειωθεί ο όγκος του κώδικα που έχουμε δημιουργήσει.

Τέλος, μια πρόκληση που προκύπτει από την παρούσα εργασία είναι όλα όσα αναλύθηκαν να εφαρμοστούν σε μεγαλύτερου εύρους λογισμικά που χρησιμοποιούνται ή θα χρησιμοποιούνται από πολλούς χρήστες. Αυτή η πρόκληση αποτελεί και το μελλοντικό στόχο του συγγραφέα στα πλαίσια της επαγγελματικής του πορείας και εργασίας.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Software Development Life Cycle	Κύκλος Ζωής Ανάπτυξης Λογισμικού
Test environment	Περιβάλλον ελέγχου
Testing	Έλεγχος Λογισμικού
Tester	Δοκιμαστής
Change Request	Αίτημα Αλλαγής
Waterfall model	Μοντέλο καταρράκτη
Acceptance Testing	Έλεγχος Αποδοχής
Unit Testing	Έλεγχος Μονάδας
Integration Testing	Έλεγχος Ενσωμάτωσης
System Testing	Έλεγχος Συστήματος
Agile model	Ευκίνητο μοντέλο
Scrum	-
Sprint	Επανάληψη
Product Owner	Ιδιοκτήτης Προϊόντος
Bug	Έντομο
Software Testing Life Cycle	Κύκλος Ζωής Ελέγχου Λογισμικού
Functional Testing	Έλεγχος Λειτουργίας
Non-Functional Testing	Έλεγχος που δεν σχετίζεται με τη λειτουργία
Performance Testing	Έλεγχος Απόδοσης
Load Testing	Έλεγχος Φόρτωσης
Stress Testing	Έλεγχος Ακραίων Καταστάσεων
Usability Testing	Έλεγχος Ευχρηστίας
Regression Testing	Έλεγχος Παλινδρόμησης
Confirmation Testing	Έλεγχος Επιβεβαίωσης
Review	Ανασκόπηση

Cyclomatic Complexity	Κυκλωματική Πολυπλοκότητα
Manual Testing	«Χειρωνακτικός» Έλεγχος Λογισμικού
Scripts	Αυτοματοποιημένα Σενάρια
Integrated Development Environments	Περιβάλλοντα Ολοκληρωμένης Ανάπτυξης
Test Cases	Σενάρια Ελέγχου
Page Object Model	Μοντέλο Αντικειμένου Σελίδας
Keywords	Λέξεις-Κλειδιά

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ - ΑΚΡΩΝΥΜΙΑ

CR	Change Request
HL	High Level Design
IDE	Integrated Development Environments
LLD	Low Level Design
POM	Page Object Model
SDLC	Software Development Life Cycle
STLC	Software Testing Life Cycle
UAT	User Acceptance Testing

ΠΑΡΑΡΤΗΜΑ Ι

Μέσω του GitHub και συγκεκριμένα του παρακάτω URL μπορεί να βρεθεί ο κώδικας της εφαρμογής. Ο κώδικας βρίσκεται κάτω από το master branch και είναι private για λόγους πνευματικών δικαιωμάτων. Όποιος αναγνώστης επιθυμεί πρόσβαση, θα πρέπει να τη ζητήσει από το συγγραφέα της παρούσας διπλωματικής εργασίας.

https://github.com/VasilisTsakirakis/Master_Project/tree/master

ΠΑΡΑΡΤΗΜΑ ΙΙ

Στο παράρτημα ΙΙ παρουσιάζονται οι αναφορές με τα αναλυτικά αποτελέσματα των αυτοματοποιημένων ελέγχων λογισμικού για τον προβληματικό και το μη προβληματικό χρήστη.

All | Tags | Suites | Search

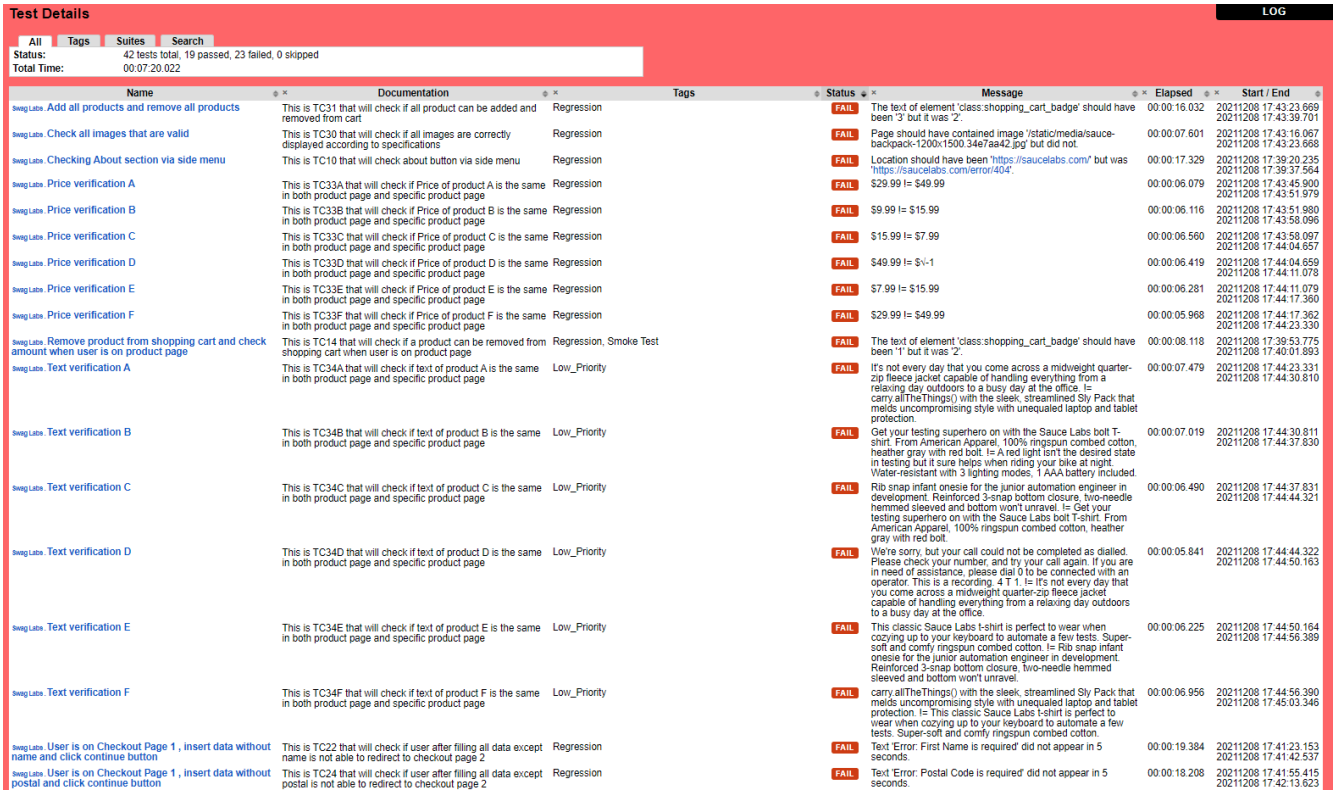
Status: 42 tests total, 42 passed, 0 failed, 0 skipped
Total Time: 00:07:27.501

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
Swag Labs. Add all products and remove all products	This is TC31 that will check if all product can be added and removed from cart	Regression	PASS		00:00:43.862	20211208 17:30:36.549 20211208 17:31:20.411
Swag Labs. Add product to shopping cart and check amount	This is TC13 that will check if a product is succesfully added to cart	Regression, Smoke Test	PASS		00:00:07.815	20211208 17:27:14.049 20211208 17:27:21.864
Swag Labs. Back button on Specific Product page	This is TC32 that will check if back button on specific product page is functional and redirect user to previous page	Low_Priority	PASS		00:00:06.393	20211208 17:31:20.412 20211208 17:31:26.805
Swag Labs. Check all images that are valid	This is TC30 that will check if all images are correctly displayed according to specifications	Regression	PASS		00:00:06.923	20211208 17:30:29.625 20211208 17:30:36.548
Swag Labs. Check amount of all products	This is TC12 that will check amount of all products and if this amount is the one from the specifications	6, Regression, Smoke Test	PASS	==	00:00:06.703	20211208 17:27:07.344 20211208 17:27:14.047
Swag Labs. Checking About section via side menu	This is TC10 that will check about button via side menu	Regression	PASS		00:00:23.068	20211208 17:26:44.274 20211208 17:27:07.342
Swag Labs. Click checkout button when in cart page and redirect to Checkout Page1	This is TC19 that will check if user can click checkout button and redirect to the next page	Regression, Smoke Test	PASS		00:00:06.394	20211208 17:28:11.004 20211208 17:28:17.398
Swag Labs. Go back to product page when user is on cart page	This is TC17 that will check if user can go back to home page when he is on cart page	Regression	PASS		00:00:06.780	20211208 17:27:58.508 20211208 17:28:05.288
Swag Labs. Locked user is not able to login	This is TC04 that will check if a locked user is able to login to the system	Regression	PASS		00:00:10.953	20211208 17:25:42.852 20211208 17:25:53.805
Swag Labs. Logout from side menu after logging	This is TC09 that will check if a user after logging to the system is able to logout via side menu	Regression, Smoke Test	PASS		00:00:16.854	20211208 17:26:27.419 20211208 17:26:44.273
Swag Labs. Navigate to cart page	This is TC15 that will check cart button and if user is redirected to cart page	Regression, Smoke Test	PASS		00:00:11.854	20211208 17:27:29.625 20211208 17:27:41.479
Swag Labs. No Password User is not able to login	This is TC06 that will check if a user without filling password is able to login to the system	Low_Priority	PASS		00:00:11.688	20211208 17:26:04.778 20211208 17:26:16.466
Swag Labs. No Username User is not able to login	This is TC07 that will check if a user without filling username is able to login to the system	Low_Priority	PASS		00:00:10.950	20211208 17:26:16.468 20211208 17:26:27.418
Swag Labs. No Username and Password user is not able to login	This is TC05 that will check if a user without username and password is able to login to the system		PASS		00:00:10.971	20211208 17:25:53.806 20211208 17:26:04.777
Swag Labs. Price verification A	This is TC33A that will check if Price of product A is the same in both product page and specific product page	Regression	PASS		00:00:06.736	20211208 17:31:26.806 20211208 17:31:33.542
Swag Labs. Price verification B	This is TC33B that will check if Price of product B is the same in both product page and specific product page	Regression	PASS		00:00:06.456	20211208 17:31:33.543 20211208 17:31:39.999
Swag Labs. Price verification C	This is TC33C that will check if Price of product C is the same in both product page and specific product page	Regression	PASS		00:00:07.180	20211208 17:31:40.000 20211208 17:31:47.180
Swag Labs. Price verification D	This is TC33D that will check if Price of product D is the same in both product page and specific product page	Regression	PASS		00:00:05.988	20211208 17:31:47.182 20211208 17:31:53.170
Swag Labs. Price verification E	This is TC33E that will check if Price of product E is the same in both product page and specific product page	Regression	PASS		00:00:06.337	20211208 17:31:53.171 20211208 17:31:59.508
Swag Labs. Price verification F	This is TC33F that will check if Price of product F is the same in both product page and specific product page	Regression	PASS		00:00:06.457	20211208 17:31:59.509 20211208 17:32:05.966
Swag Labs. Privacy check verification in product page	This is TC18 that will check if privacy text based on specifications is displayed properly	Regression	PASS		00:00:05.714	20211208 17:28:05.289 20211208 17:28:11.003
Swag Labs. Remove product from shopping cart and check amount when user is on product page	This is TC14 that will check if a product can be removed from shopping cart when user is on product page	Regression, Smoke Test	PASS		00:00:07.758	20211208 17:27:21.865 20211208 17:27:29.623
Swag Labs. Remove product when user is on cart page	This is TC16 that will check if a product can be removed from shopping cart when user is on cart page	Regression, Smoke Test	PASS		00:00:17.027	20211208 17:27:41.480 20211208 17:27:58.507
Swag Labs. Text verification A	This is TC34A that will check if text of product A is the same in both product page and specific product page	Low_Priority	PASS		00:00:06.856	20211208 17:32:05.967 20211208 17:32:12.823

Εικόνα 25: Λεπτομερή αποτελέσματα σεναρίων ελέγχου λογισμικού με το μη-προβληματικό χρήστη (Μέρος 1ο)

Swag Labs. Text verification B	This is TC34B that will check if text of product B is the same in both product page and specific product page	Low_Priority	PASS	00:00:06.739	20211208 17:32:12.824 20211208 17:32:19.563
Swag Labs. Text verification C	This is TC34C that will check if text of product C is the same in both product page and specific product page	Low_Priority	PASS	00:00:06.395	20211208 17:32:19.564 20211208 17:32:25.959
Swag Labs. Text verification D	This is TC34D that will check if text of product D is the same in both product page and specific product page	Low_Priority	PASS	00:00:06.539	20211208 17:32:25.960 20211208 17:32:32.499
Swag Labs. Text verification E	This is TC34E that will check if text of product E is the same in both product page and specific product page	Low_Priority	PASS	00:00:06.798	20211208 17:32:32.501 20211208 17:32:39.299
Swag Labs. Text verification F	This is TC34F that will check if text of product F is the same in both product page and specific product page	Low_Priority	PASS	00:00:06.801	20211208 17:32:39.300 20211208 17:32:46.101
Swag Labs. User is on Checkout Page 1 , and decide to go back to cart page	This is TC25 that will check if user is on Checkout Page 1 and want to go back to Cart page he is able to do this action	Low_Priority	PASS	00:00:07.936	20211208 17:29:24.734 20211208 17:29:32.670
Swag Labs. User is on Checkout Page 1 , insert data without name and click continue button	This is TC22 that will check if user after filling all data except name is not able to redirect to checkout page 2	Regression	PASS	00:00:12.968	20211208 17:28:44.428 20211208 17:28:57.396
Swag Labs. User is on Checkout Page 1 , insert data without postal and click continue button	This is TC24 that will check if user after filling all data except postal is not able to redirect to checkout page 2	Regression	PASS	00:00:12.938	20211208 17:29:11.795 20211208 17:29:24.733
Swag Labs. User is on Checkout Page 1 , insert data without surname and click continue button	This is TC23 that will check if user after filling all data except surname is not able to redirect to checkout page 2	Regression	PASS	00:00:14.397	20211208 17:28:57.397 20211208 17:29:11.794
Swag Labs. User is on Checkout Page 1 , insert empty data and click continue button	This is TC21 that will check if user after not filling any data is not able to redirect to checkout page 2	Regression	PASS	00:00:13.014	20211208 17:28:31.413 20211208 17:28:44.427
Swag Labs. User is on Checkout Page 1 , insert valid data and redirect to Checkout Page 2	This is TC20 that will check if user after filling valid data can redirect to checkout page 2	Regression, Smoke Test	PASS	00:00:14.012	20211208 17:28:17.399 20211208 17:28:31.411
Swag Labs. User is on Checkout Page 2 , and checking that all data are valid	This is TC26 that will check if user is on Checkout Page 2 all data are correct	Regression, Smoke Test	PASS	00:00:12.913	20211208 17:29:32.671 20211208 17:29:45.584
Swag Labs. User is on Checkout Page 2 and cancel the order	This is TC28 that will check if user is on Checkout Page and click the appropriate button then the order is cancelled	Regression	PASS	00:00:15.257	20211208 17:30:00.490 20211208 17:30:15.747
Swag Labs. User is on Checkout Page 2 and complete the order	This is TC27 that will check if user is on Checkout Page and click the appropriate button then the order is completed	Regression, Smoke Test	PASS	00:00:14.904	20211208 17:29:45.585 20211208 17:30:00.489
Swag Labs. User is on Complete order page and click to go back to product page	This is TC29 that will check if user is on Last Page of the website and click the appropriate button he is redirected to Product Page	Low_Priority	PASS	00:00:13.876	20211208 17:30:15.748 20211208 17:30:29.624
Swag Labs. Valid user is able to login	This is TC01 that will check if a valid user is able to login to the system	Regression, Smoke_Test	PASS	00:00:06.331	20211208 17:25:18.550 20211208 17:25:24.881
Swag Labs. Wrong Password user is not able to login	This is TC03 that will check if a user with invalid password is able to login to the system	Regression	PASS	00:00:11.625	20211208 17:25:31.225 20211208 17:25:42.850
Swag Labs. Wrong Username user is not able to login	This is TC02 that will check if a user with invalid username is able to login to the system	Regression	PASS	00:00:06.341	20211208 17:25:24.883 20211208 17:25:31.224

Εικόνα 26: Λεπτομερή αποτελέσματα σεναρίων ελέγχου λογισμικού με το μη-προβληματικό χρήστη (Μέρος 2ο)



Test Details LOG

All | Tags | Suites | Search

Status: 42 tests total, 19 passed, 23 failed, 0 skipped
Total Time: 00:07:20.022

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
SwagLabs.Add all products and remove all products	This is TC31 that will check if all product can be added and removed from cart	Regression	FAIL	The text of element 'class:shopping_cart_badge' should have been '3' but it was '2'.	00:00:16.032	20211208 17:43:23.669 20211208 17:43:39.701
SwagLabs.Check all images that are valid	This is TC30 that will check if all images are correctly displayed according to specifications	Regression	FAIL	Page should have contained image '/static/media/sauce-backpack-1200x1500.34e7aa42.jpg' but did not.	00:00:07.601	20211208 17:43:16.067 20211208 17:43:23.668
SwagLabs.Checking About section via side menu	This is TC10 that will check about button via side menu	Regression	FAIL	Location should have been 'https://saucelabs.com/' but was 'https://saucelabs.com/error/404'.	00:00:17.329	20211208 17:39:20.235 20211208 17:39:37.564
SwagLabs.Price verification A	This is TC33A that will check if Price of product A is the same in both product page and specific product page	Regression	FAIL	$29.99 != $49.99	00:00:06.079	20211208 17:43:45.900 20211208 17:43:51.979
SwagLabs.Price verification B	This is TC33B that will check if Price of product B is the same in both product page and specific product page	Regression	FAIL	$9.99 != $15.99	00:00:06.116	20211208 17:43:51.980 20211208 17:43:58.096
SwagLabs.Price verification C	This is TC33C that will check if Price of product C is the same in both product page and specific product page	Regression	FAIL	$15.99 != $7.99	00:00:06.560	20211208 17:43:58.097 20211208 17:44:04.657
SwagLabs.Price verification D	This is TC33D that will check if Price of product D is the same in both product page and specific product page	Regression	FAIL	$49.99 != $\-1	00:00:06.419	20211208 17:44:04.659 20211208 17:44:11.078
SwagLabs.Price verification E	This is TC33E that will check if Price of product E is the same in both product page and specific product page	Regression	FAIL	$7.99 != $15.99	00:00:06.281	20211208 17:44:11.079 20211208 17:44:17.360
SwagLabs.Price verification F	This is TC33F that will check if Price of product F is the same in both product page and specific product page	Regression	FAIL	$29.99 != $49.99	00:00:05.968	20211208 17:44:17.362 20211208 17:44:23.330
SwagLabs.Remove product from shopping cart and check amount when user is on product page	This is TC14 that will check if a product can be removed from shopping cart when user is on product page	Regression, Smoke Test	FAIL	The text of element 'class:shopping_cart_badge' should have been '1' but it was '2'.	00:00:08.118	20211208 17:39:53.775 20211208 17:40:01.893
SwagLabs.Text verification A	This is TC34A that will check if text of product A is the same in both product page and specific product page	Low_Priority	FAIL	It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office. != carryAllTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.	00:00:07.479	20211208 17:44:23.331 20211208 17:44:30.810
SwagLabs.Text verification B	This is TC34B that will check if text of product B is the same in both product page and specific product page	Low_Priority	FAIL	Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt. != A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.	00:00:07.019	20211208 17:44:30.811 20211208 17:44:37.830
SwagLabs.Text verification C	This is TC34C that will check if text of product C is the same in both product page and specific product page	Low_Priority	FAIL	Rib snap infant onesie for the junior automation engineer in development. Reinforced 3-snap bottom closure, two-needle hemmed sleeved and bottom won't unravel. != Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.	00:00:06.490	20211208 17:44:37.831 20211208 17:44:44.321
SwagLabs.Text verification D	This is TC34D that will check if text of product D is the same in both product page and specific product page	Low_Priority	FAIL	We're sorry, but your call could not be completed as dialled. Please check your number, and try your call again. If you are in need of assistance, please dial 0 to be connected with an operator. This is a recording. 4 T 1. != It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.	00:00:05.841	20211208 17:44:44.322 20211208 17:44:50.163
SwagLabs.Text verification E	This is TC34E that will check if text of product E is the same in both product page and specific product page	Low_Priority	FAIL	This classic Sauce Labs t-shirt is perfect to wear when cozying up to your keyboard to automate a few tests. Super-soft and comfy ringspun combed cotton. != Rib snap infant onesie for the junior automation engineer in development. Reinforced 3-snap bottom closure, two-needle hemmed sleeved and bottom won't unravel.	00:00:06.225	20211208 17:44:50.164 20211208 17:44:56.389
SwagLabs.Text verification F	This is TC34F that will check if text of product F is the same in both product page and specific product page	Low_Priority	FAIL	carryAllTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection. != This classic Sauce Labs t-shirt is perfect to wear when cozying up to your keyboard to automate a few tests. Super-soft and comfy ringspun combed cotton.	00:00:06.956	20211208 17:44:56.390 20211208 17:45:03.346
SwagLabs.User is on Checkout Page 1 , insert data without name and click continue button	This is TC22 that will check if user after filling all data except name is not able to redirect to checkout page 2	Regression	FAIL	Text 'Error: First Name is required' did not appear in 5 seconds.	00:00:19.384	20211208 17:41:23.153 20211208 17:41:42.537
SwagLabs.User is on Checkout Page 1 , insert data without postal and click continue button	This is TC24 that will check if user after filling all data except postal is not able to redirect to checkout page 2	Regression	FAIL	Text 'Error: Postal Code is required' did not appear in 5 seconds.	00:00:18.208	20211208 17:41:55.415 20211208 17:42:13.623

Εικόνα 27: Λεπτομερή αποτελέσματα σεναρίων ελέγχου λογισμικού με τον προβληματικό χρήστη (Μέρος 1ο)

Swag Labs. User is on Checkout Page 1 , insert valid data and redirect to Checkout Page 2	This is TC20 that will check if user after filling valid data can redirect to checkout page 2	Regression, Smoke Test	FAIL	Location should have been 'https://www.saucedemo.com/checkout-step-two.html' but was 'https://www.saucedemo.com/checkout-step-one.html'.	00:00:13.432	20211208 17:40:56.208 20211208 17:41:09.640
Swag Labs. User is on Checkout Page 2 , and checking that all data are valid	This is TC26 that will check if user is on Checkout Page 2 all data are correct	Regression, Smoke Test	FAIL	Location should have been 'https://www.saucedemo.com/checkout-step-two.html' but was 'https://www.saucedemo.com/checkout-step-one.html'.	00:00:12.892	20211208 17:42:21.644 20211208 17:42:34.536
Swag Labs. User is on Checkout Page 2 and cancel the order	This is TC28 that will check if user is on Checkout Page and click the appropriate button then the order is cancelled	Regression	FAIL	Location should have been 'https://www.saucedemo.com/checkout-step-two.html' but was 'https://www.saucedemo.com/checkout-step-one.html'.	00:00:14.944	20211208 17:42:48.070 20211208 17:43:03.014
Swag Labs. User is on Checkout Page 2 and complete the order	This is TC27 that will check if user is on Checkout Page and click the appropriate button then the order is completed	Regression, Smoke Test	FAIL	Location should have been 'https://www.saucedemo.com/checkout-step-two.html' but was 'https://www.saucedemo.com/checkout-step-one.html'.	00:00:13.530	20211208 17:42:34.538 20211208 17:42:48.068
Swag Labs. User is on Complete order page and click to go back to product page	This is TC29 that will check if user is on Last Page of the website and click the appropriate button he is redirected to Product Page	Low_Priority	FAIL	Location should have been 'https://www.saucedemo.com/checkout-step-two.html' but was 'https://www.saucedemo.com/checkout-step-one.html'.	00:00:12.998	20211208 17:43:03.016 20211208 17:43:16.014
Swag Labs. Add product to shopping cart and check amount	This is TC13 that will check if a product is succesfully added to cart	Regression, Smoke Test	PASS		00:00:07.963	20211208 17:39:45.782 20211208 17:39:53.745
Swag Labs. Back button on Specific Product page	This is TC32 that will check if back button on specific product page is functional and redirect user to previous page	Low_Priority	PASS		00:00:06.197	20211208 17:43:39.702 20211208 17:43:45.899
Swag Labs. Check amount of all products	This is TC12 that will check amount of all products and if this amount is the one from the specifications	6, Regression, Smoke Test	PASS	==	00:00:08.215	20211208 17:39:37.565 20211208 17:39:45.780
Swag Labs. Click checkout button when in cart page and redirect to Checkout Page1	This is TC19 that will check if user can click checkout button and redirect to the next page	Regression, Smoke Test	PASS		00:00:07.166	20211208 17:40:49.040 20211208 17:40:56.206
Swag Labs. Go back to product page when user is on cart page	This is TC17 that will check if user can go back to home page when he is on cart page	Regression	PASS		00:00:07.440	20211208 17:40:32.415 20211208 17:40:39.855
Swag Labs. Locked user is not able to login	This is TC04 that will check if a locked user is able to login to the system	Regression	PASS		00:00:13.296	20211208 17:38:10.173 20211208 17:38:23.469
Swag Labs. Logout from side menu after logging	This is TC09 that will check if a user after logging to the system is able to logout via side menu	Regression, Smoke Test	PASS		00:00:21.281	20211208 17:38:58.951 20211208 17:39:20.232
Swag Labs. Navigate to cart page	This is TC15 that will check cart button and if user is redirected to cart page	Regression, Smoke Test	PASS		00:00:12.620	20211208 17:40:01.895 20211208 17:40:14.515
Swag Labs. No Password User is not able to login	This is TC06 that will check if a user without filling password is able to login to the system	Low_Priority	PASS		00:00:11.570	20211208 17:38:35.836 20211208 17:38:47.406
Swag Labs. No Username User is not able to login	This is TC07 that will check if a user without filling username is able to login to the system	Low_Priority	PASS		00:00:11.525	20211208 17:38:47.425 20211208 17:38:58.950
Swag Labs. No Username and Password user is not able to login	This is TC05 that will check if a user without username and password is able to login to the system		PASS		00:00:12.365	20211208 17:38:23.470 20211208 17:38:35.835
Swag Labs. Privacy check verification in product page	This is TC18 that will check if privacy text based on specifications is displayed properly	Regression	PASS		00:00:09.183	20211208 17:40:39.856 20211208 17:40:49.039
Swag Labs. Remove product when user is on cart page	This is TC16 that will check if a product can be removed from shopping cart when user is on cart page	Regression, Smoke Test	PASS		00:00:17.898	20211208 17:40:14.516 20211208 17:40:32.414
Swag Labs. User is on Checkout Page 1 , and decide to go back to cart page	This is TC25 that will check if user is on Checkout Page 1 and want to go back to Cart page he is able to do this action	Low_Priority	PASS		00:00:08.018	20211208 17:42:13.624 20211208 17:42:21.642
Swag Labs. User is on Checkout Page 1 , insert data without surname and click continue button	This is TC23 that will check if user after filling all data except surname is not able to redirect to checkout page 2	Regression	PASS		00:00:12.838	20211208 17:41:42.576 20211208 17:41:55.414
Swag Labs. User is on Checkout Page 1 , insert empty data and click continue button	This is TC21 that will check if user after not filling any data is not able to redirect to checkout page 2	Regression	PASS		00:00:13.510	20211208 17:41:09.642 20211208 17:41:23.152
Swag Labs. Valid user is able to login	This is TC01 that will check if a valid user is able to login to the system	Regression, Smoke_Test	PASS		00:00:07.926	20211208 17:37:43.134 20211208 17:37:51.060
Swag Labs. Wrong Password user is not able to login	This is TC03 that will check if a user with invalid password is able to login to the system	Regression	PASS		00:00:11.276	20211208 17:37:58.896 20211208 17:38:10.172
Swag Labs. Wrong Username user is not able to login	This is TC02 that will check if a user with invalid username is able to login to the system	Regression	PASS		00:00:07.834	20211208 17:37:51.061 20211208 17:37:58.895

Εικόνα 28: Λεπτομερή αποτελέσματα σεναρίων ελέγχου λογισμικού με τον προβληματικό χρήστη (Μέρος 2ο)

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] “Certified Tester Foundation Level Syllabus.” International Software Testing Qualifications Board, Nov. 11, 2019. [Online]. Available: https://www.istqb.org/certification-path-root/foundation-level-2018.html#syllabus (accessed May. 2, 2021).

[2] “SDLC (Software Development Life Cycle) Phases, Process, Models,” Nov. 01, 2021. https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/ (accessed May. 2, 2021).

[3] Andreas Spillner and Tilo Linz, Software Testing Foundations: A Study Guide for the Certified Tester Exam, 5th ed.

[4] J. Kwon, “What is Software Development Lifecycle (SDLC)?,” *Medium*, May 20, 2019. https://joycekwon.medium.com/what-is-software-development-lifecycle-sdlc-523fd09340a6 (accessed May. 2, 2021).

[5] Richard Rogers, “A simple comparison of Sequential and Iterative software development methods.” https://richrtesting.com/a-simple-comparison-of-sequential-and-iterative-software-development-methods/ (accessed May. 5, 2021).

[6] A. V. Patil, “Comparative Analysis between Sequential and Iterative Project Management Approaches,” vol. 06, no. 07, p. 6, 2019.

[7] M. Martin, “What is Waterfall Model in SDLC? Advantages and Disadvantages.” https://www.guru99.com/what-is-sdlc-or-waterfall-model.html (accessed May. 10, 2021).

[8]“ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΤΟ ΔΙΑΔΙΚΤΥΟ.” https://repository.kallipos.gr/bitstream/11419/3982/1/01_chapter_13.pdf(accessed May.10,2021)

[9] “SoftwareEngineering SDLC V-Model GeeksforGeeks.”

https://www.geeksforgeeks.org/ (accessed May. 10, 2021).

[10] “SDLC - V-Model.” https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm (accessed May. 10, 2021).

[11] B. W. Boehm, "A spiral model of software development and enhancement," in *Computer*, vol. 21, no. 5, pp. 61-72, May 1988, doi: 10.1109/2.59.

[12] “Spiral model,” *Wikipedia*. May. 12, 2021. Accessed: May. 12, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Spiral_model&oldid=1068244887

[13] “SDLC - Agile Model.” https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm (accessed May. 20, 2021).

[14] Gaurav Kumar, Pradeep Kumar Bhatia “Impact of Agile Methodology on Software Development Process,” International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 4, August 2012.

[15]“Make Your Startup Work With Agile SDLC Model,” *Product Tribe*. https://producttribe.com/project-management/agile-sdlc-guide (accessed May. 20, 2021).

[16] Kaner, Cem (November 17, 2006). Exploratory Testing (PDF). Quality Assurance Institute Worldwide Annual Software Testing Conference. Orlando, FL. (accessed May. 30, 2021).

[17]“ISTQB Foundation Level | Quality House Ltd.”

https://www.qualityhouse.com/index.php?page=certification-courses-foundation level(accessed May. 30, 2021).

[18]“Software Testing Principles - javatpoint,” *www.javatpoint.com*. https://www.javatpoint.com/software-testing-principles (accessed May. 30, 2021).

[19]R. Choubisa and P. Dalal, “Software Testing: A Review,” International Journal of Engineering Research, vol. 2, no. 11, p. 4, 2013.

[20]“Software Testing Life Cycle | Different stages of Software Testing,” *Edureka*, Feb. 08, 2019. https://www.edureka.co/blog/software-testing-life-cycle/ (accessed Jun. 2, 2021).

[21] Ιωάννης Τζανάκης, “Ανάπτυξη εφαρμογής δοκιμών αυτοματισμού, που προσαρμόζεται σε ηλεκτρονικό κατάστημα, με απώτερο σκοπό τη βελτιστοποίηση της ποιότητας του λογισμικού.,” Τμήμα Μηχανικών Πληροφορικής Τεχνικό Εκπαιδευτικό Ίδρυμα Κρήτης, Κοπεγχάγη, 2014. [Online].

Available: https://apothesis.lib.hmu.gr/bitstream/handle/20.500.12688/3229/TzanakisIoannis2014.pdf?sequence=1

[22] N. Anwar and S. Kar, “Review Paper on Various Software Testing Techniques & Strategies,” GJCST, pp. 43–49, May 2019, doi: 10.34257/GJCSTCVOL19IS2PG43.

[23] “Levels Of Testing | Software Testing Material,” *web-tech-hosting.net*, Jun. 01, 2021. https://web-tech-hosting.net/levels-of-testing-software-testing-material/ (accessed Jun. 5, 2021).

[24] “Differences between Functional and Non-functional Testing - GeeksforGeeks.” https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/(accessed Jun. 10, 2021).

[25] Γεωργία Μπαρνασά, “Πλαίσιο Ελέγχου Λογισμικού,” Πανεπιστήμιο Πειραιώς, Τμήμα Ψηφιακών Συστημάτων, Πειραιάς, 2012. [Online].

Available: https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/5023/Mparnasa.pdf?sequence=2&isAllowed=y

[26] “Testing types | 4 things every tester should know (Part 2/4),” *Lotus QA - Leading IT Outsourcing Company In Vietnam*, May 07, 2020. https://www.lotus-qa.com/testing-types-2/ (accessed Jun. 10, 2021).

[27] “Static Testing Vs Dynamic Testing.” https://www.tutorialspoint.com/static-testing-vs-dynamic-testing (accessed Jun. 15, 2021).

[28] R. E. Fairley, "Tutorial: Static Analysis and Dynamic Testing of Computer Software," in Computer, vol. 11, no. 4, pp. 14-23, April 1978, doi: 10.1109/C-M.1978.218132.

[29] “Software Testing | Static Testing - GeeksforGeeks.” https://www.geeksforgeeks.org/software-testing-static-testing/ (accessed Jun. 15, 2021).

[30] M. Ehmer and F. Khan, “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques,” IJACSA, vol. 3, no. 6, 2012, doi: 10.14569/IJACSA.2012.030603.

[31] “Dynamic Testing Techniques.” https://www.tutorialride.com/software-testing/dynamic-testing-techniques.htm (accessed Jun. 18, 2021).

[32] http://www.softwaretestinggenius.com/photos/wbtut1.JPG (accessed Jun. 18, 2021).

[33] ΜΑΘΙΟΥΔΑΚΗ ΕΛΕΝΗ, “Τεχνικές Αυτοματοποιημένου Ελέγχου Λογισμικού,” ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ, Αθήνα, 2015. [Online].

Available: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/40738/Eleni_Mathioudaki_diploma.pdf?sequence=1

[34] E. Enoiu, D. Sundmark, A. Causevic, and P. Pettersson, “A Comparative Study of Manual and Automated Testing for Industrial Control Software,” in 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), Tokyo, Japan, Mar. 2017, pp. 412–417. doi: 10.1109/ICST.2017.44.

[35] B. Kumari and N. Chauhan, “A COMPARISON BETWEEN MANUAL TESTING AND AUTOMATED TESTING,” vol. 5, no. 12, p. 9, 2018.

[36] “NUnit.org.” https://nunit.org/ (accessed Sep. 5, 2021).

[37] “TestNG - Welcome.” https://testng.org/doc/) (accessed Sep. 8, 2021).

[38] “Mockito framework site.” https://site.mockito.org/ (accessed Sep. 8, 2021).

[39] “Selenium,” *Selenium*. https://www.selenium.dev/ (accessed Oct. 5, 2021).

[40] “Robot Framework.” https://robotframework.org/ (accessed Oct. 5, 2021).

[41] “BDD Testing & Collaboration Tools for Teams | Cucumber.” https://cucumber.io/) (accessed Oct. 5, 2021).

[42] “Apache JMeter - Apache JMeter™.” https://jmeter.apache.org/ (accessed Oct. 20, 2021).

[43] “Cross Browser Testing, Selenium Testing, Mobile Testing,” Sauce Labs. https://saucelabs.com/) (accessed Nov. 10, 2021).

[44] ΓΙΩΡΓΟΣ Α. ΓΑΤΟΣ, “Εκπόνηση Σχεδίου Ελέγχου και Δοκιμών Διαδικτυακής Πλατφόρμας Ιατρικού Φακέλου,” Αθήνα, 2019.

[45] “Welcome to Python.org,” *Python.org*. https://www.python.org/ (accessed Sep. 1, 2021).

[46] “PyCharm: the Python IDE for Professional Developers by JetBrains.” https://www.jetbrains.com/pycharm/ (accessed Sep. 1, 2021).

[47] Bryan Lamb, Robot Framework Test Automation: Level 1 (Selenium).

[Online Video]. Available: https://www.linkedin.com/learning/robot-framework-test-automation-level-1-selenium/ (accessed Nov. 1, 2021).

[48] N. S. Batni and J. Shetty, “A Comprehensive Study on Automation using Robot Framework,” vol. 9, no. 7, p. 4, 2018.

[49] “Robot Framework Tutorial #13-Page Object Model(POM) in Robot Framework – RCV Academy,” Jul. 21, 2020. https://www.rcvacademy.com/robot-framework-tutorial-13-page-object-modelpom-in-robot-framework/ (accessed Nov. 20, 2021).