Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Διαδικτυακής Εφαρμογής στη μεριά του χρήστη, συμβατής με desktop, mobile και tablet συσκευών, για τη διαχείριση αθλητικών δεδομένων αθλητών στίβου, με χρήση της μεθοδολογίας Scrum Agile για την ανάπτυξη λογισμικού.** **Implementation of a cross-device client-side web application for managing and analyzing personal performances for track and field, using Scrum Agile software development methodology.**
Ονοματεπώνυμο Φοιτητή	**ΜΑΘΙΟΥΔΑΚΗΣ ΘΕΟΔΩΡΟΣ**
Πατρώνυμο	**ΧΡΗΣΤΟΣ**
Αριθμός Μητρώου	**ΜΠΣΠ 12039**
Επιβλέπων	**Ευθύμιος Αλέπης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Επίκουρος Καθηγητης	Μαρία Βίρβου Καθηγητρια	Κωνσταντίνος Πατσάκης Λέκτορας

ABSTRACT

Σε αυτή την πτυχιακή εργασία υλοποιήθηκε μια διαδικτιακή εφαρμογή, που σκοπό έχει να βοηθάει αθλητές του κλασσικού αθλητισμού (στίβος) να αποθηκεύουν και να αναλύουν τις επιδόσεις τους σε βάθος χρόνου. Η εφαρμογή θα προσφέρεται σε οποιαδήποτε συσκευή έχει πρόσβαση στο ίντερνετ και έναν περιηγητή ιστού (web browser). Θεωρήσαμε σημαντικό να καλυφθεί με σύγχρονο τρόπο μια βασική ανάγκη των συγκεκριμένων αθλητών και να δημιουργηθει και η βάση για μια πλατφόρμα που μελλοντικά θα μπορεί να υποστηρίζει επιπλέον λειτουργίες στον συγκεκριμένο χώρο. Επίσης θελήσαμε να ακολουθήσουμε τις πιο σύγχρονες μεθολογίες ανάπτυξης λογισμικού με σκοπό να δούμε στην πράξη που αυτές υπερτερούν, και που όχι, συγκριτικά με τις συμβατικές μεθοδολογίες που ακολουθούνται σήμερα στις μεγάλες και μικρές εταιρίες ανάπτυξης λογισμικού. Η εφαρμογή υλοποιήθηκε εξ'ολοκλήρου με τεχνολογίες open Web (HTML, CSS, Javascript, web APIs), δίνοντας μας την δυνατότητα να προσφέρουμε ουσιαστικά κάτι που τρέχει σε όλες τις σύγχρονες πλατφόρμες ανεξαρτήτου λειτουργικού συστήματος ή τύπου συσκευής. Ακολουθώντας την Scrum Agile Software Development Methodology είδαμε στην πράξη τα πλεονεκτήματα που εχει το Agile σαν προσέγγιση, συγκριτικά με τις συμβατικές μεθολογίες ανάπτυξης, αναφορικά με την ταχύτητα και την προσαρμοστικότητα, σε σημαντικές ή μη, αλλαγές στο περιβάλλον και στις ανάγκες της εφαρμογής ή των χρηστών.

In this thesis we implemented a web application, in order to help track and field athletes to keep track and analyse their performances through time. This application is offered to any device that has internet access, through a web browser. We believe that is important to cover this need for these specific athletes, in a modern technological way, and create the base for a platform which can support additional functionality in the future, helping the specific sports area to change and innovate. We, also, wanted to see in practice how latest software development methodologies are applied, in comparison with the older ones, which are widely used.

The application has been implemented with open Web technologies(HTML, CSS, Javascript, web APIs), giving us the possibility to offer a service that runs on all modern platforms, independent of Operating System or type of device. Following Scrum Agile Software development methodology we saw the advantages of such a methodology, compared to older legacy methods, in speed and adaption to big or small changes, in general environment and requirements of application or users.

1. INTRODUCTION

1.1 Purpose of the thesis

In this thesis I will show the agile methodology of software development as it is applied on the development process of a cross-device web application. I will focus on the iterations between defining the specifications of the application and its development. These phases will be called “Sprints” of the application, and will have as a final result a fully functional version of the application according to the agile methodology. The flexibility and the efficiency of the agile methodology will be shown in multiple cases by changing the specifications in different stages during the development process and by adding completely new features in phases which were not planned in the previous steps. The selected software engineering method will prove its flexibility as these changes will have little effect on the speed of the development and the quality of the final outcome of the current iteration.

Since it’s a thesis about software engineering, there will be a wide range of diagrams, showing the different states of the application in each phase, the decisions taken before each phase and how the application was designed before it was developed. Also there will be a lot of User Interfaces Designs describing the whole process and evolution of the application.

Since a big part of the philosophy behind the agile methodology is the efficient cooperation of the team which is developing the application, the back-end development has be done by my colleague Georgios Balasis, whose thesis is referenced in this document and who designs the code that will run in the server side, complementing the parts of the front-end that I will design and develop. In each phase, it will be clear how the communication between the browser and the server takes place, and which are the API endpoints that make this communication happen, essentially showing the cooperation of the team in accordance to the agile guidelines.

The parts of the application that will be designed and developed by me will be the User Interface design, the User Testing and all the code required for the logic on web-client in order to connect with back end and serve the information to the end user. All these parts will be described as parts of the software implementation in each iteration of Agile methodology.

Finally, after the end of the analysis of the development process of the application, there will be a review of the steps followed and the strong and weak points of the agile methodology will be pointed out, for each of these steps.

1.2 Purpose of the application

The main goal of the application is to keep various data of the performances of athletes, and display them in a way which can help them get useful deductions that will eventually help them improve. It also helps athletes keep an archive of all their performances so that they can track their progress.

Our users need to enter raw data in a form, related to the performance. That data is saved in a database and then it can be displayed as distinct activities which can be filtered or as graphs with various functionalities.

The application also has a variation of settings, regarding the users profiles and the accounts. Users can search for other users, using a search input field, visit their profiles and view their performances, if specific permissions allow that. Also, the API of the application allows it to be integrated in other devices like mobile phones or, later, accessories.

1.3 Structure of this document

This introduction is followed by 7 sections. On the first one I will analyse the decisions we had to take, about technologies and methodology we wanted to follow. On the second one I’ll describe the “Scrum Agile software development methodology”, and how it is compared with older and most common software development methodologies. After that, on the fourth section I will describe the technologies, the frameworks and the libraries we used and the overall architecture of the application. On the fifth section I will describe the requirements analysis and then on the sixth I will analyse in depth all the software development phases, using the Scrum agile methodology. Finally on the seventh section I make a conclusion and describe some future work that can be made.

2. DECISIONS

2.1 General decisions:

We decided to use technologies for the development of the application that are all web-based. The server is node.js, a web server running JavaScript on the V8 engine that Google uses in its Chrome web browser. On top of node.js we implemented the MVC so that the models and the interaction of the database is clearly separated from the controllers and the logic of the application. The framework used for this purpose on node.js is called Express. Express takes care of the routes of the application, the sessions, the controllers and binds all the logic of the back-end system.

For the model part of the MVC, we choose to use a module named Mongoose, which forms the schemas of the databases and takes care of all the queries. Q module is handling the asynchronous database calls. The database used is mongoDB which is a NoSQL database that has features that fit, conveniently, into the application.

The HTML that will be displayed in the browser is generated by the JADE template engine from backend and AngularJS in front-end. Cascade Style Sheets (CSS) are generated using the LESS framework, which helps us to keep a more structured code for this kind of files. On the web client, the logic is implemented with the AngularJS JavaScript framework, which helps us to clearly separate the view part, from the logic one and the connection to the backend. For the responsive design, the Bootstrap CSS framework is used in parallel with our custom css files.

These are the technologies we finally choose to use in our application. At initial point we started with a different stack. We started with PHP as our choice for server side logic and common HTML, javascript, CSS in web-client side. This stack change in two different pivot points in our process, as will explained in detailed in our sprint analysis in later chapter.

2.2 Why we choose Agile

The agile methodology was chosen because of its flexibility. It’s obvious that most of our decisions, were affected by the nature of the application.The application is developed using the latest web technologies which by the time seemed suitable for the application but we, as developers, were not entirely familiar with them. In addition, the functionality and the features of the application is driven by athletes who were asked to test it. Flexibility regarding the changes of the specifications, the agile methodology is perfect for this case.

Since the application is functional between the iterations, athletes can have the demos and test if the features suit their needs and expectations. This allows us to adapt in changes and be fast in case we need to change completely, or partially, our direction. This happened twice in the whole process, where we need to change the implementation in web-client side where we replaced standard html-javascript with AngularJS framework, and in server side, replacing PHP with node.js.

2.3 Why only JavaScript

In our application everything is JavaScript. Even the database. Being able to read and understand your colleague's code is very important. This allows the better collaboration between all developers (front and back end) and even semi-technical designers.

The application is written in JavaScript, both on the front-end and on the back-end. There aren’t many choices for writing the front-end of the application, so JavaScript was an easy choice, but there were many choices for the language of the back-end. Having JavaScript both on the front-end and on the back-end makes the communication of the browser and the server seamless. JSON objects are supported by default on both ends, so this is the format used for the communication between them, instead of XML. Also, when the logic of the front-end and the back-end is similar, code can be easily reused, following the don't repeat yourself (DRY) principle.

2.4 Why MongoDB

JSON objects are similar to the BSON objects supported by MongoDB, so the next step of communicating without converting any data is using MongoDB. NoSQL databases have a good performance advantage in certain cases. The application doesn't have data that require a relational database (table merge scenarios), so in this case MongoDB is the perfect database and it can scale really well, since it has been implement with scalability in mind.

2.5 Why we choose a web version over a native one (or more)

There is just a web version of the application. That is because, one of the requirements was to be platform independent . Not only it should be accessible on desktops running different operating systems (iOS, Android, etc), but it should also be on tablets and mobile phones. Make our application web based has the advantage that it can be used from every device that has a web browser. This means that with this kind of implementation we can have direct access to 100% of the market share, without needed to write code for different platforms and environments.

For example, if we wanted to support natively iOS and Android mobile device, which currently have more than 92% of market share combined, we would needed to build a server and a database system in the way we build it now, and also an application in Java for Android and one in Objective-C for iOS. Of course for supporting natively these platforms we should test more than 100 devices and more than 10 different versions of these operating systems. Even with this scenario there would be no support for desktop version. Today web usage is shared as desktop: 41%, mobile: 52%, tablets: 7% [https://mixpanel.com/trends/#report/desktop_vs_mobile_vs_tablet]. So not supporting desktop computers would be a great miss for our users.

2.6 Why mobile support

As mentioned before mobile devices (cellular phones and tablets combined) are causing about 50% of web usage today. This makes these devices' support crucial for the usability and publicity of our application.

Also the application is about recording the performances of athletes. Track and field athletes travel all the time, around the world joining competitions in different countries. This means, of course, that there is not always a computer nearby. And of course there is not a computer in the stadium where the athletes compete. So we think that it's crucial to give them the opportunity to add their performance on the fly, from everywhere they are, with their mobile.

2.7 Why clear separation of back-end and front-end

Another decision made is the clear separation between the front-end and the back-end. The application is designed with third-party interfaces and devices in mind. In order to give the opportunity to others to create their own interface and access the API of the application, the back-end should be completely independent from front-end. This means that now we can create any native application we want, just by interact with the existing API we use in our current app.

3. AGILE SOFTWARE DEVELOPMENT METHODOLOGY

3.1 WHAT IS AGILE AND HOW IS COMPARED WITH WATERFALL

In 1970, Dr. Winston Royce presented a paper entitled “Managing the Development of Large Software Systems,” which criticised sequential development. He asserted that software should not be developed like an automobile on an assembly line, in which each piece is added in sequential phases, each phase depending on the previous. Dr. Royce recommended against the phase based approach in which developers first gather all of a project's requirements, then complete all of its architecture and design, then write all of the code, and so on. Royce specifically objected to the lack of communication between the specialised groups that complete each phase of work.

It’s easy to see the problems with the waterfall method. It assumes that every requirement can be identified before any design or coding occurs. Could you tell a team of developers everything that needed to be in a software product before any of it was up and running? Or would it be easier to describe your vision to the team if you could react to functional software? Many software developers have learned the answer to that question the hard way: At the end of a project, a team might have built the software it was asked to build, but, in the time it took to create, business realities have changed so dramatically that the product is irrelevant. The company has spent time and money to create software that no one wants. Couldn’t it have been possible to ensure the end product would still be relevant before it was actually finished

There are many challenges that waterfall has to face. We can visualise the waterfall’s software development phases in the following diagram.

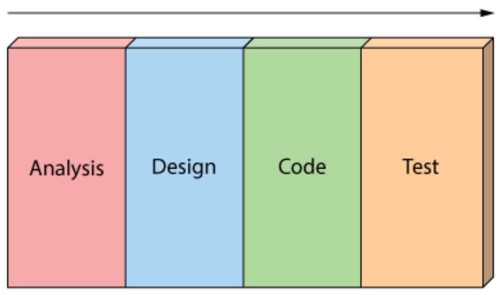


image 0.1 : Waterfall’s software development phases

Traditional Waterfall treats analysis, design, coding, and testing as discrete phases in a software project. This worked fine when the cost of change was high. But now that it's low it hurts us in a couple of ways. First off, when the project starts to run out of time and money, testing is the only phase left. This means good projects are forced to cut testing short and quality suffers. Secondly, because working software isn't produced until the end of the project, you never really know where you are on a Waterfall project. That last 20% of the project always seems to take 80% of the time. When design, the second of four phases, is completed it doesn’t mean you are half way! Thirdly you've got schedule risk because you never know if you are going to make it until the end. You've got technical risk because you don't actually get to test your design or architecture until late in the project. And you've got product risk because don't even know if you are building the right until it's too late to make any changes. Finally, most importantly, it's just not a great way for handling change.

The Agile movement proposes alternatives to traditional project management. Agile approaches are typically used in software development to help businesses respond to unpredictability. Agile development provides opportunities to assess the direction throughout the development lifecycle. This is achieved through regular cadences of work, known as Sprints or iterations, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development — requirements, design, etc. — is continually revisited. When a team stops and re-evaluates the direction of a project every two weeks, there’s time to steer it in another direction.

This “inspect-and-adapt” approach to development reduces development costs and time to market. Because teams can develop software at the same time they’re gathering requirements, “analysis paralysis” is less likely to impede a team from making progress. And because a team’s work cycle is limited to two weeks, stakeholders have recurring opportunities to calibrate releases for success in the real world. Agile development helps companies build the right product. Instead of committing to market a piece of software that hasn’t been written yet, agile empowers teams to continuously re-plan their release to optimise its value throughout development, allowing them to be as competitive as possible in the marketplace. Agile development preserves a product’s critical market relevance and ensures a team’s work doesn’t wind up on a shelf, never released.

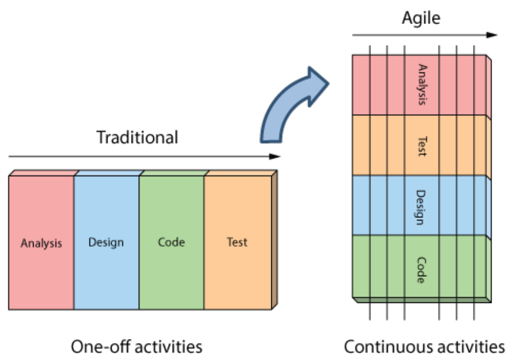


image 0.2 : Visualise the difference of one-off activities of waterfall, to continuous activities of Agile.

3.2 SCRUM FRAMEWORK

The Agile Manifesto doesn’t provide concrete steps. Organisations usually seek more specific methods within the Agile movement. These include Crystal Clear, Extreme Programming, Feature Driven Development, Dynamic Systems Development Method (DSDM), Scrum, and others. Scrum’s simple definitions gave our team the autonomy we needed to do our best work while helping us get the business results he wanted.

Scrum is the most popular way of introducing Agility due to its simplicity and flexibility. Scrum emphasises empirical feedback, team self management, and striving to build properly tested product increments within short iterations. Doing Scrum as it’s actually defined usually comes into conflict with existing habits at established non-Agile organisations.

Scrum has only three roles: Product Owner, Team, and Scrum Master. The responsibilities of the traditional project manager role are split up among these three Scrum roles. Scrum has five meetings: Backlog Grooming (aka Backlog Refinement), Sprint Planning, Daily Scrum (aka 15-minute standup), the Sprint Review Meeting, and the Sprint Retrospective Meeting.

Scrum uses fixed-length iterations, called Sprints, which are typically 1-2 weeks long and never more than 30 days. Scrum teams attempt to build a potentially shippable (properly tested) product increment every iteration.

3.3 COMPARISON BETWEEN AGILE AND RUP

As noticed before there are a lot of methodologies that are Agile. One of them is RUP which is commonly used in big corporation and is widely recognised from software engineers. Thus we found meaningful to notice the differences between SCRUM and RUP. Both methodologies are considered to be Agile and approach project activities in the iterative way. However, RUP methodology calls for a formal definition of scope and major project milestones are associated with specific dates. SCRUM methodology uses project backlog instead of scope and allows the backlog to be redefined at the end of each iteration (usually about every 4 weeks).

In addition, RUP subdivides the project lifecycle into 4 major phases (Inception, Elaboration, Construction, Transition). Even though it encourages concurrent workflows across the entire cycle, the general understanding is that certain activities will peak during certain phases (for instance, requirements analysis will spike during the elaboration phase). On the contrary, SCRUM dictates that the entire “traditional” lifecycle fits into one iteration. In other words, a workload for one iteration at a time is determined and then the entire cycle occurs within one iteration (e.g. the requirements for a particular feature are collected, documented as a user story, then coded, tested and presented for the user review).

	RUP	SCRUM
Approach	Iterative	Iterative
Cycle	Formal Cycle is defined across 4 phases, but some workflows can be concurrent.	Each sprint (iteration) is a complete cycle.
Planning	Formal project plan, associated with multiple iterations, is used. The plan is end-date driven and also has intermediate milestones.	No end-to-end project plan. Each next iteration plan is determined at the end of the current iteration (NOT end-date driven). Product Owner (Key Business User) determines when the project is done. In general 'Definition of Done' defines when a task is complete.
Scope	Scope is predefined ahead of the project start and documented in the Scope document. Scope can be revised during the project, as requirements are being clarified, but these revisions are subject to a strictly controlled procedure.	Instead of scope, SCRUM uses a Project Backlog, which is re-evaluated at the end of each iteration (sprint).
Artifacts	Vision/Scope Document, Formal functional requirements package, system architecture document, development plan, test plan, test scripts, etc.	The only formal artifact is the operational software.
Type of Project	Recommended for large, long- term, enterprise-level projects with medium-to-high complexity.	Recommended for quick enhancements and organizations that are not dependent on a deadline.

image 0.3 : Comparison between Scrum and RUP Agile process.

3.4 USABILITY EVALUATION METHODS

Before starting implementing functionality and design our application we had to take a fast feedback from real users of how our application should look. We need to answer questions like what design is more familiar and which functionality is important for an MVP(minimum value product).

As interfaces become ever more complex and development schedules seem to get shorter and shorter, is very useful to give up the user-interface modelling software for awhile in favour of something simpler. So in order to achieve our target, we used paper prototypes for many different User Interface concepts. All we were needed was paper, pens and scissors. Users run through a set of existing paper mock-ups or are given blank paper and asked to represent a concept by sketching it. The prototyping stage is the right time to catch design flaws and change directions, and the flexibility and disposability of paper encourages experimentation and speedy iteration. Instead of “deleting” hours worth of layout code you’ve used to position a column in the right place, you can draw a prototype, throw away the ideas that don’t work, and move on.

A paper usability-testing session works much like any other usability-testing session. Begin by selecting a range of testers who represent your expected audience. Have scenarios ready for the user to perform. Document the testing sessions with video to review the users’ emotional state when using your mocked-up interface. Debrief users afterward to measure interface recall. With paper, you can also allow users to mock up ideas they think would solve a problem, mark on the prototype where a user attempted to “click” or otherwise interact with the interface, ask users to draw what they expect to happen next and keep going even if you don’t have access to a testing lab or if computers, networks, or high-tech prototypes don’t work as expected.

Most of the prototypes will be shown in chapter 6, in which we analyse the whole software development process.

4. TECHNOLOGIES AND ARCHITECTURE

4.1 JavaScript

JavaScript® (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, most known as the scripting language for Web pages, but used in many non-browser environments as well such as node.js or Apache CouchDB. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.The JavaScript standard is ECMAScript. As of 2012, all modern browsers fully support ECMAScript 5.1. A 6th major revision of the standard is in the works and is expected to be finished around mid 2015. JavaScript can function as both a procedural and an object oriented language. JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation, object introspection, and source code recovery (JavaScript programs can decompile function bodies back into their source text). Intrinsic objects are Number, String, Boolean, Date, RegExp, and Math.

4.2 AngularJS

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets us extend HTML vocabulary for our application. The resulting environment is extraordinarily expressive, readable, and quick to develop. Other frameworks deal with HTML’s shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views.

AngularJS is a toolset for building the framework most suited to our application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit our unique development workflow and feature needs.Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is very important because it eliminates DOM manipulation from the list of things we have to worry about. Controllers are the behaviour behind the DOM elements. AngularJS lets us express the behaviour in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes.

Unlike other frameworks, there is no need to inherit from proprietary types in order to wrap the model in accessors methods. Angular models are plain old JavaScript objects. This makes our code easy to test, maintain, reuse, and again free from boilerplate.Client-side form validation is an important part of great user experience. AngularJS lets us declare the validation rules of the form without having to write JavaScript code. AngularJS provides built-in services on top of XHR as well as various other backends using third party libraries. Promises further simplify your code by handling asynchronous return of data.

4.3 Node.js

Node.js is event based and asynchronous/non-blocking. Events, like an incoming HTTP connection will fire off a JavaScript function that does a little bit of work and kicks off other asynchronous tasks like connecting to a database or pulling content from another server. Once these tasks have been kicked off, the event function finishes and Node.js goes back to sleep. As soon as something else happens, like the database connection being established or the external server responding with content, the callback functions fire, and more JavaScript code executes, potentially kicking off even more asynchronous tasks (like a database query). In this way, Node.js will happily interleave activities for multiple parallel workflows, running whatever activities are unblocked at any point in time. This is why Node.js does such a great job managing thousands of simultaneous connections.

Node.js is single threaded and lock free. Node.js, as a very deliberate design choice only has a single thread per process. Because of this, it's fundamentally impossible for multiple threads to access data simultaneously. Thus, no locks are needed. Eliminating locks and multi-threading makes one of the nastiest classes of bugs just go away. This might be the single biggest advantage of node.

Some advantages are that web development in a dynamic language (JavaScript) on a VM that is incredibly fast (V8). It is much faster than Ruby, Python, or Perl. The ability to handle thousands of concurrent connections with minimal overhead on a single process. The fact that JavaScript is perfect for event loops with first class function objects and closures. Using JavaScript on a web server as well as the browser reduces the impedance mismatch between the two programming environments which can communicate data structures via JSON that work the same on both sides of the equation. Duplicate form validation code can be shared between server and client, etc.

4.4 Express and Connect

Node.js itself offers an http module, whose createServer method returns an object that you can use to respond to HTTP requests. That object inherits the http. Server prototype. Connect also offers a createServer method, which returns an object that inherits an extended version of http.Server. Connect's extensions are mainly there to make it easy to plug in middleware. That's why Connect describes itself as a "middleware framework". Express does to Connect what Connect does to the http module: It offers a createServer method that extends Connect's Server prototype. So all of the functionality of Connect is there, plus view rendering and a handy DSL for describing routes.

4.6 MongoDB

MongoDB is an open-source, document-oriented database designed for ease of development and scaling. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents. Some of the advantages of using documents are that documents (i.e. objects) correspond to native data types in many programming languages. Embedded documents and arrays reduce need for expensive joins. Dynamic schema supports fluent polymorphism.

Some key features are the high performance(embedded data models reduces I/O, Indexes support faster queries and can include keys from embedded documents and arrays), the high availability(automatic failover, data redundancy) and the automatic scaling(automatic sharding distributes data across a cluster of machines, replica sets can provide eventually-consistent reads for low-latency high throughput deployments)

4.7 LESS (CSS, transitions)

Less is a CSS pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions, operations and many other techniques that allow us to make CSS more maintainable, them-able and extendable. Variables are actually constants that can be defined once and used across the whole implementation. Mixins are a way of including ("mixing in") a bunch of properties from one rule-set into another rule-set. This is very helpful and convenient when we need to support attributes that are not well defined and have different syntax between the famous browsers we support.

4.8 Jade

Jade is a high performance template engine heavily influenced by Haml and implemented with JavaScript for node and browsers. Jade is a terse language for writing HTML templates. It produces HTML, supports dynamic code and also reusability (DRY).

4.9 Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.Bootstrap makes front-end web development faster and easier. It's made for devices of all shapes, and projects of all sizes. Bootstrap easily and efficiently scales our websites and applications with a single code base, from phones to tablets to desktops with CSS media queries. We used bootstrap in parallel with our code, for User Interface elements in LESS (CSS) and logic in AngularJS.

4.10 Charts Libraries (Google Charts and C3/D3)

When we started the project we choose Google Charts as a good option that is provided and supported from Google. It provides a good way to visualise data on your website. From simple line charts to complex hierarchical tree maps, the chart gallery provides a large number of ready-to-use chart types. The most common way to use Google Charts is with simple JavaScript that you embed in your web page. This is what we also did in our application at first stages and before we choose AngularJS as the framework on the web-client side. Charts are highly interactive and expose events that let you connect them to create complex dashboards or other experiences integrated with your webpage. Charts are rendered using HTML5/SVG technology to provide cross-browser compatibility (including VML for older IE versions) and cross platform portability to iPhones, iPads and Android.

After the addition of AngularJS at front end we decided to make a transition to D3.js. D3.js (or just D3 for Data-Driven Documents) is a JavaScript library that uses digital data to drive the creation and control of dynamic and interactive graphical forms which run in web browsers. It is a tool for data visualisation in W3C-compliant computing, making use of the widely implemented Scalable Vector Graphics (SVG), JavaScript, HTML5, and Cascading Style Sheets (CSS3) standards.

C3 is a library on top of D3 which makes it easy to generate D3-based charts by wrapping the code required to construct the entire chart. We don't need to write D3 code any more. It gives some classes to each element when generating, so you can define a custom style by the class and it's possible to extend the structure directly by D3. C3 provides a variety of APIs and callbacks to access the state of the chart.

4.11 System Architecture

Our system architecture changed rapidly through the development process due to new decisions and requirements that raised. On our initial implementation we started with a software stack that had three completely different environments from end-to-end. It had plain JavaScript, HTML and pure CSS in the web client side. PHP on the server and MongoDB as a storage.

As described in more details inside the whole process analysis, this stack had some limitations, mainly in scalability and maintenance. The following architecture diagram shows how the layers of our application are positions one over the other

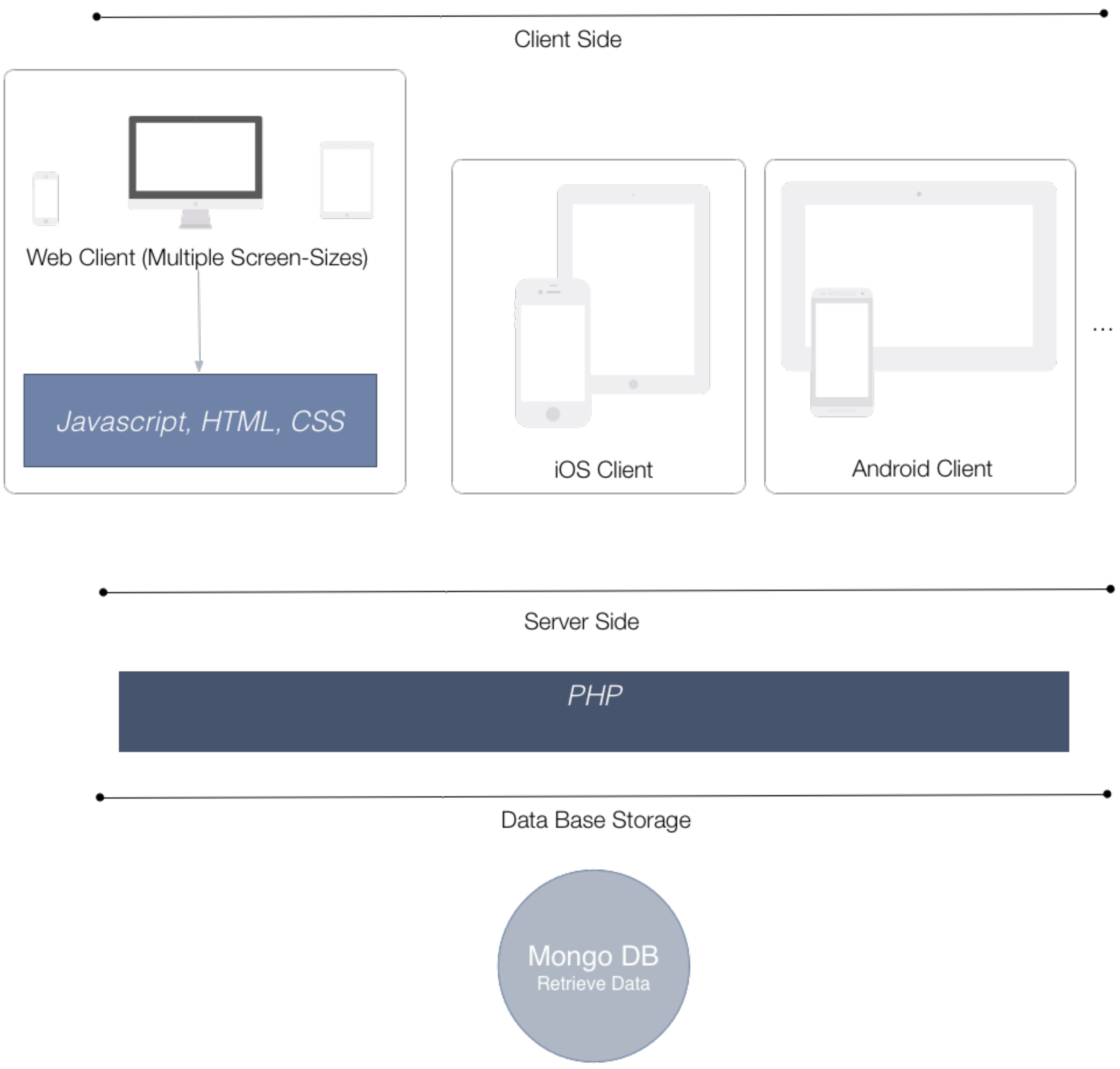


image 0.3 : Initial Software Architecture Diagram, with MongoDB-PHP-JavaScript-HTML-CSS

After implementing almost all the basic functionality we decided to move to a more scalable solution, which will help us use only one programming language avoiding in this way the object transformation and mapping between components. For these reasons we replaced PHP with Node.js on server. So after this change the new architecture diagram looks like the following.

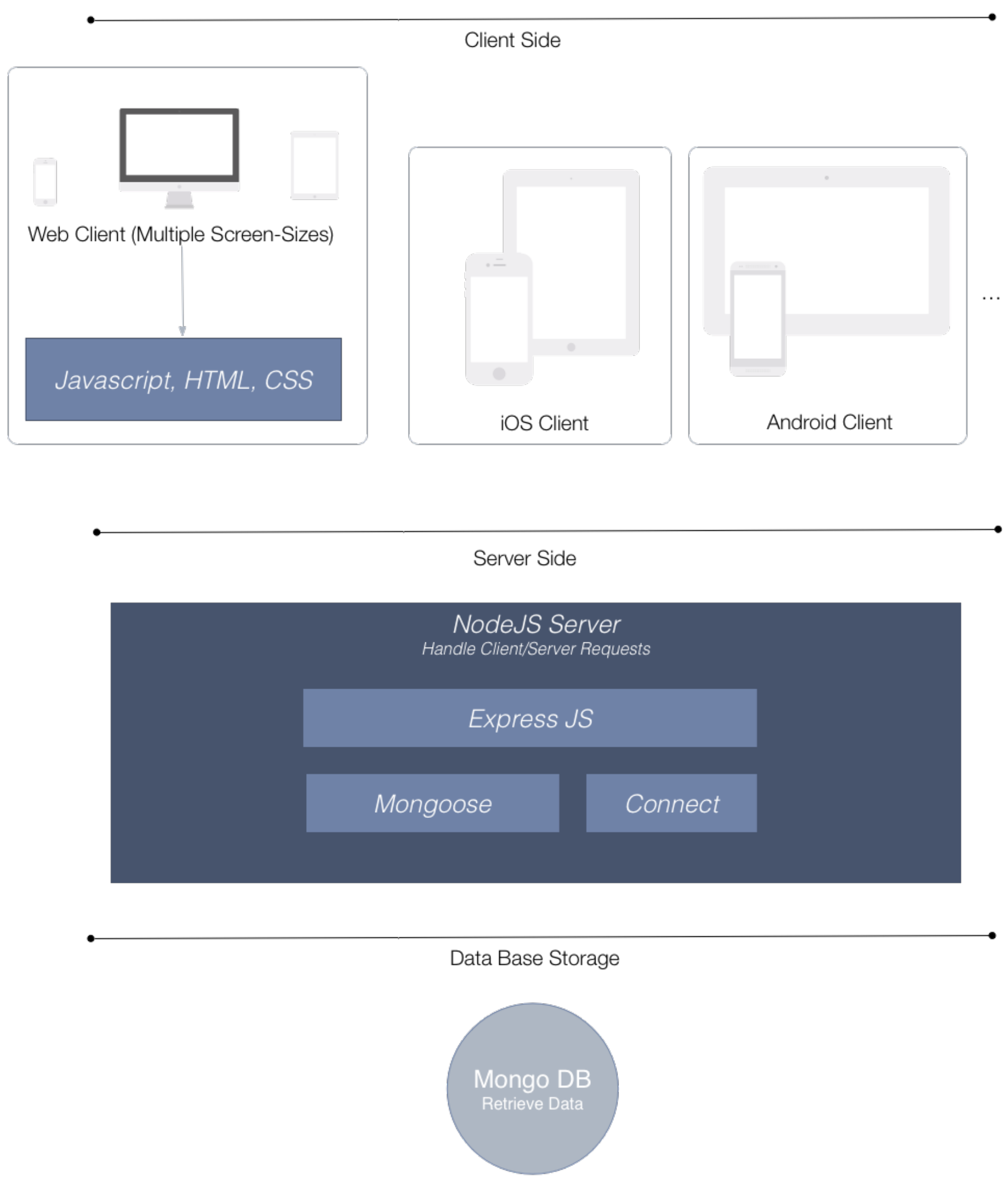


image 0.4 : Software Architecture Diagram, with MongoDB - Node.js - JavaScript - CSS

After the successful transformation of server-side from PHP to Node.js, we made one more pivot. This time on web client. The main idea of our final stack is that server will implements a pure REST API that will be interact with multiple clients (web or natives). This means that we have to implement all the business logic on client side, which is something obvious for native applications but not for web applications, till now.

Introducing AngularJS on web client side allows us to build the business logic there and use pure HTTP and AJAX request for communication between web client and server. We also replace pure CSS with LESS library in order to keep our styling assets and code more structured and make it easier to maintain them. So the final architecture is as seen on the following diagram.

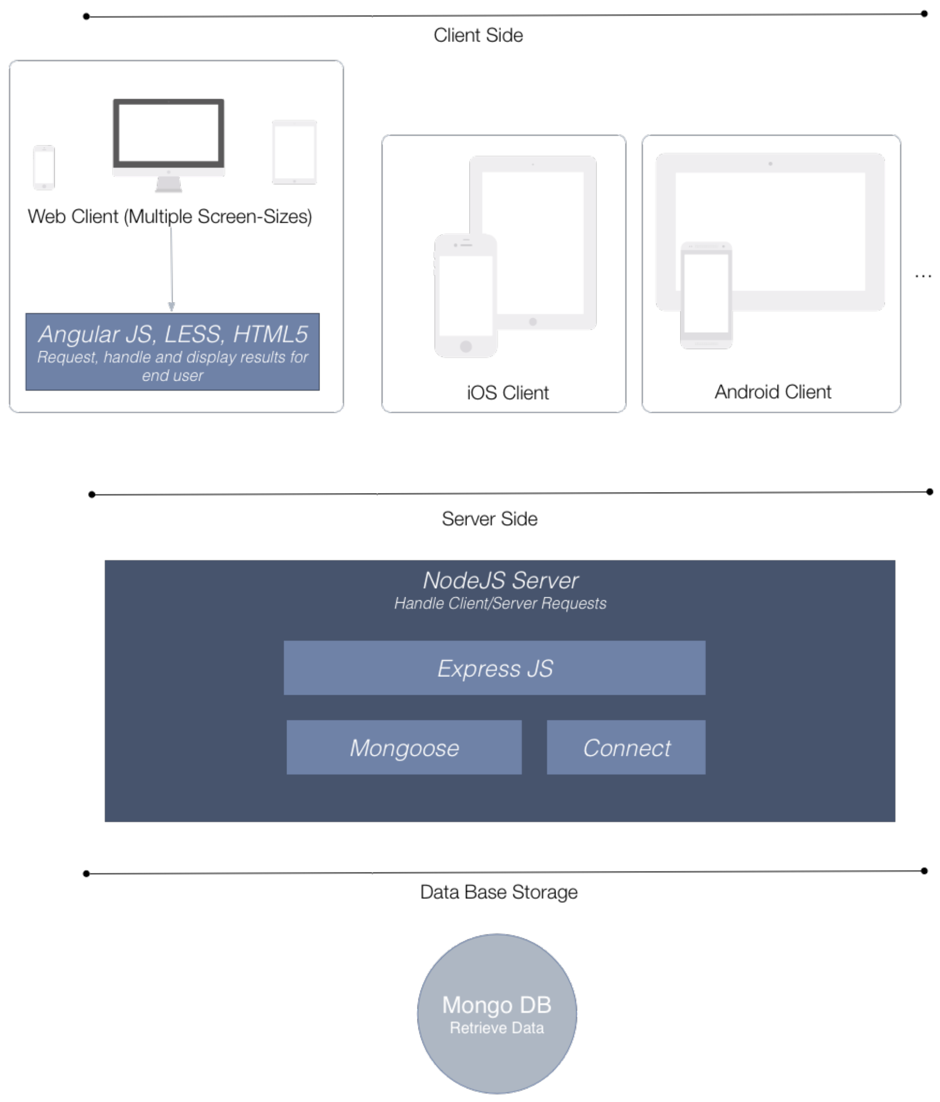


image 0.5 : Software Architecture Diagram, with MongoDB - Node.js - AngularJS - LESS

5. REQUIREMENTS ANALYSIS

5.1 Functionality

Before designing the application, the requirements of the sections and the overall functionality is defined. The final version of the application should be a fully functional tool for the athletes, with all the settings and functionalities that will make the process of tracking their progress easy and satisfying.

5.1.1. Login

Since our application will be accessed by individual users, there’s a need for a login section. Because the email address is needed in many cases by the application, the login credentials are an email address and a password, avoiding unnecessary values like usernames, names etc. The login section should create a session for the user and keep them logged in, until they decide to logout. It should also be simplistic and look similar to the registration section. There should also be a link to the “Forgot password” section.

5.1.2. Register

There should be a page where the users are creating their accounts. The registration section should be simple, and similar with login section and after a validated registration phase, records of the new users should be added to the database. The required fields of the registration section should be five: a first name, a last name, an email, a password and the same password repeated for safety reasons. The first name and the last name are required because they should be displayed in the user’s profile which makes the user easier to find and be recognised by other users. They also make the application feel more personal, which may work as a self inspiration for their progress. The password is used for the login process and it should repeated in order to avoid typographical errors. The email is used for the registration process and for email notifications.

5.1.3. Email validation

The email address that the user submitted during the registration process should be validated. Right after they have registered, an email should be sent to the given address, providing a unique link, created specifically for this user. If they click on this link, the application can assume that it was clicked from the existing email address of the user, that they have access to that address, and therefore it will complete the registration process, allowing the user to use the application. After the validation of the email address, the user should be immediately logged in, since there is no reason to ask for the password when the link is clicked in the user’s own email client.

5.1.4. Forgot password

Following the “Forgot password” link should lead the users to a page where they can enter their email address, making a request for a new password. After submitting the form, an email should be sent to the given email address and that email should include a unique link to a page where the user can enter a new password and repeat it once for safety reasons. After submitting this form, the password of the user should have changed to the new one and the user should be logged in without using the login form.

5.1.5. Automatic detection of user’s preferences

When the user logs in for the first time (after their email address has been validated), some preferences should be preselected. The country of the user should be detected and, using this information, the “country” setting and the “unit system” setting, should be preselected by the name of the country of the user and the unit system that is used in this country respectively, and they should be saved as the default settings of their account. Also, the profile will be private by default, meaning that other users won’t be able to access it at all. This will be preselected for all users after registration.

When these settings get set automatically, a notification will be showing the values of these settings to the user, stating that if these settings are not correct or if the user wants to change them, they can visit the settings page and do that manually, and a link of that page should be provided in the notification.

5.1.6. Profile

The profile has to be the main section of our web application. The registration process and the login process should take the user here. In this section, the data of the user should be visible and the activities can be shown, edited, created, deleted, and they can also be filtered.

The profile section should display the user’s profile picture, the first and last name, the discipline and the country. There should also be a list of all the activities previously submitted, which should be editable and deletable, and a form to submit new activities. There should also be a way to filter the activities shown in the list.

5.1.7 Edit activity

Each activity in the list, in the profile page, should have a way to make the activity editable. The activity should turn into a form, similar to the one that is used to submit new activities, and the data of the activity should be in the appropriate fields. The user should be able to change the values and submit the form. After the form is submitted, the activity should change to its initial state, not editable anymore, with the new data.

5.1.8 Add activity

In the profile page, it should be possible to create new activities. There should be a form where a user can select their discipline, and according to their selection, several more fields should appear. In case the user has chosen a main discipline in the settings, that discipline should be preselected.

There should be one or multiple fields that accept the user’s performance. Other fields needed are the date of the activity, a field showing whether the activity is training or not, a field showing the location it took place, a text area for notes, and if the activity is a competition, there should also be fields of the name of the competition and the place that the athlete came.

Since the form is big for the page, and not always needed, the fields should be hidden, when the page is accessed, and the should be shown when the user clicks on the “add activity” option. When the user has submitted the form, the form should close again and the new activity should be added to the list of the activities, in the correct place, according to the date of the activity.

5.1.9 Delete activity

In a similar way that an activity can be turned to editable, an activity should be deleted. When the user chooses to delete an activity, a confirmation message should be shown, and if the user confirms the deletion, the activity should be removed from the list. Otherwise, the message should disappear and the list should remain intact.

5.1.10 Activity filtering

It should be possible to filter the activities in the profile. There should be a way to display all the activities or the activities of only one discipline.

5.1.11 User search

When a user is logged in, on every page of the application, there should be a field that can be used in order to search for others who use the application. The search should require a first name or a last name or both, the server should search for the users on every user’s key press, and the top five results should be displayed right below the search area. Clicking on one of the results should lead the user to the profile of the selected user.

5.1.12 Visiting profiles

Users should be able to visit the profiles of other users, when they are not private. The profile should have the same interface as when its owner visits it, but there should not be any options to add, edit or delete activities. The whole form used to add activities should be missing, and the options to remove or edit individual activities should not appear on them as well. There should still be a way to filter the activities. Also, in the same way, the statistics of the user should be available for visiting.

5.1.13 Statistics

The data entered in the profile should be displayed as graphs in the “statistics” section. The graphs should have the performance on the y axis and the dates on the x axis. There should be a way to change the discipline shown in the graph and there should be a way to focus on a certain timespan.

5.1.14 Profile settings

There should be a section where the settings and the information that will appear in the profile can be edited. The section of the profile settings, should have as editable fields the user’s first name, last name, main discipline, gender, birthday, country, profile picture, privacy and a short text about themselves.

Changing the first name, last name, profile picture, country or discipline, should change the relevant information in the profile page. Also, changing the first and last name should change the way that particular user can be found using the search functionality. The old first and last name shouldn’t be used as search parameters anymore.

Changing the privacy of the profile to public should make the profile visible to others, while changing it to private should remove it from the search results and visiting that profile by its url should show a page stating that this profile does not exist.

5.1.15 Account settings

In the same page as the profile settings, there should be a section with editable fields that affect values related to the user’s account. These are the password, the language, the date format, the system of measurement and the url of the user’s profile.

When the user selects to change the password, three fields should appear, one that will require the old password, and two that will require the new one for safety reasons. Entering the correct old password and the exact same password twice as a new password, should immediately take effect.

Changing the language setting should change the language used for the whole interface of the application. The value preselected when the user first accesses the application, should be the default language of the user’s country. The files of the translated values should be separated from the rest of the application, so that changes applied to them will affect all occurrences of the value throughout the application.

The date format input should have two options: day-month-year and month-day-year. Changing this setting should change they way the dates are shown throughout the application, even when visiting profiles of other users, regardless of their settings.

There should be two systems of measurement, the metric and the imperial. A user should have one of them preselected after registration, based on the country they live in. All the units should be displayed in the selected system, even the units shown in other profiles.

The initial url of a user’s profile should be the user’s id in the database. In the account settings, can be set a new url for the profile, one that a user can choose. Changing the url shouldn’t make the profile inaccessible by the user’s id.

5.1.16 Privacy

Privacy should be divided into two sections. Profile privacy and activity privacy. Setting a profile to private, makes the profile visible only to its owner. It should not appear in the search results and when it will be visited by its URL, it should display a message stating that the profile was not found. Accessing an activity of a private profile by using the API should fetch no data.

Activity privacy should be applied when a profile is public. Each of the activities should be either public or private. The public activities should be visible by anyone who can view that user's profile, including the users who are not currently logged in. Private activities should be visible only to its owner, and the other users shouldn’t be able to see them on the profile page or access them using the API.

5.1.17 API

Our API is RESTful and the application itself should use it. It should mainly interact with the user and activity entities. The GET method should be used for getting data, POST for creating new data, PUT for updating data and DELETE for deleting data.

5.1.18 Responsive design

The application should be displayed correctly on desktop computers, mobiles and tablets, both in portrait and landscape mode. The controls should be easily accessible for both platforms and the interface should be change in order to adapt to each screen, and possible hide or collapse features in smaller screens.

5.2 Terms definition

5.2.1 Activities

I will refer to the performances of the athletes at competitions as activities. They will have several parameters such as the performance, the discipline, the date, the location of the event, the name of the event, the place the athlete got in the event, notes, and whether it is a private activity or not.

5.2.2 Statistics

The diagrams that show statistics of the athletes' performances through time and the respective section of the application will be referred to as "statistics".

5.2.3 Discipline

A discipline is the sport that an athlete does. An athlete, as user of the application, has a main discipline, the discipline that they prefer and possibly have most records of.

5.2.5 Filtering

Filtering would be the process of narrowing down the results of the activities or the data that are used in order to create the statistics. Filtering may be applied by discipline or by date.

5.3 Personas

In order to better know our users and the features we need to implement, we had to create personas. A persona is a way to model, summarise and communicate research about people who have been observed or researched in some way. A persona is depicted as a specific person but is not a real individual; rather, it is synthesised from observations of many people. Each persona represents a significant portion of people in the real world and enables the designer to focus on a manageable and memorable cast of characters, instead of focusing on thousands of individuals. Personas aid designers to create different designs for different kinds of people and to design for a specific somebody, rather than a generic everybody. Personas are an essential part of goal-directed design. Each group of users researched is represented by a persona, which in turn is represented by a document. Several personas are not uncommon in a typical project.

The components of goal directed design that support personas are the following two:

1. End goal(s) : This is an objective that a persona wants or needs to fulfil by using software. The software would aid the persona to accomplish their end goal(s) by enabling them to accomplish their tasks via certain features.

2. Scenario(s) : This is a narrative that describes how a persona would interact with software in a particular context to achieve their end goal(s). Scenarios are written from the persona's perspective, at a high level, and articulate use cases that will likely happen in the future.

Personas, end goals and scenarios relate to one another in the same way that the main character in a novel or movie goes on a journey to accomplish an objective. The classic "hero's journey” narrative device and its accompanying constructs have been appropriated for the purpose of designing better software.

Anna Chicherova (RUSSIA)

“ I use trafie as a tool. It makes it very easy for my to visualise my progress and find out things that affects my performance.” ~ Anna

Anna is a 28 years old High Jump Olympic Champion, with many achievements the last decade in world class Championships.
She competes in about twenty to thirty competitions around the world per year. And she wants to see her progress during the year, and easily understand how her training affects her competition.

She adds the performances from it’s mobile device (iPhone 5S, iOS, 4” screen size), exactly after the competition, because she needs to remember how the competition went and keep notes about it. This means that most of the times this is happening at night and on the way back to hotel or any other area of the competition venue.

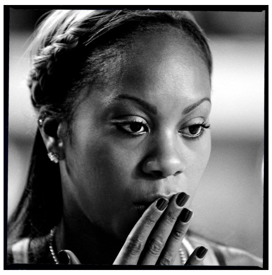
For Anna a mobile device that allows her to keep track of performance and notes chronically close to the event is very important.

Tomasz Majewski (POLAND)

“ I use trafie as an archive. I keep track of my progress through the years.” ~ Tomasz

Tomasz Majewski is a 33 years old Polish Olympic Gold medalist in Shot Put. He competes in about thirty competitions around the world per year. He rarely uses his smartphone for adding performances because it’s very hard for him to use them, due to his big hand size. He is 2.05m height and weights 145Kg!

For this reason he uses our application from his laptop (Dell, Windows 7 OS, 15” screen size). He wants to just to keep an archive with his performances in time. For Tomasz an easy to use desktop application is all that he needs.



Sanya Richards-Ross (USA)

“ I use trafie as a performance archive. I like the way that visualise my performances and show how they changed in time.” ~ Sanya

Sanya Richards-Ross of the United States she won the gold in the Women's 400 Metres Final during the 14th IAAF World Indoor Championships. She has many achievements the last decade in 400m and 200m. She competes in about thirty to forty events around the world per year. She loves technology and she mainly uses her smartphone (LG Nexus, Android Lollipop, 4.5” screen size) and her tablet (iPad mini, iOS, 7” screen size). She always has these devices with her.

She may use our application in many different ways. She uses it on the way to stadiums or back to hotel, in the bus after the competition or in cafes while she relaxes between trainings. She wants to have instant access to their performances and see how their affected by her training, diet etc.

5.4 DEVICES SUPPORT

5.4.1 Desktop

The user interface of the application should adapt to the screens of almost all devices. On desktop computers and laptops with medium and large screens, it should reach its maximum size (around 1000 pixels) and not expand more than that. All the elements of the application should occupy 100% of the width of that size, so that there is no empty space.

5.4.2 Tablet

On tablets and laptops with small screens, the width of the application should match the width of the screen. All the options and the features should be available but possible some of them can be grouped into a single option that has to be activated in order to reveal the full set of options, in order to save some space. It should work on both landscape and portrait mode.

5.4.3 Mobile

On mobile phones, the width of the application should match the width of the screen again, and the features should be reduced to the minimum required just for user input. Parts of the application (for example the section of the statistics) should not be available at all. Options should be grouped in order to save space and the application should work in both landscape and portrait mode.

5.4.4 Wearables (usage of APIs)

Wearable devices should be able to access the application using the available API. All the features regarding manipulating the data of the activities, should be available using the API, so that users can track their performances using the sensors of their device.

6. SOFTWARE DEVELOPMENT PROCESS

6.1 Sprint 1

6.1.1 Aim

In this first sprint, we will make a requirement analysis and analysis of the technologies that should be used. Also prototypes and mockups will be designed, but no code will be written. Everything will be prepared in order to start the second sprint by developing the actual application. This sprint should have a duration of one week.

6.1.2 Requirement Analysis

The main target of the application is to help track and field athletes, to keep track of their performances and visualise their data in a way that they can find out how much they have improved and extract additional data in an easy and fast way. Therefore there are some feature that the application should definitely have and some others that can improve the users' experience and the interaction of each user with the other users. We will use the agile development methodology, so while the application is being developed, we will be getting feedback from the users and change the requirements accordingly.

The most important element is the form where users can add, edit and remove activities. The activities are the main data that our application will handle. For this reason it should be easy to complete and change, so our users can import and manipulate their data often and without hesitating. This raw data will be the source for all the visualisations and diagrams in the application. So diagrams is another important feature that should be implemented. Diagrams should use the user's data and display line charts of time spans selected by the user. They should also filter data by discipline. Apart from the diagrams, another way to view your activities is in a list view at profile page. Activities should also be filtered by discipline or year of the activity.

There will be social features so the profile should resemble that of a social network. There should be a profile picture, search input that searches for users, the profile and the individual activities should be set to private, hiding them from the other users and also the diagrams of the users who have their profiles set to public, should be accessible.

The agile software development method is flexible towards requirements which are likely to change, and the analysis for the first sprints of the project should be done only for the features that will be implemented in these and have detail analysis for final product. In this sprint we will discuss the requirements that will likely be the foundations for the upcoming features, so they will include ideas that may affect the way features are implemented in the future.

The most important feature that should be implemented in the next sprints is the page where the athlete can record and view their activities. This is the core feature of the application. Activities, is the main content of the application and without them it offers nothing to the user. Therefore, the feature that should be implemented in the next sprint, as a prototype, is the way that athletes will record their activities.

The second core feature is the user as an entity in the application. There should be a system for creating users and a system that discerns individual users that use the application, namely, a registration and a login system with user sessions.

Both core features require records in the database, so the design process should include the db schema and its interaction with the application. When the prototype is ready, the core features should be enhanced and more features should be added in the next sprints.

6.1.3 User Interface prototypes and evaluation

In this Sprint we need to take some basic decisions about our User Interface and Interaction design. How our pages and our main flows should look and work. Because of the small timeframes we use in Agile we were needed a fast evaluation method that evolve real users, in order to get a first feedback about our design. Thus we choose Paper Prototypes and User Walkthroughs. Combining the information of our users, with design principles that are followed in web application design, lead us to our first design. In order to create many different User Interfaces we split the total UI in partials and for each one we create a lot of variations. With this technique we could easily generate a lot of different User Interfaces and let our users interact with them. On the following images are shown different variations of the basic pages and elements of our design.

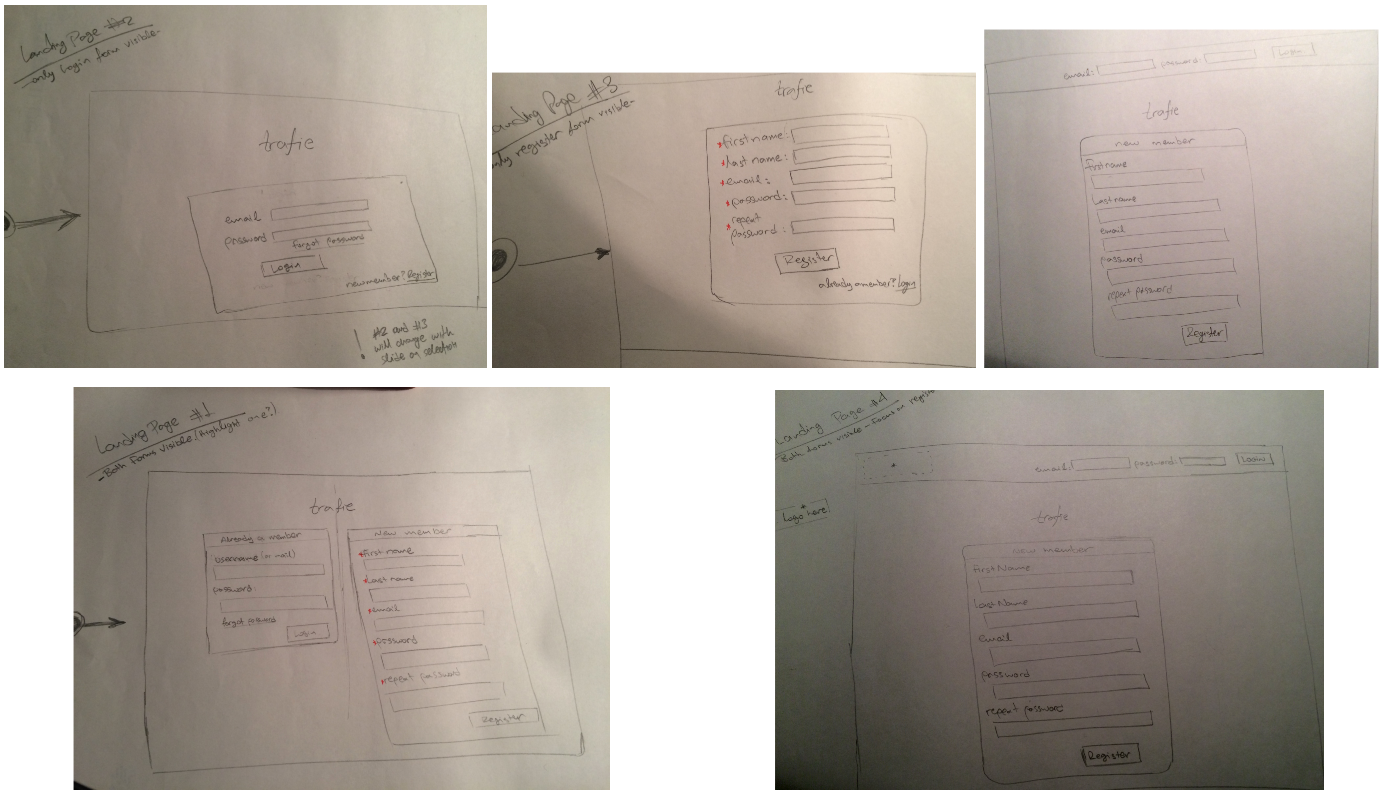


image 1.0: Paper prototypes for login pages

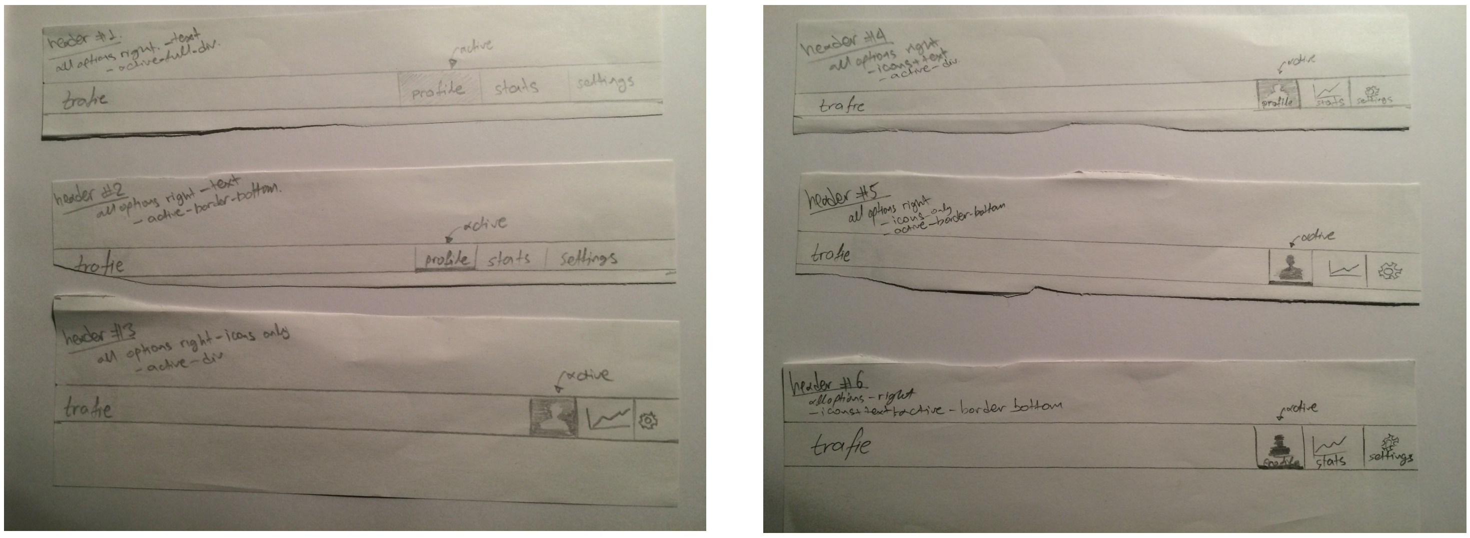

header #1
all options right - text - active-full-div
active
trafie
profile
stats
settings
header #2
all options right - text - active-border-bottom
header #3
all options right - icons only - active-div
header #4
all options right - icons+text - active-div
header #5
all options right - icons only - active-border-bottom
header #6
all options - right - icons+text - active - border bottom

image 1.1: Paper prototypes for application header

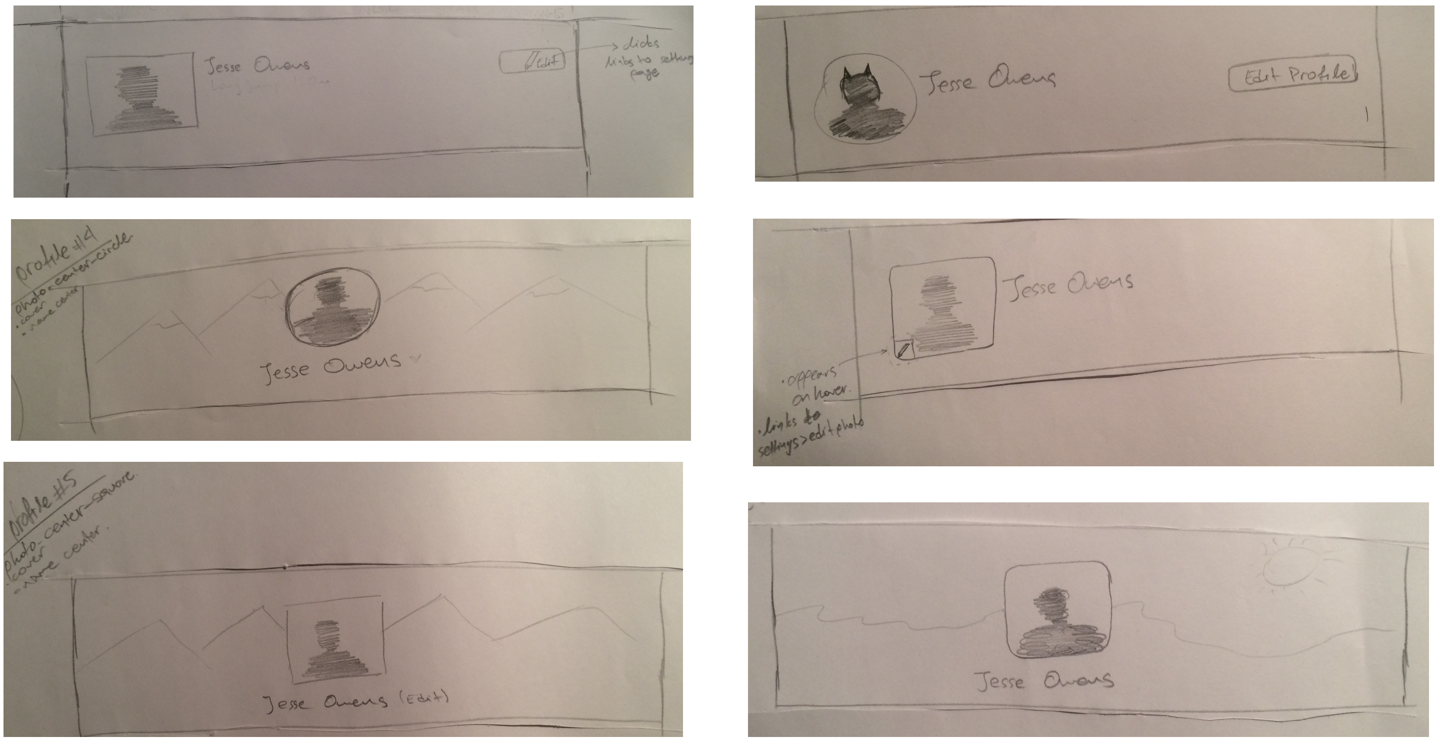


image 1.2: Paper prototypes for profile header

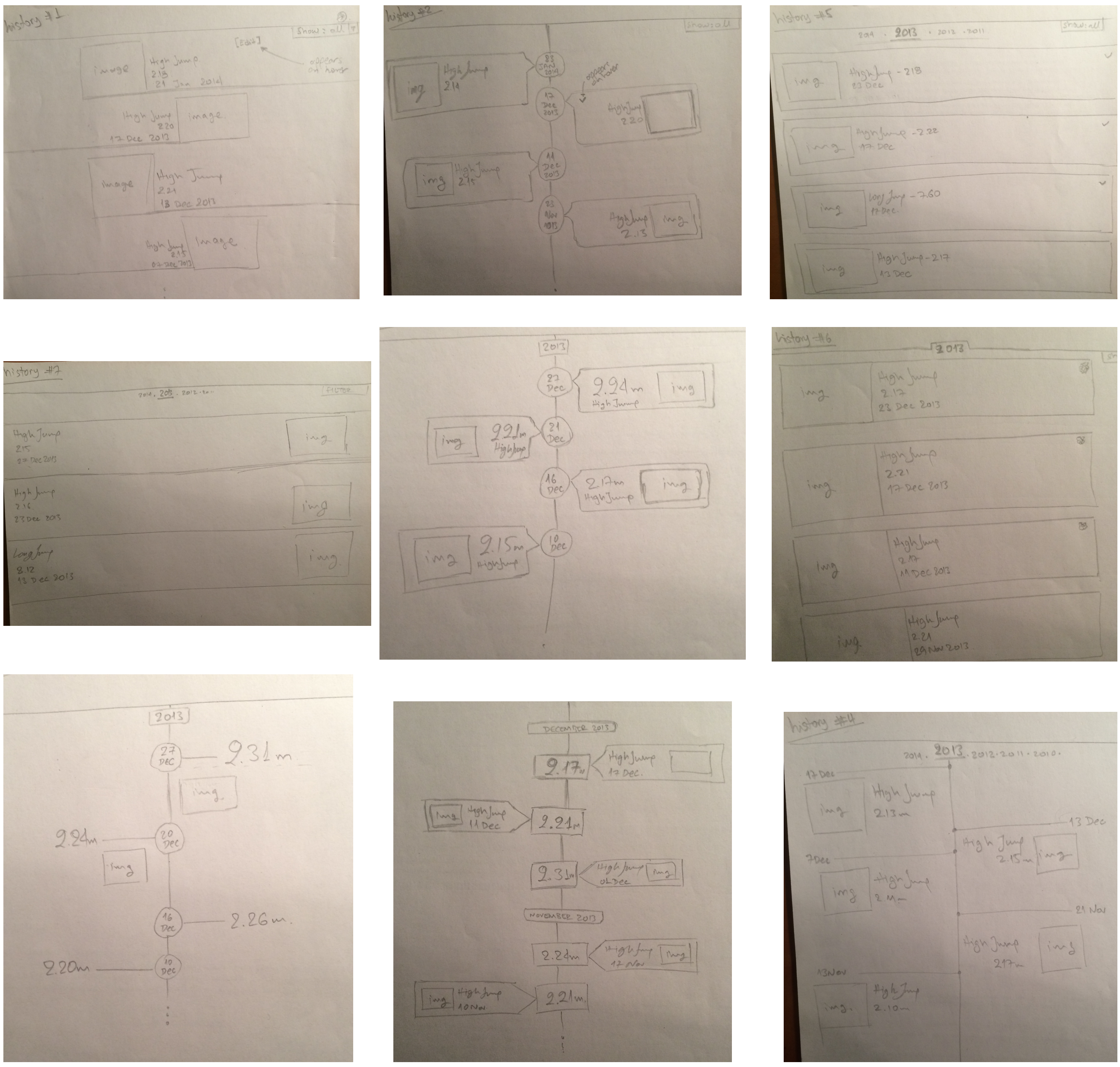


image 1.3: Paper prototypes for activities timeline (initially we named this timeline ‘history’)

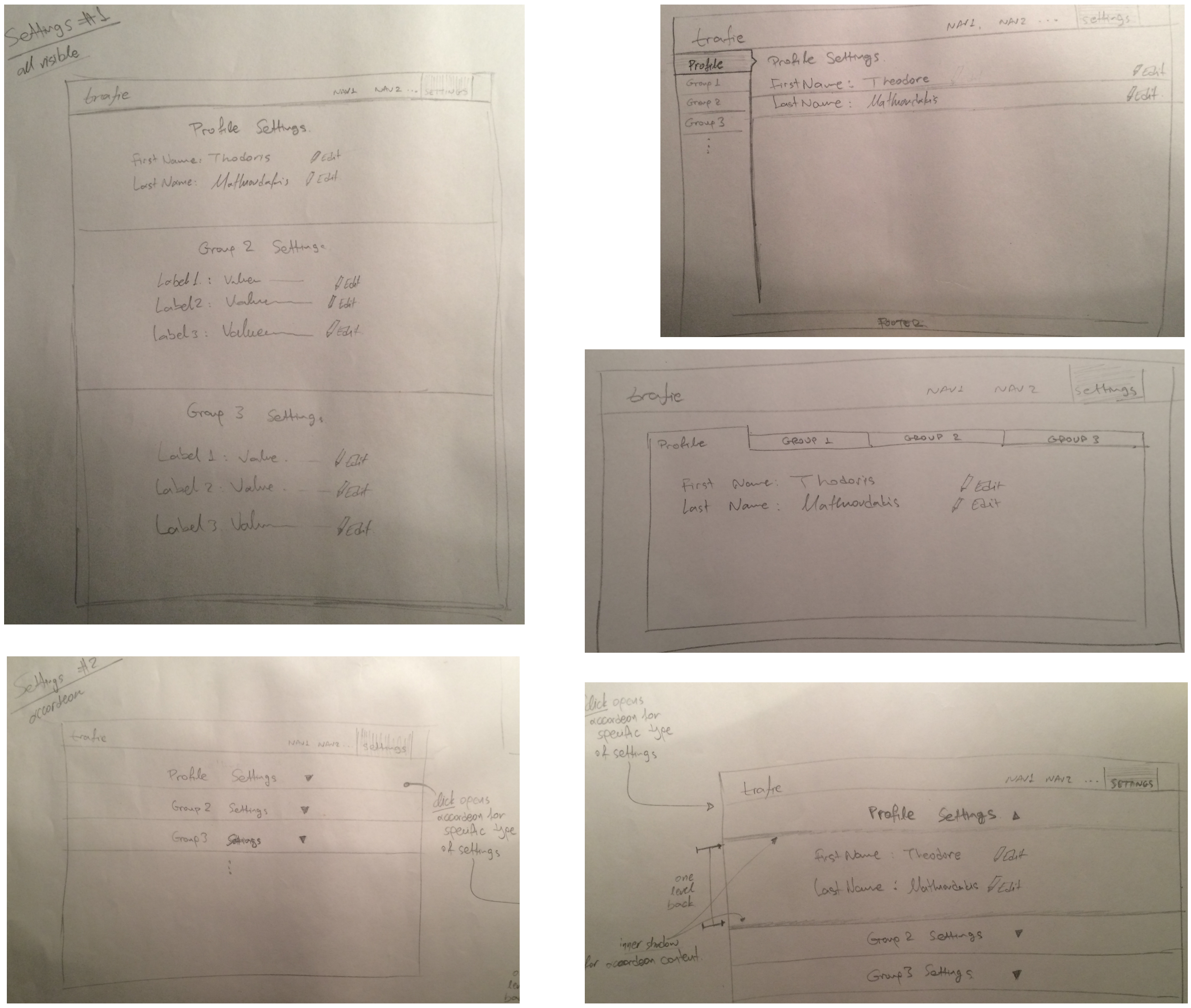


image 1.4: Paper prototypes for settings pages

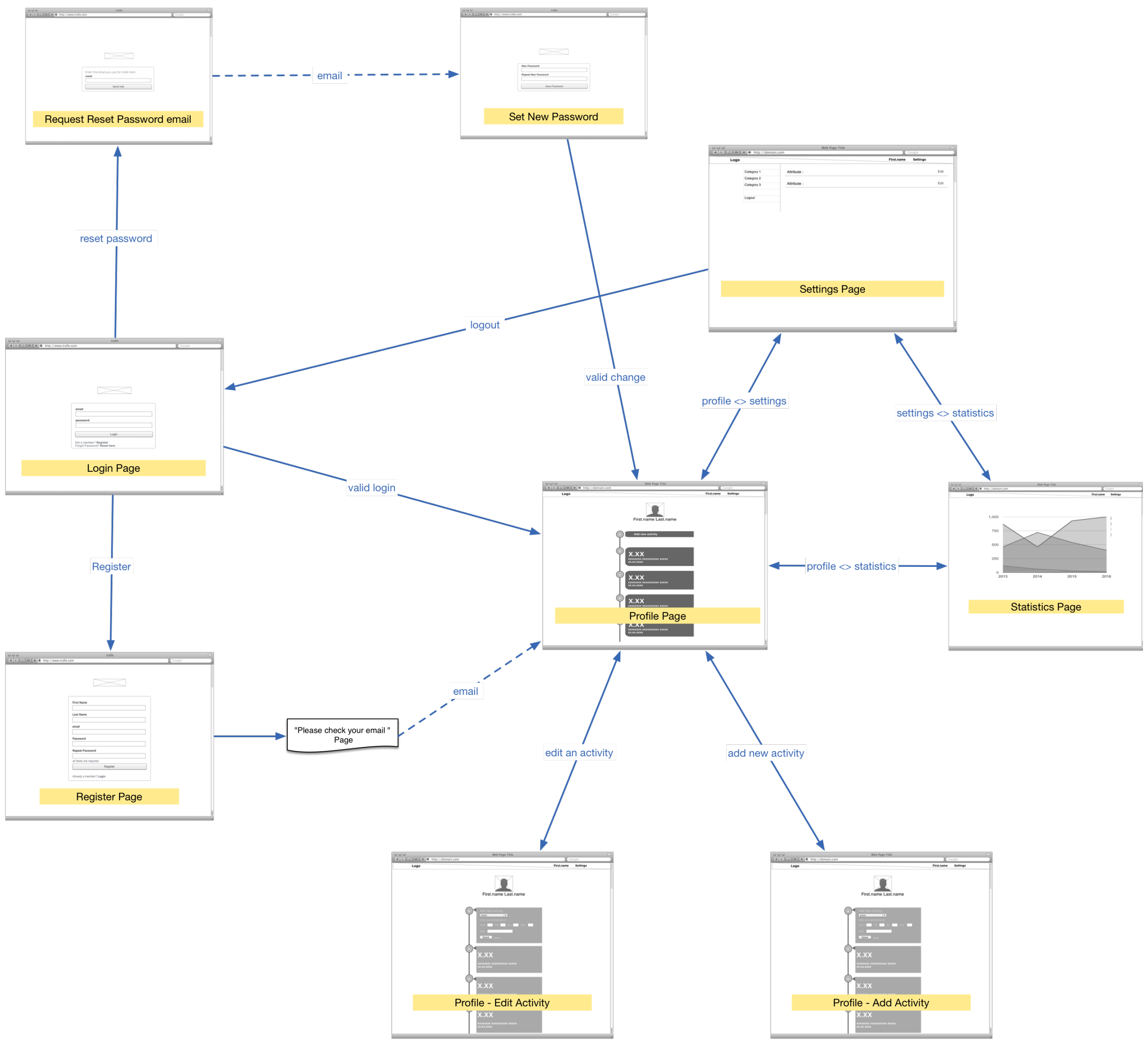


image 1.5 : Full application’s page flow

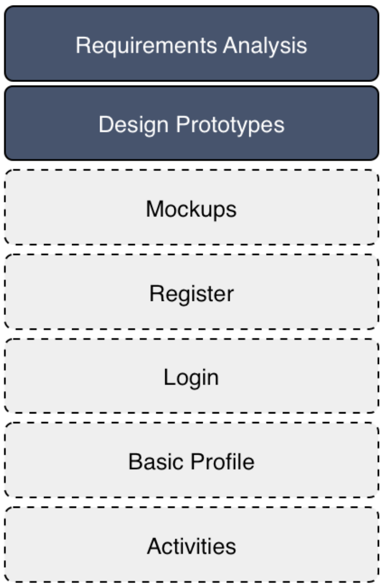


image 1.6 : Backlog status for Sprint Planning of Sprint 1

6.2 Sprint 2

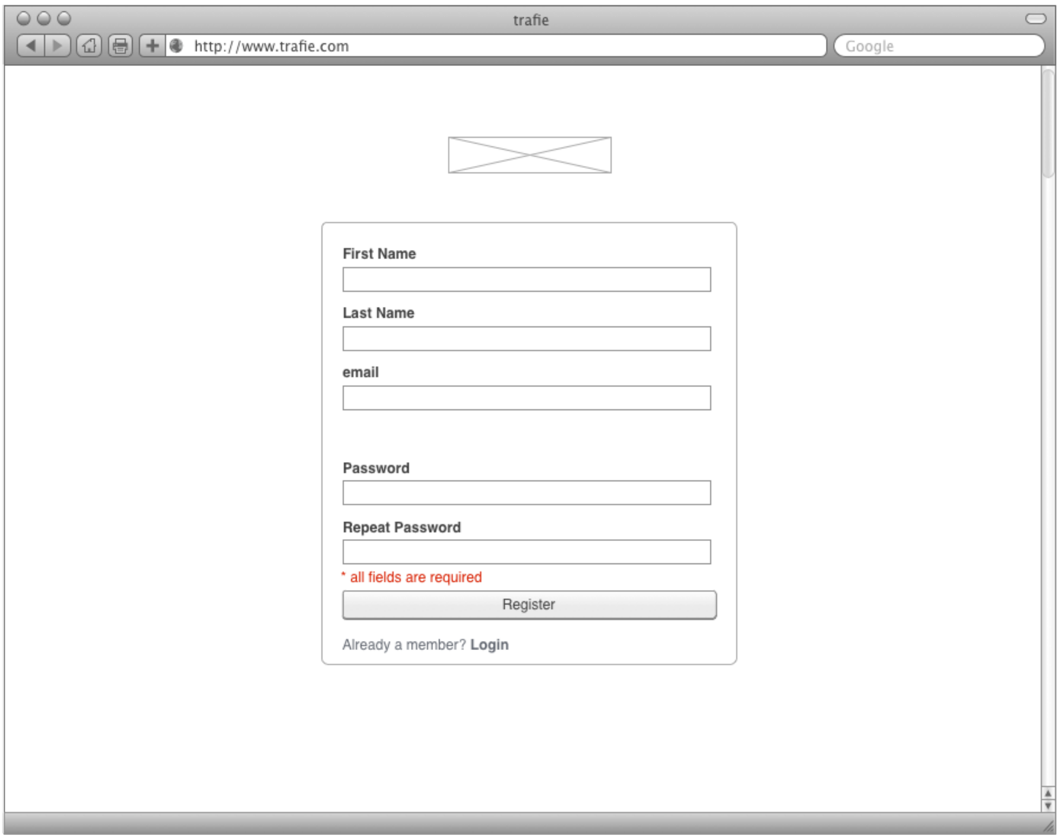
6.2.1 Sprint planning

The features that should be implemented in this sprint are the login page, the registration page and the profile page that allows only creating new activities. My colleague will set the fields for the users and the activities in the database and I will create the user interface for the login and registration functionality. On the server side we will also keep the sessions of the users. Then we will create the routes and the functionality for activity creation and retrieval.

This sprint can be split in 3 stories: the login functionality, the registration functionality, the activities API(login, register, profile).

The server, should expect from the registration page five parameters for registration: the user's first name, last name, email address, password and the password one more time for verification. The first name and last name values should have only characters and no symbols.

The email address should have a valid format and there should be no other such email in the database. The passwords should be at least 6 characters long and they should match.



First Name

Last Name

email

Password

Repeat Password

* all fields are required

Register

Already a member? **Login**

image 2.1 User Interface of registration page with an error message visible

Nothing more should be implemented regarding the registration process at this time. After registering successfully, the user should be redirected to the profile page. The user's submitted data should be stored in the database as they were given, apart from the passwords which should be encrypted with the sha-1 hash function.

For the login process, the email address and the user's password should be asked. These two pieces of information should match a record in the database in order to let the user login, otherwise, a message should be shown, stating that the credentials were wrong. After submitting the credentials successfully, a session should be created for the users, which should be stored in the memory for the time being, and a cookie should be sent to the browser, allowing the user to browse the private pages without any other authentication.

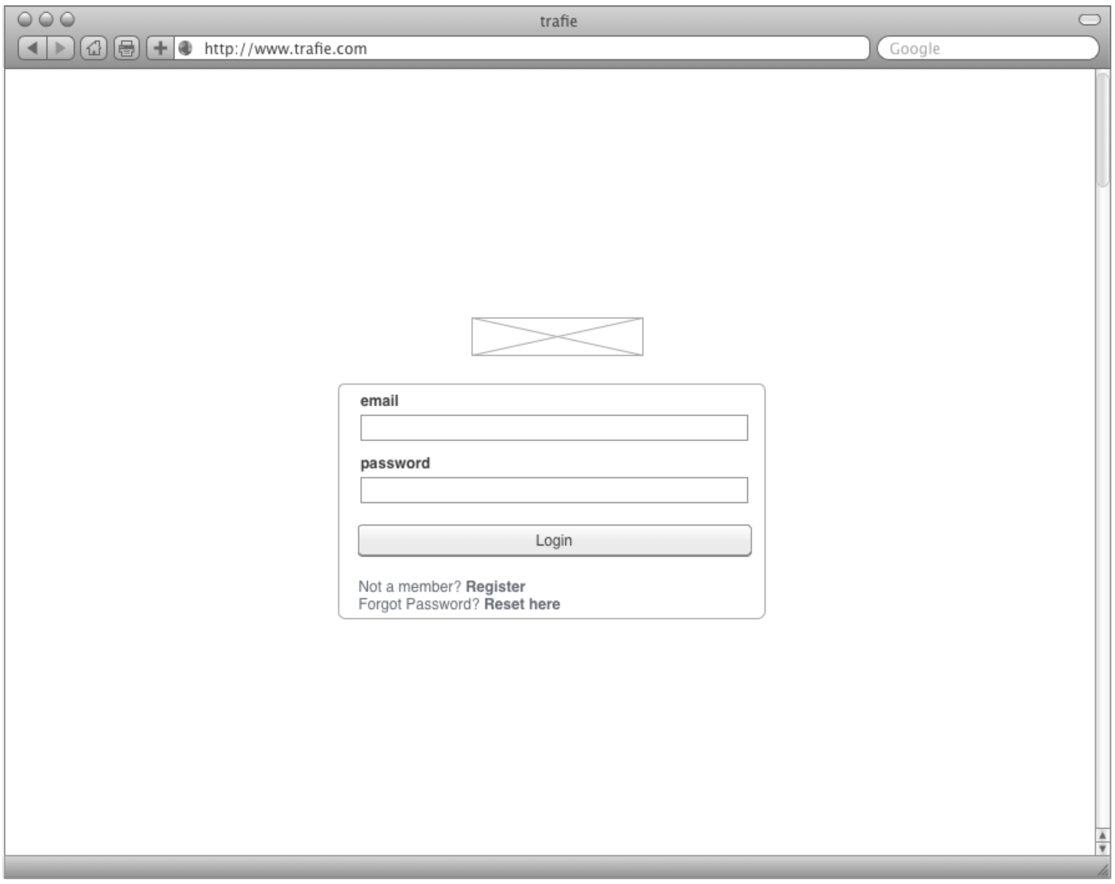


image 2.2: User Interface of Login page

All data are computed in backend with PHP and added inside the markup before they will be send to the web client.

The API that manages the activities should only support two methods, GET and POST. The resources should be on the route /user/{user_id}/activities. A GET request on this route will fetch all the activities of the user with the given user_id, and if an id is given for an activity, only that activity should be fetched. An example route is GET /user/1/activities/1, which should fetch the activity with id 1 of the user with id 1. When a POST request is made on the resource route, a new activity should be created, according to the posted data. Whenever the API is accessed, the user should be authenticated by their session data. Also, when a user posts an activity, the posted data should be validated before they get stored in the database.

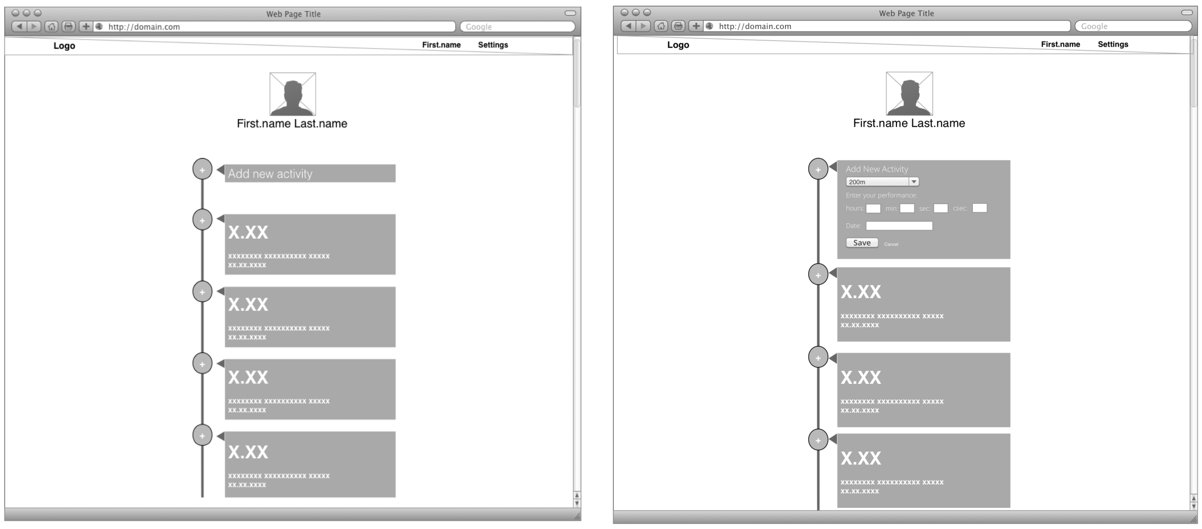


image 2.3: User Interface of Profile page with and without the add activity form

6.2.2 Stand up

Our initial point is the registration and login page where our users have the first contact with our app. On the registration section, the user will have to fill 5 fields. We tried to have only necessary fields in order to ensure security,prevent errors and get as less as data are needed in order to initiate our user profile. The fields we finally ask from user to fill are first name, last name, email address, password and repeat password. When these fields are submitted, a POST request is made, on the “/register” route. We check the validity of the values on server side. If we get invalid data from client, the same markup should be sent back with additional informing messages that will help the user enter the correct values. If they are valid, the first_name, last_name, email and the password(hashed by the sha256 algorithm) are stored in the database.

The login page has two fields, email and password. When the user submits the form, the server gets the posted data, hash the password with the sha-1 algorithm and perform a query, searching for a record in the “users” table, that corresponds to this data. If a result is not found, the server should return a message, stating that the credentials entered was wrong. If a data match then a session is created, cookie data should be sent to the browser and the user should be redirected to the profile page. The user should be authorised to access his profile and his statistics page, plus the routes that fetch and alter the activities. The sessions are stored in memory, using the middleware named “connect”. Whenever a request is made to the server that includes the cookie data, the server will be able to check if the session data in the cookie is the same as the data in the memory and verify the user as logged in. Otherwise, it should redirect them to the login page.

In the database schema there are two tables where the users’ data are stored. The “users” table, which stores data that is related to the user’s account, and the “profile” table, which stores data related to the users’ profile pages. The first name and last name values will be stored to the profile page, while the email address and the password will be stored in the users table. Of course as the development continues, in later sprints, new values will be added in these tables in order to extend users’ profiles. Since there is no email validation, a session is created in the memory of the server, and a cookie with the session id is sent back to the client.

When the user requests the profile page, the data from the table of the database “profile” (first_name, last_name) should be sent to the browser, personalising the page. Then we will request the activities of the user, from the API route /users/#user_id/activities. The table “activities” should be queried and the activities with the user’s id should be fetched and returned in order to create the page with the correct data and return them to the client.

A main problem that will also be faced is the validation of the activities when they get posted to the server. In order to handle easier the different activities we had to categorise them. The criterion, with which we categorised them, is the performance unit, so there will be three categories. The first category includes the disciplines that are measured with time, which are races. The second one the disciplines, the performances of which are measured by distance, like jumps or throws. The last category includes the disciplines that are measured by points, which are the combined events(pentathlon, decathlon etc).

When a user posts an activity, the first step of the validation process would be to check if the posted discipline is a valid discipline. Then, based on the type of the discipline, the performance should be checked for validity. Performances counted in points should be integer numbers, distances should be float numbers and time should be a time-formatted String. Also when posting an activity we get a value for the date, which should also be validated for its format and it should be a past date. After we receive the date value we convert it to the format that the database accepts.

After the validations have passed and the activity is considered valid, a new record should be created in the “activities” table, with all the posted data plus the user’s id. A JSON object with the formatted activity data should be returned to the clients, in order to update the view with the new activity. In case the data is not well formatted or not valid, a message should be sent to the client stating where the error is located. In this sprint, the activities can only be created and not deleted nor edited.

The last feature that will be implemented is the page of the graph. When user navigates to the graph’s page, web client sends a request to the server using user ID in order to get back the activities of the user and rendered them as graphs using Google Charts library. When the request is done the server side just queries the database for the activities of the user and sends them back to the client.

The development stack we are going to implement is mongoDB as a database management system, PHP for backend in order to handle logic and html templating, HTML-Javascript-CSS in client side in order to render the UI pages.

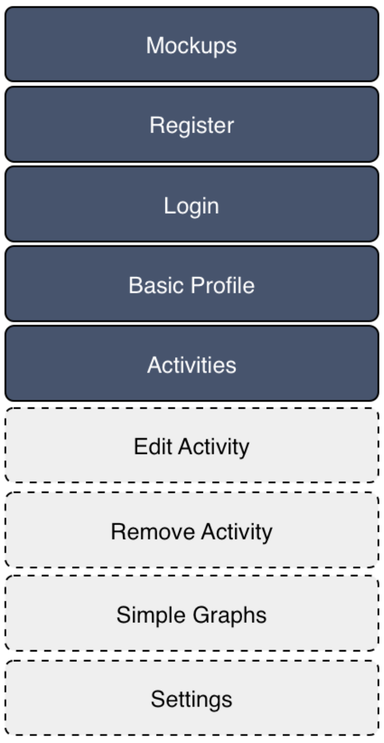


image 2.4: Backlog State for Sprint Planning of Sprint 2

6.2.3 Diagrams

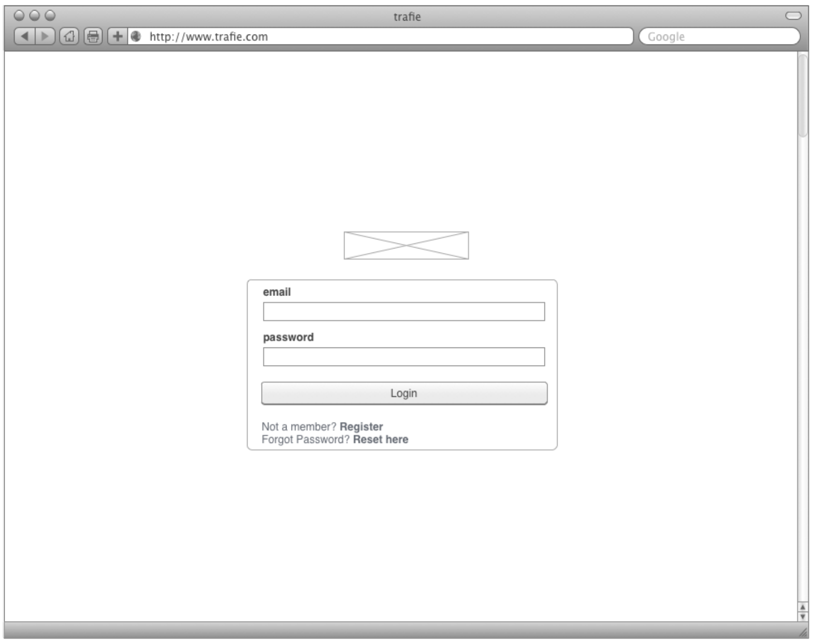


image 2.5: User Interface of Login page

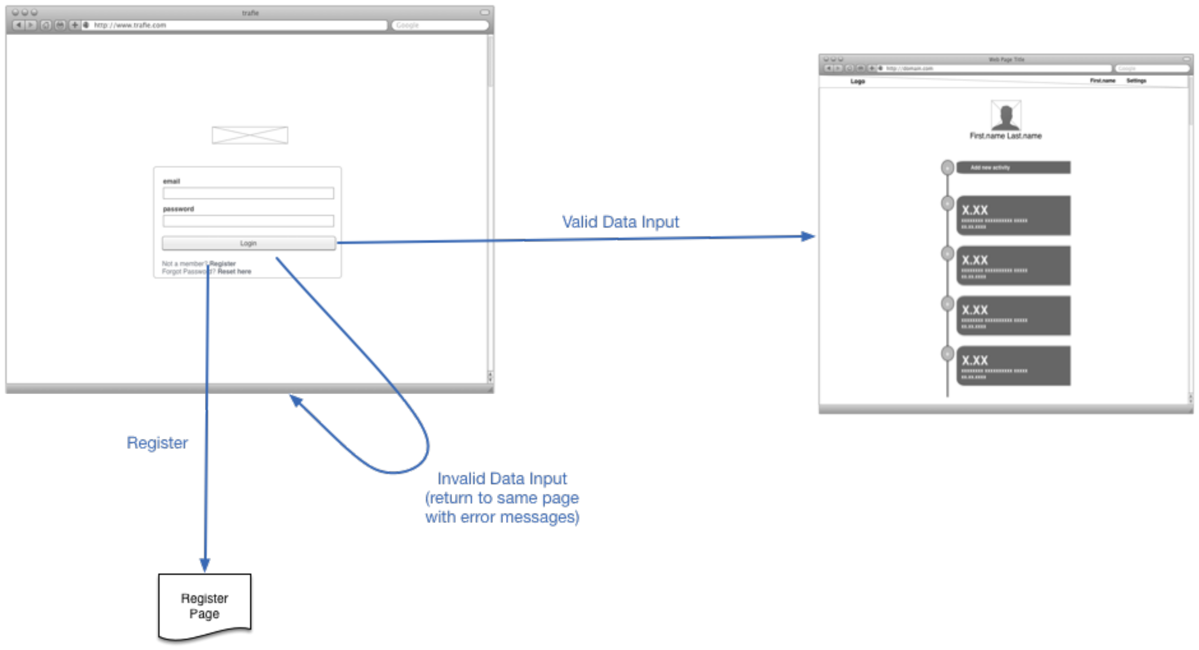


image 2.6: Login Wireframe

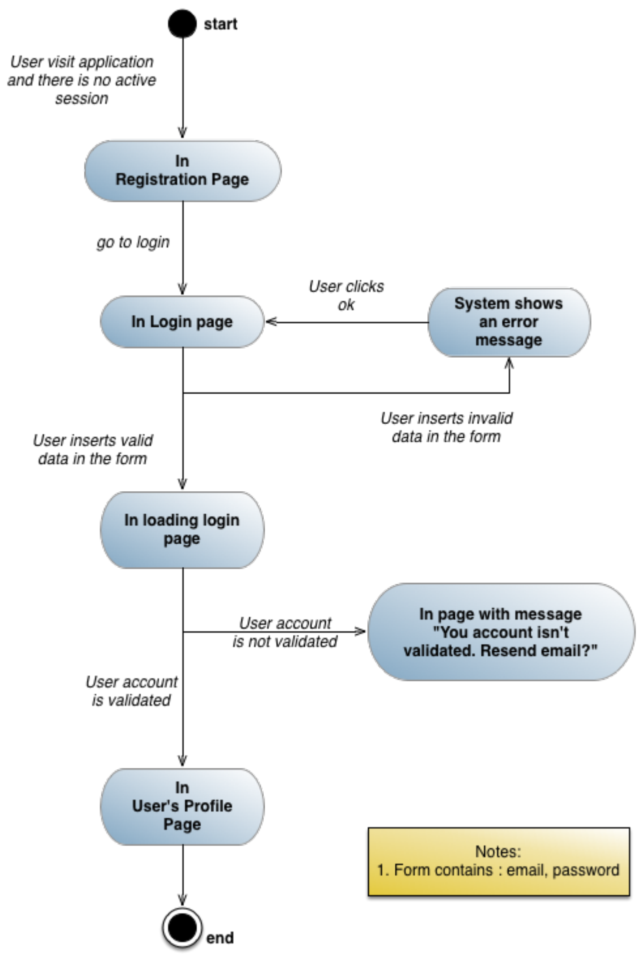

start
User visit application and there is no active session
In Registration Page
go to login
In Login page
User clicks ok
System shows an error message
User inserts invalid data in the form
User inserts valid data in the form
In loading login page
User account is not validated
In page with message "You account isn't validated. Resend email?"
User account is validated
In User's Profile Page
Notes:
1. Form contains : email, password
end

image 2.7: Login State Diagram

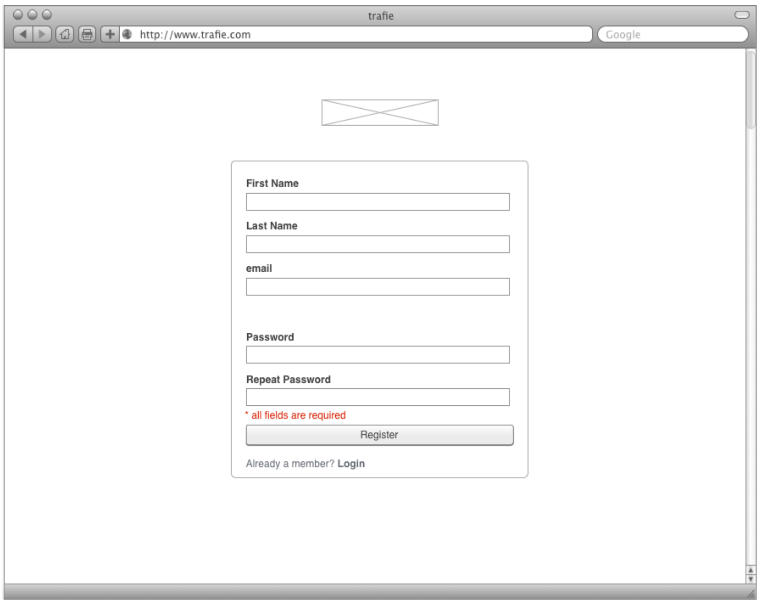


image 2.8: Register User Interface

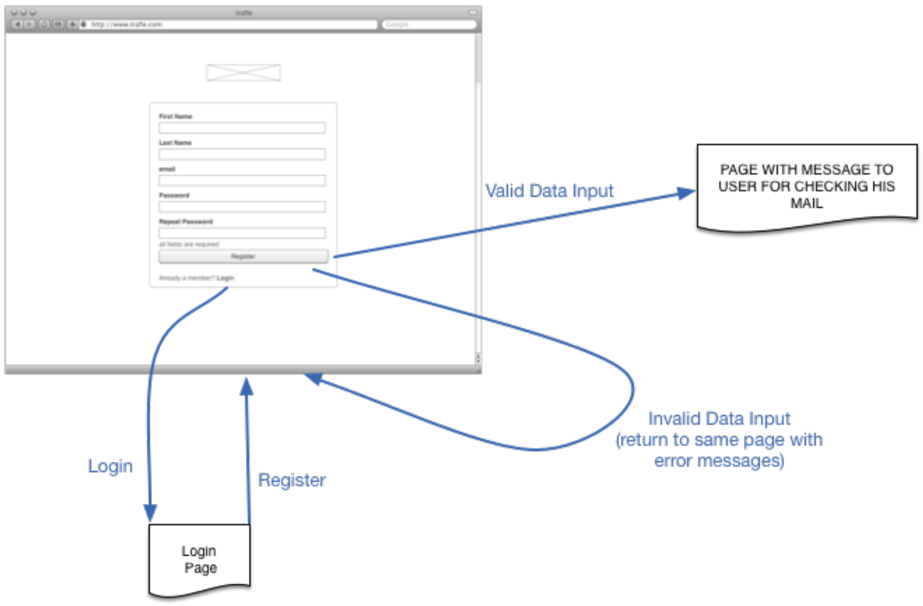

Valid Data Input
PAGE WITH MESSAGE TO USER FOR CHECKING HIS MAIL
Invalid Data Input (return to same page with error messages)
Login
Register
Login Page

image 2.9: Register Wireframe

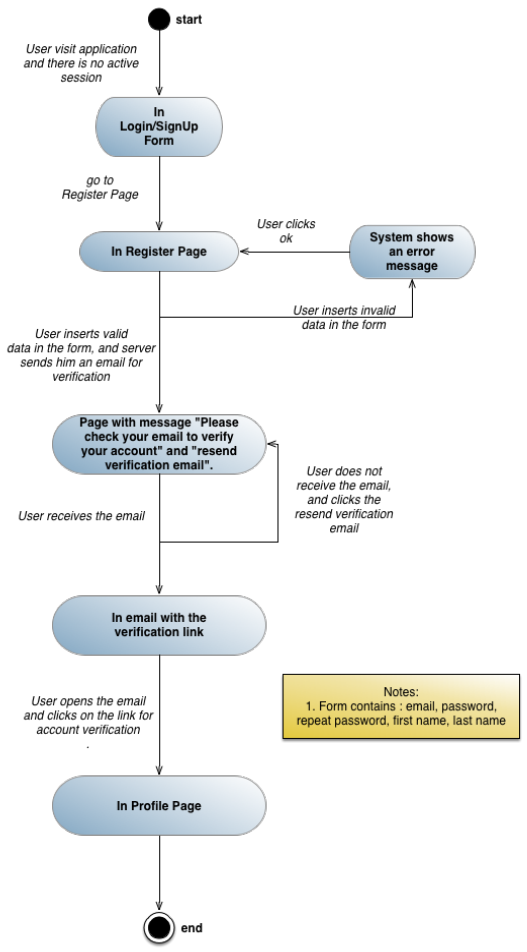

start
User visit application and there is no active session
In Login/SignUp Form
go to Register Page
In Register Page
User clicks ok
System shows an error message
User inserts invalid data in the form
User inserts valid data in the form, and server sends him an email for verification
Page with message "Please check your email to verify your account" and "resend verification email".
User does not receive the email, and clicks the resend verification email
User receives the email
In email with the verification link
Notes:
1. Form contains : email, password, repeat password, first name, last name
User opens the email and clicks on the link for account verification
In Profile Page
end

image 2.10: Register State Diagram

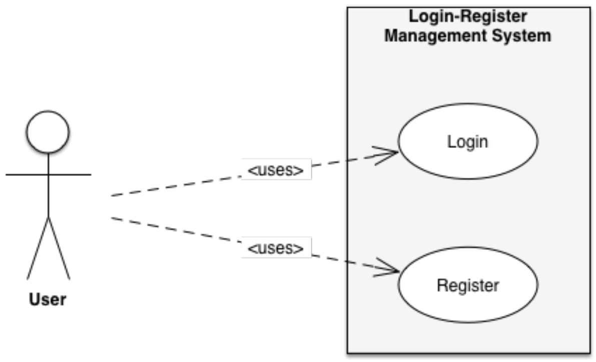


image 2.11: Login and Register Use Case

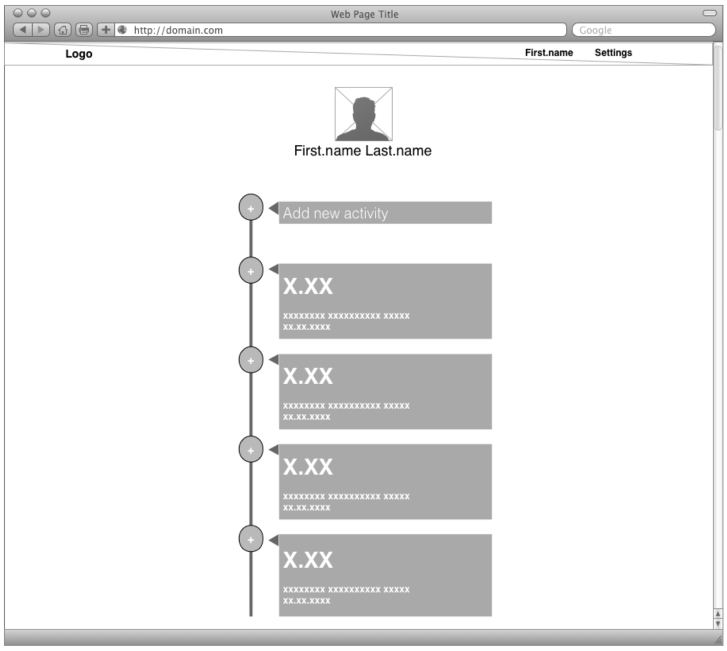


image 2.12: Profile User Interface

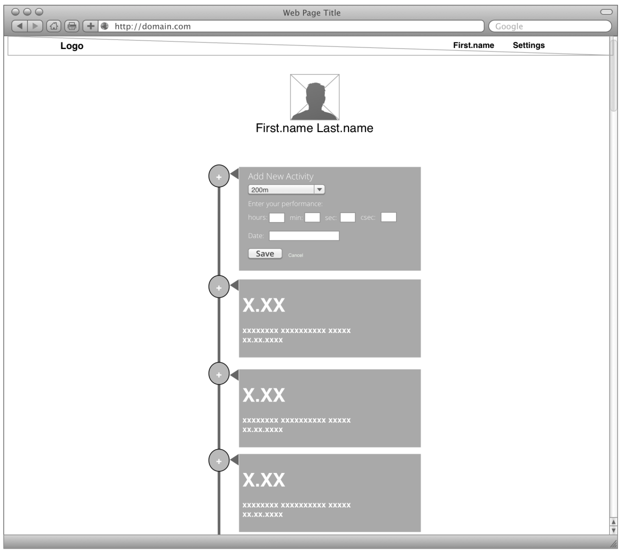


image 2.13: User Interface for Add Activity in Profile page

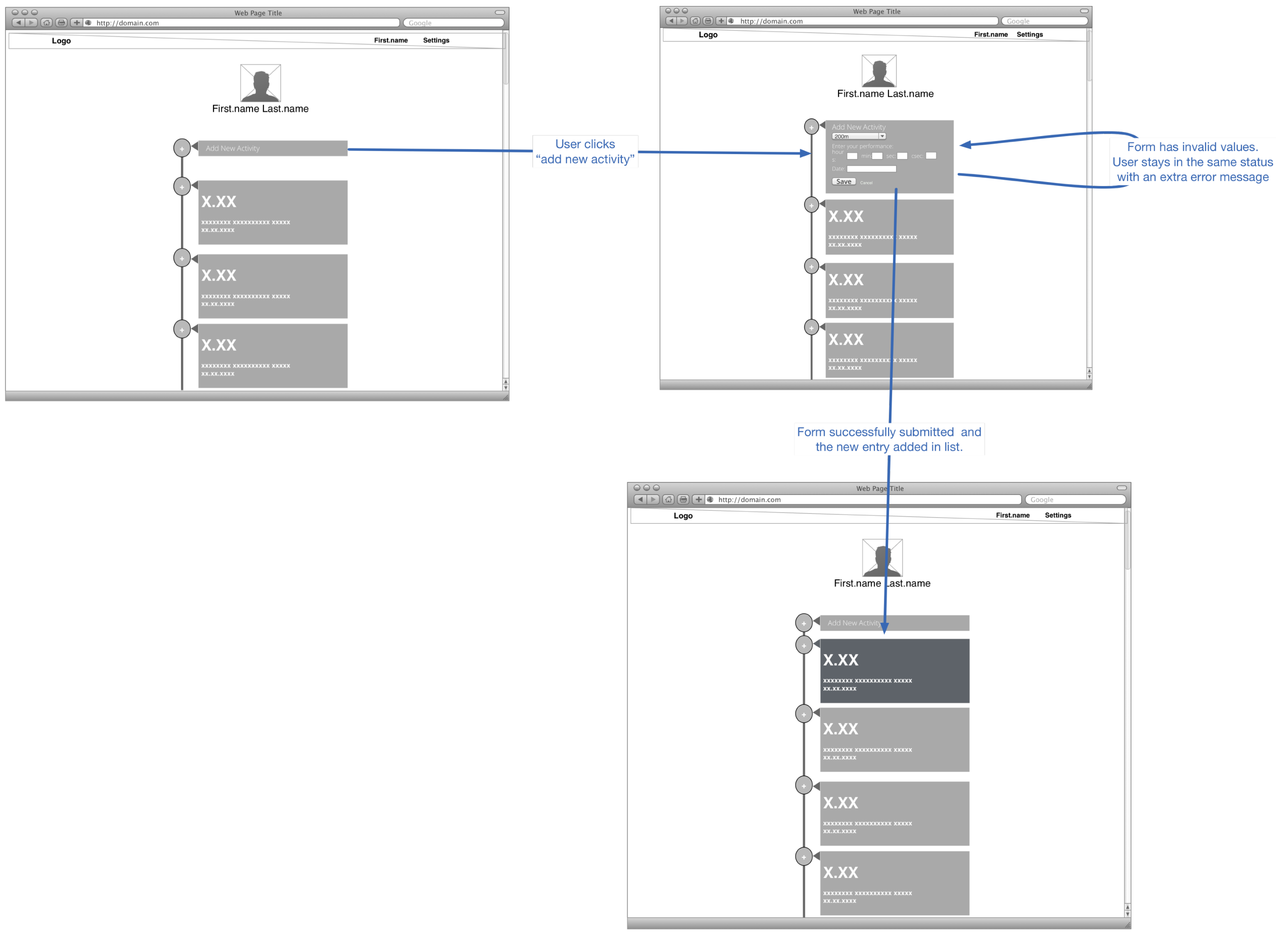


image 2.14: Add Activity Wireframe

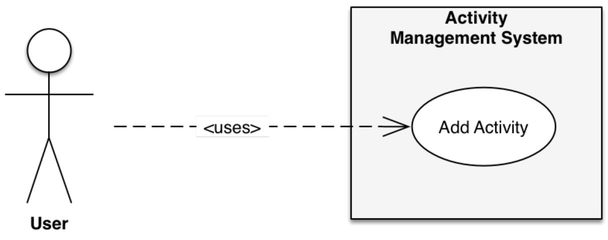


image 2.15: Add Activity Use Case

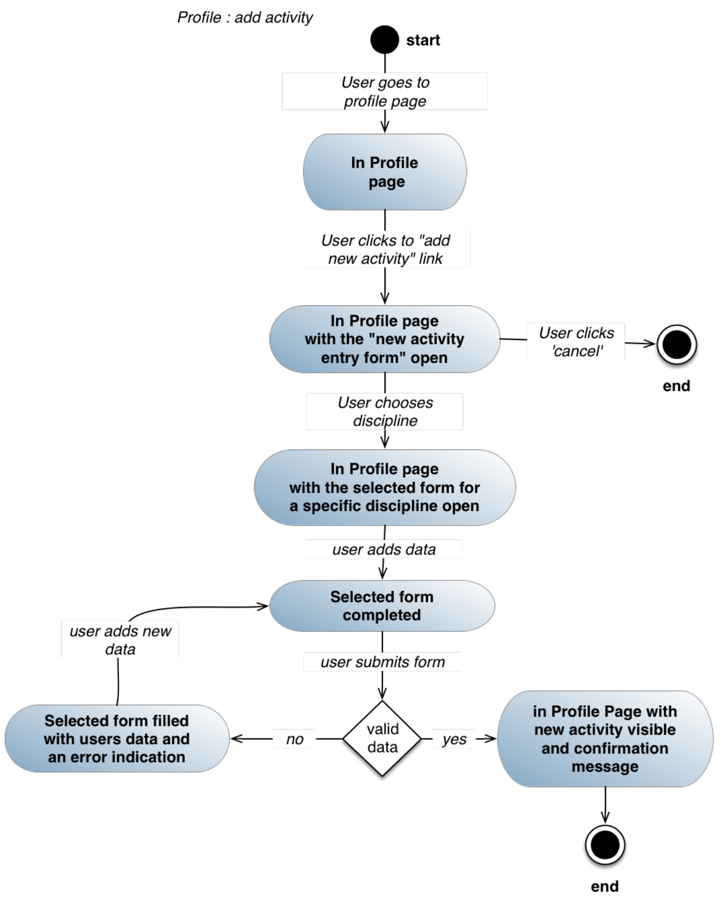

Profile : add activity
start
User goes to profile page
In Profile page
User clicks to "add new activity" link
In Profile page with the "new activity entry form" open
User clicks 'cancel'
end
User chooses discipline
In Profile page with the selected form for a specific discipline open
user adds data
Selected form completed
user adds new data
user submits form
valid data
no
yes
Selected form filled with users data and an error indication
in Profile Page with new activity visible and confirmation message
end

image 2.16: Add Activity State Diagram

6.2.4 Sprint review

In this sprint there where some delays because we were needed sometime to learn and understand how mongoDB works and how it's differentiated from common SQL DBMS. Using a backend that needs to handle all the information, the logic of the UI and page navigation was the first issue we faced in the JS-PHP-MongoDB stack.

6.3 Sprint 3

6.3.1 Sprint planning

In this sprint, the essential feature we will try to implement is the deletion and editing of the existing activities. Also we will add an alternate view for existed activities, through graphs. This will add one more page in our total page flow.

Editing an activity should happen on the activity itself. The element that represents the activity in the client should have an icon, indicating that the activity is editable, which will transform all elements of the activity to input fields when clicked, creating a form. When the form is submitted, the changes should be sent to the server through an ajax call and the client should update the activity element with the data from the response.]

There should also be an element that when clicked, will delete the activity, by sending a delete request to the server and then updating the client. When user choose to delete an activity a pop up will be shown, asking the user to verify their action of deletion, protecting the activities from being accidentally deleted.

One new page will be added in order to have a different type of view for activities. This page will be called “Statistics”. For this functionality we will use Google Charts library in order to create the graphs in client side, just by using the same data we have for activities in profile page. For get the data from server we will request the activities of the user, from the API route / users/#user_id/activities, exactly in the same way we have in the profile page.

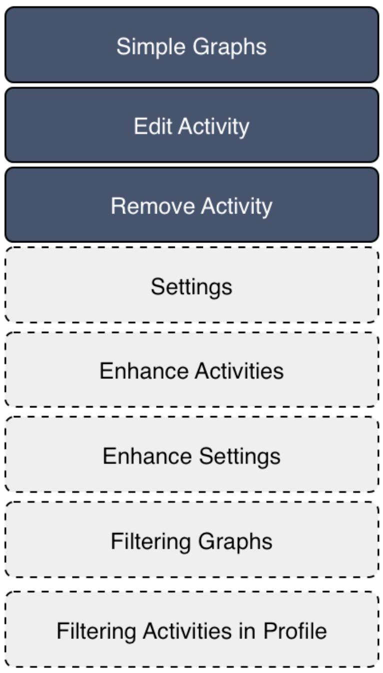


image 3.0: Backlog State for Sprint Planning of Sprint 3

6.3.2 Stand up

When the user has submitted the changes for an activity, the data should be parsed and validated by the server, the same way they are validated when a new activity is being created. Same validations apply here. The main difference here is that the request should be a PUT request this time, and not a POST request which is the case in the activity creation. Also, the activity_id value of the existing activity, that should be edited, must be included in the data sent. After the values have been validated, a query should find the existing record and change all the values according to the new data. If everything goes as expected, a JSON object should be sent back to the client, which will update the view. If something goes wrong, a message should be sent, stating where the error happened, and if the data was not valid, the user should be prompted to correct the data in the form.

Clicking the icon that deletes the activity, and confirming the action in the pop up, should send a DELETE request to the server, along with the id of the activity that should be deleted. The server should check if the activity exists, and if it does, it should delete the activity from the database. If everything goes as expected, a JSON object should be sent back to the client, so that the view can be updated, removing the element that represented the activity. If there was an error in the process, an error message should be sent, again in a JSON object.

Both actions of editing and deleting activities, should be validated by the server, regarding user permissions. This functionality may cause vulnerabilities when future functionality will be added, thus we implement validations in server side from now. Users can only edit and delete their own activities, so the id of the user to whom the activity belongs should match the id of the user in the user’s session and cookie. If the user tries to edit or delete an activity that doesn’t belong to them, the server should not proceed with the requested action and no message should be sent back to the browser, as these actions are not permitted using the interface, so the user has used other means to do achieve it.

In order to visualise user's activities in charts we need to manipulate the response with activities we get from the server, because different types of activities needs different type of charts and different type of axis. (meters, time, points)

This sprint has specific difficulties that rely mainly in different types of input data that has to be checked in front and back end. Also the usage of an unknown charts library in combination with different measurement units adds an extra difficulty in our time planning and committing for these features.

6.3.3 Diagrams

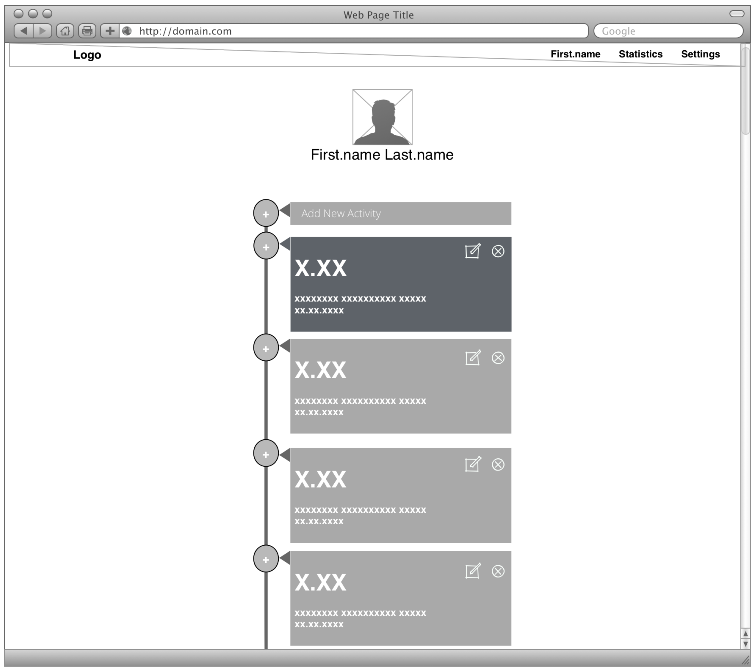


image 3.1: Profile page User Interface with edit option available

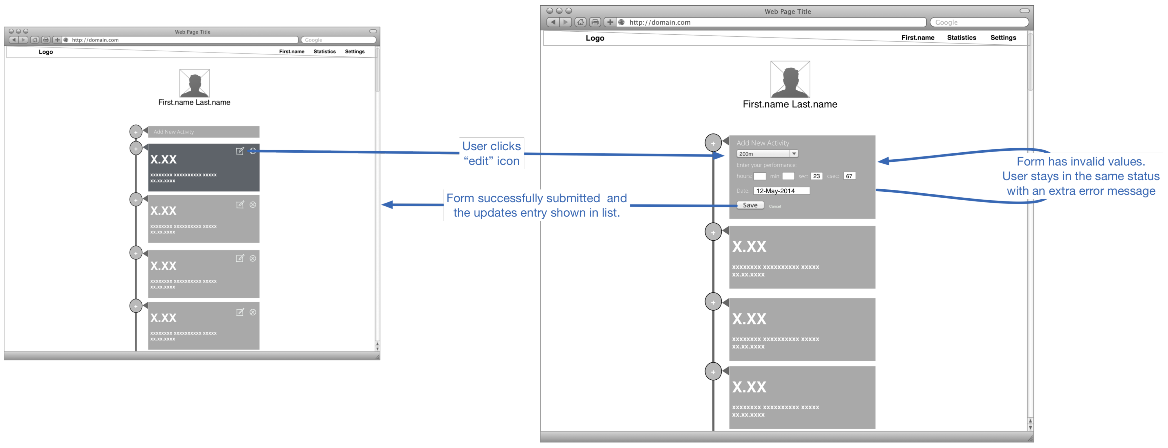

User clicks "edit" icon
Form successfully submitted and the updates entry shown in list.
Form has invalid values. User stays in the same status with an extra error message
First.name Last.name
Add New Activity
Save
X.XX

image 3.2: Edit activity Wireframe

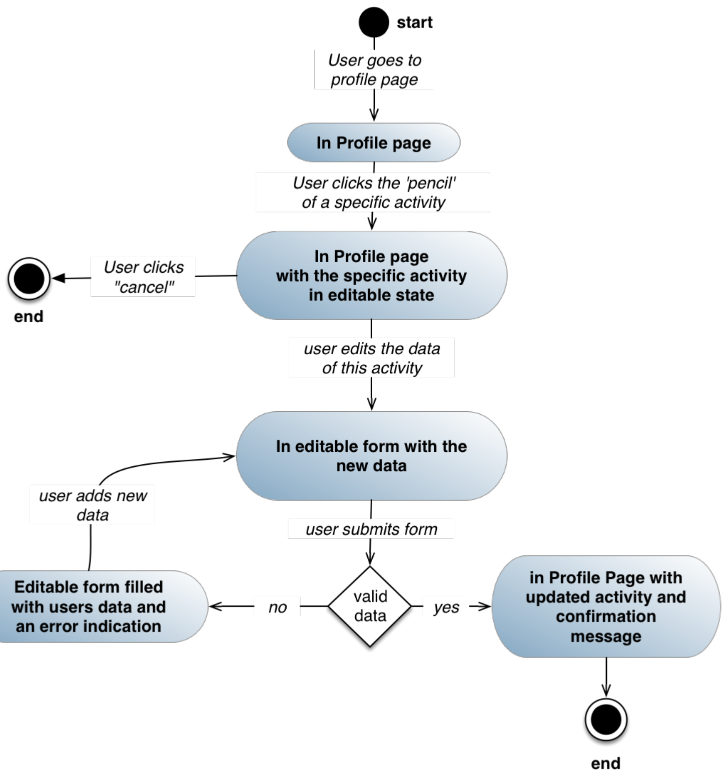

start
User goes to profile page
In Profile page
User clicks the 'pencil' of a specific activity
In Profile page with the specific activity in editable state
User clicks "cancel"
end
user edits the data of this activity
In editable form with the new data
user adds new data
user submits form
Editable form filled with users data and an error indication
no
valid data
yes
in Profile Page with updated activity and confirmation message
end

image 3.3: Profile page edit activity State Diagram

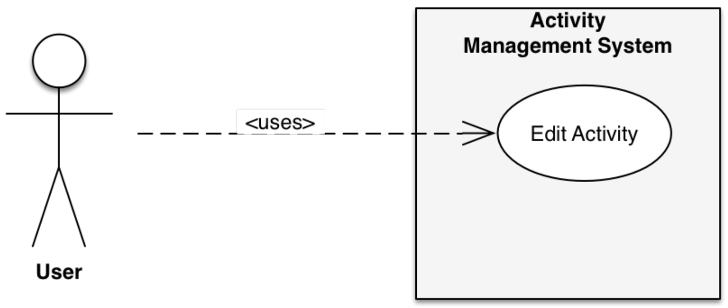


image 3.4: Profile page edit activity Use Case

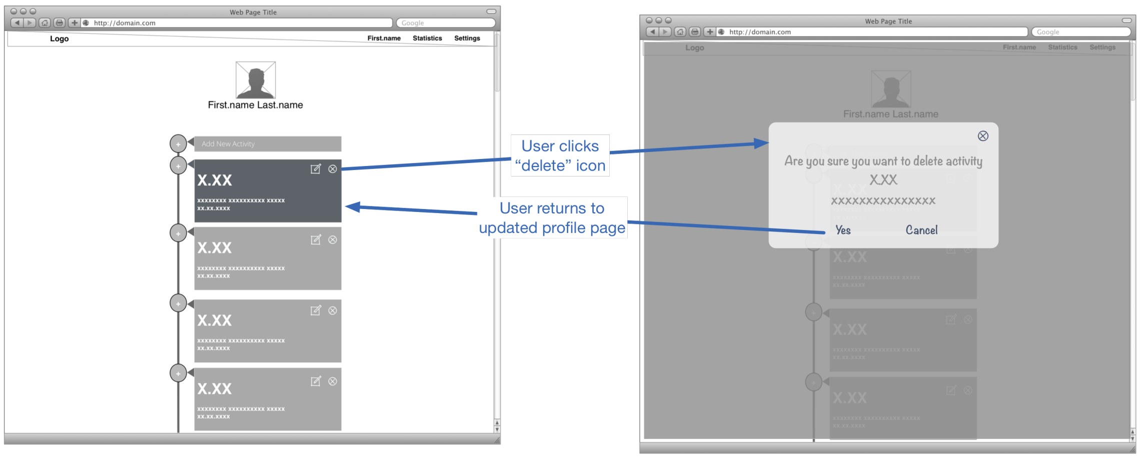


image 3.5: Delete Activity Wireframe

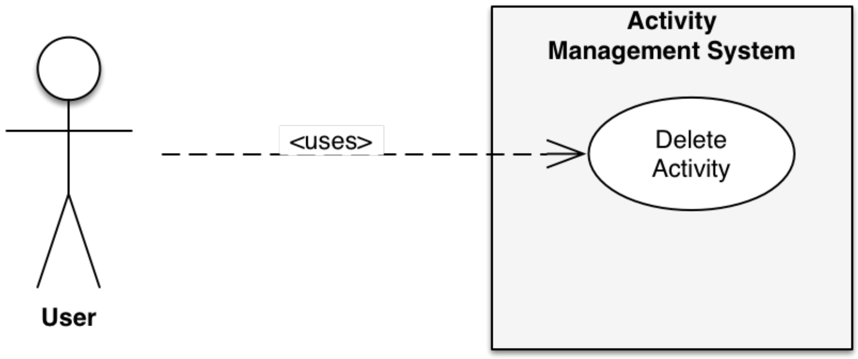


image 3.6: Delete Activity Use Case

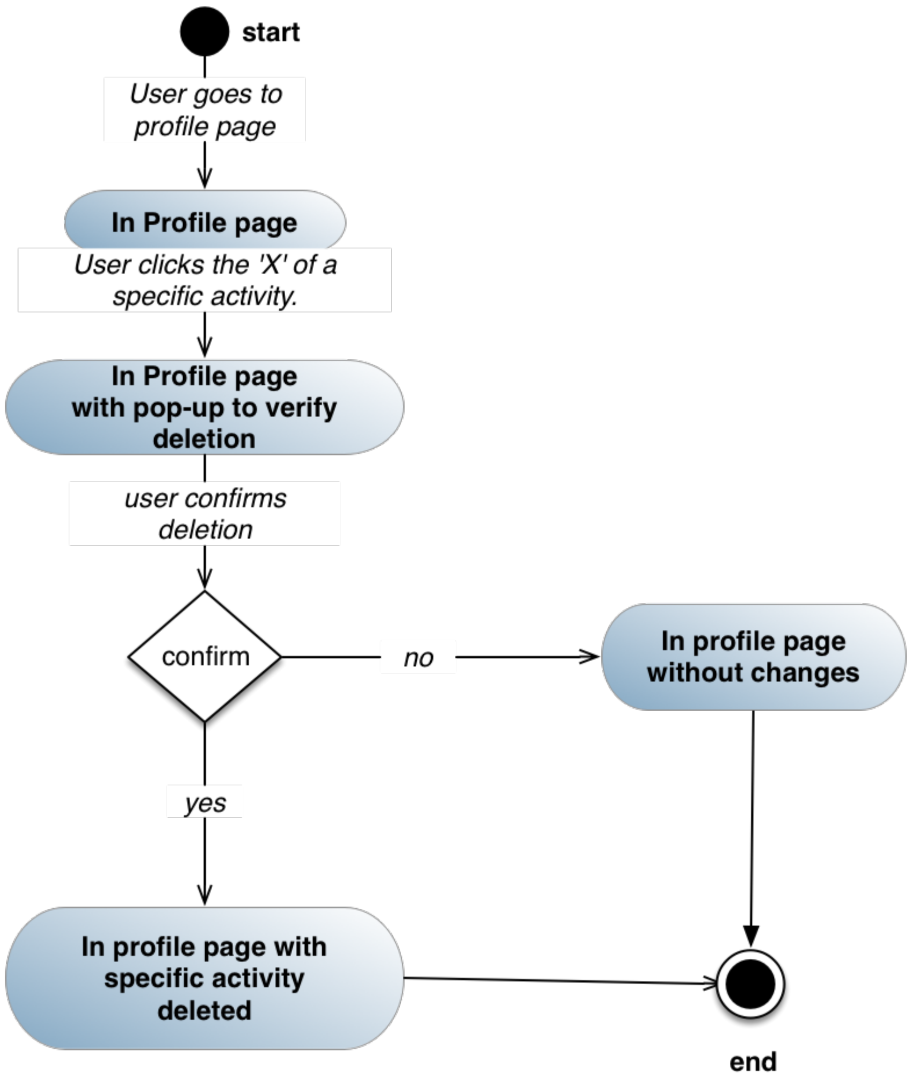

start
User goes to profile page
In Profile page
User clicks the 'X' of a specific activity.
In Profile page with pop-up to verify deletion
user confirms deletion
confirm
no
In profile page without changes
yes
In profile page with specific activity deleted
end

image 3.7: Delete Activity State Diagram

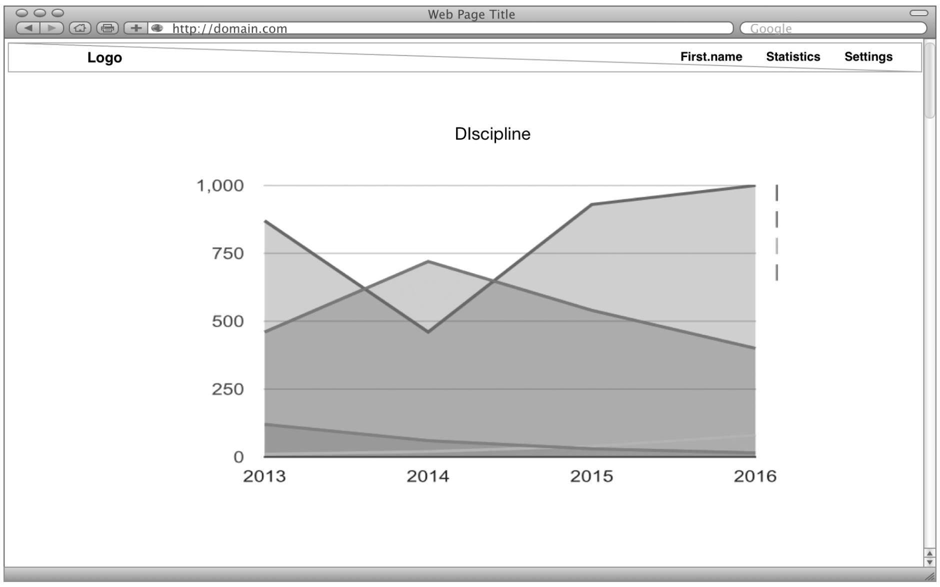


image 3.8: Statistics page User Interface

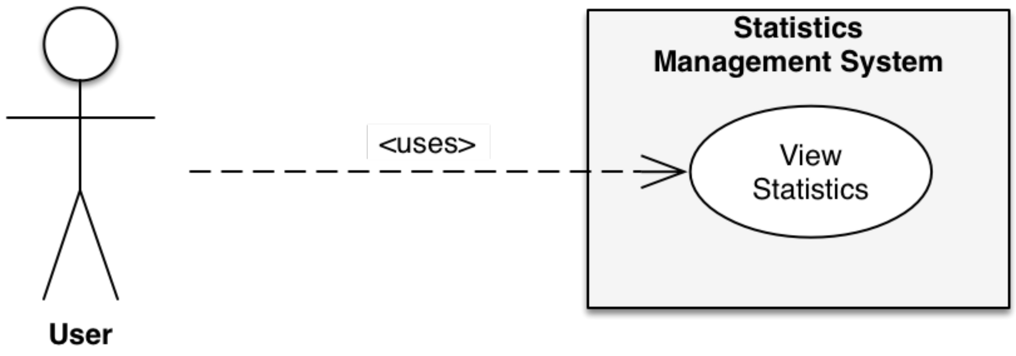


image 3.9: View Activities as graphs Use Case

6.3.4 Sprint review

Deleting and editing activities implemented on time. The same thing happened with our statistics page although there were difficulties in manipulating all these data in the client side.

In this sprint arouse the need of having a more structured application in the front-end, considering of introduce an MVC framework in the future for client-side, and remove the heavy logic of data manipulation and UI rendering from server side. We decided to implement also the settings page using the existing software stack and then reconsider if worthing to make a pivot and change the technologies we use.

6.4 Sprint 4

6.4.1 Sprint planning

It’s time to create a new section of the application where the user can personalise more their profile page, change his personal and contact data and also, in a later sprint, manage the privacy of their profile. This is the “settings page”

Inside the settings page the user can change their first and last name, a short description about itself and password. More options will be added in later sprint. Changes made to the above elements, will affect the data instantly and the UI will be updated accordingly.

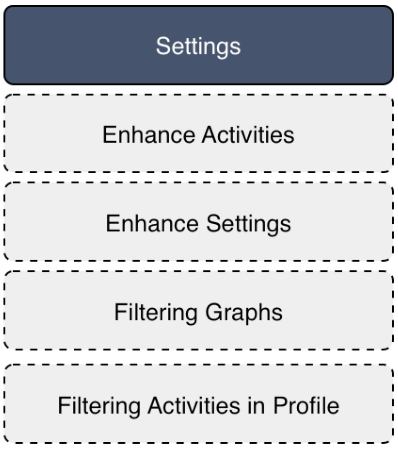


image 4.0: Backlog status for Sprint Planning of Sprint 4

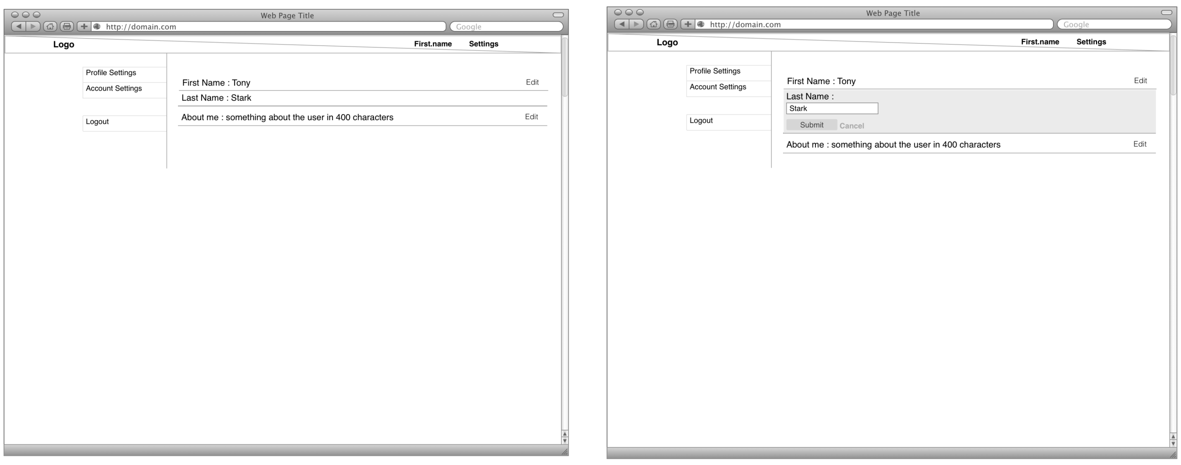
6.4.2 Stand up

The settings page is actually two forms with data that should always be validated by the server. New fields should be added to the database. Settings are separated in two different categories. Profile Settings and Account Settings.

Profile Settings tab will contain first and last name, and also the description. Account Settings tab will contain password change. All the data should be edited inline, on the form itself, made editable by clicking an icon, like the form shown and hide in edit activity. All data update request will be done with AJAX request which will add complexity and logic in client side.

Other than that it’s expected to implement this functionality without any special difficulties.

6.4.3 Diagrams



First Name : Tony
Last Name : Stark
About me : something about the user in 400 characters

First Name : Tony
Last Name :
Stark
About me : something about the user in 400 characters

image 4.1: User Interface for Settings page with and without editable field

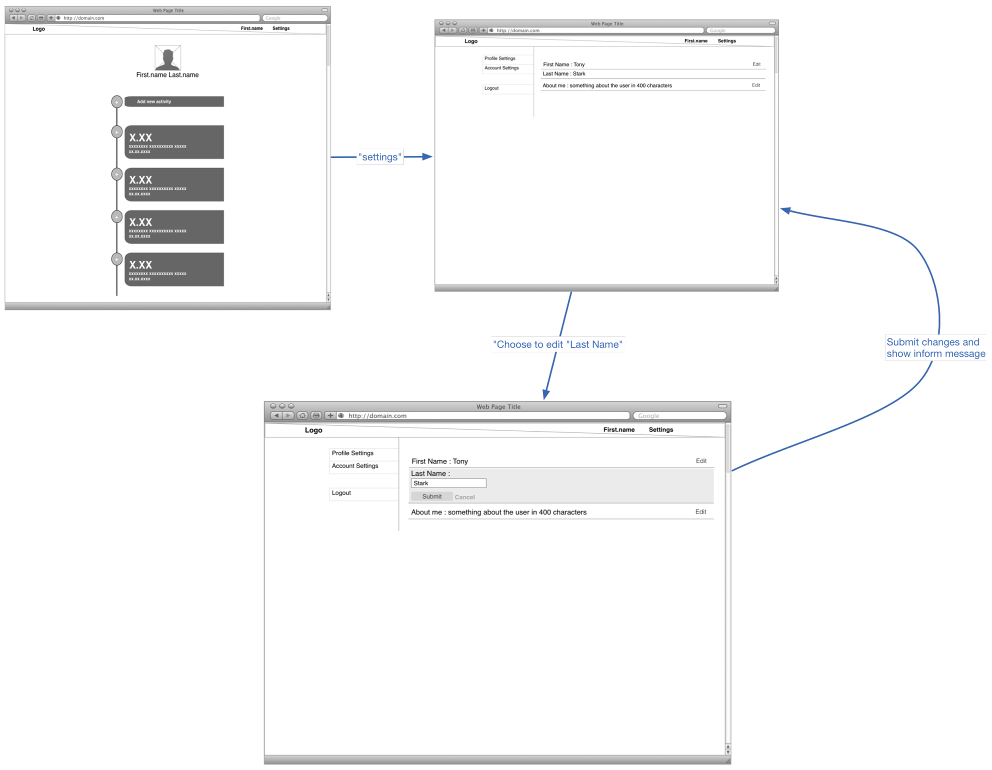


image 4.2: Navigate and Edit Activity Wireframe

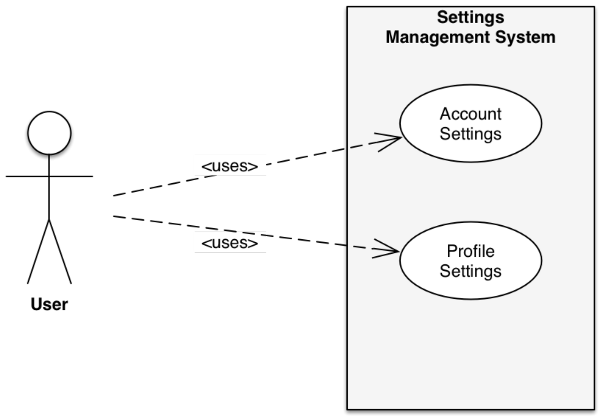


image 4.3: Account and Profile Settings Use Case

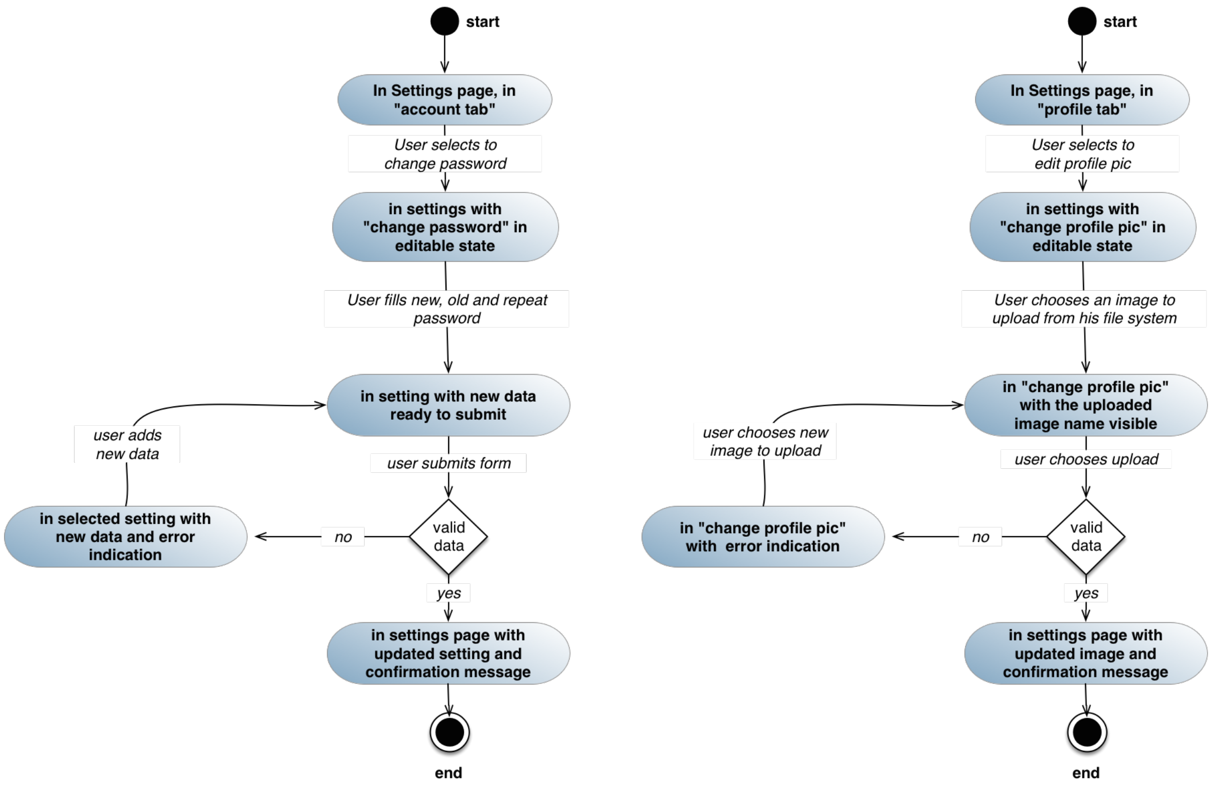

start
In Settings page, in "account tab"
User selects to change password
in settings with "change password" in editable state
User fills new, old and repeat password
in setting with new data ready to submit
user adds new data
user submits form
in selected setting with new data and error indication
no
valid data
yes
in settings page with updated setting and confirmation message
end
start
In Settings page, in "profile tab"
User selects to edit profile pic
in settings with "change profile pic" in editable state
User chooses an image to upload from his file system
in "change profile pic" with the uploaded image name visible
user chooses new image to upload
user chooses upload
in "change profile pic" with error indication
no
valid data
yes
in settings page with updated image and confirmation message
end

image 4.4: State diagrams for change a setting in account and profile tabs.

6.4.4 Sprint review

This Sprint had no special difficulties, but again we need to take the logic in the client side in order to make changes in UI without overloading the server with unnecessary work. One other thing that adds work load on the server is the conversion from different types of objects between our components. Client sends JSON objects to server, which manipulates the data in their internal object to be understandable from PHP and then converted again to JSONs for sending to mongoDB. For the above reasons we decided to replace PHP, on server side, with nodeJS. This change has some great advantages. First of all it will give us the consistency in using json objects end-to-end. Secondly nodeJS is more scalable than PHP and it will give us a technological advantage when the need for more computation power will arise. An other advantage that nodeJS will give us is that it will allow us to use only one programming language in our app. This is great for a small team like ours. We can easily work in full stack and avoid rewriting code and validations in both sides, for different languages. So the following two Sprints will have changes mainly in server side and some UI enhancements in order to follow the changed UI and logic that served from back-end.

6.5 Sprint 5

6.5.1 Description

In this Sprint we decided to move from PHP to node.js at server side. The reasons we choose nodeJS have been described in Chapter 2. In this section I'm going to describe a little more about the decision to abandon PHP over node.js. First of all, mixing code with content is a thing that can end up adding a lot of complexity. Although it's fun to mix code in with HTML, soon your code base becomes a tangled mess of logic. As developers we need to add structure and separate the cosmetic layer from the logical layer. It's cleaner for new programmers to understand the existed code and easier to maintain it in the future. The frameworks running on Node.js are built based on the logic that is better view, and controller layers to be separate.

While AJAX-based HTML5 Web apps can have too many moving parts, they are very efficient. Once the JavaScript code is in the browser cache, the only thing that moves along the wires is the new data. There's not a ton of HTML markup, and there are no repeated trips to download the entire page. Only the data has changed. Node.js is optimised to deliver the data and only the data through Web services. If your app is complex and data-rich, it's a good foundation for efficient delivery. Node.js also speaks JSON, the lingua franca for interacting with many of the latest NoSQL databases (like mongoDB we use). Of course we could use mongoDB with PHP stack, but it's much more fluid and simple to work with JSON when using JavaScript. It's one syntax from browser to Web server to database. The colons and the curly brackets work the same way everywhere. That alone will save us from hours of frustration. The callback mechanism is brilliant because it saves us from implementing, handling and managing multiple threads. The core is well-built and designed to do all that for us.

6.5.2 Sprint planning

In this and next Sprints the main job we have to do is to transform our backend, replacing PHP implementation with one of node.js. Having already implement a REST API on server side allow us to complete change the backend with only some changes in web client and its request-response logic. Working as team, and since this Sprint doesn't have a big work load for my technical part, allow me to work on backend. Since User Interface's decisions and design is my job I worked on the part on the backend that affects the HTML templating. The difference that affects my job is that now the HTML templates are build with Jade, which means that we have to implement again all the pages and embed inside them the data that come from node.js

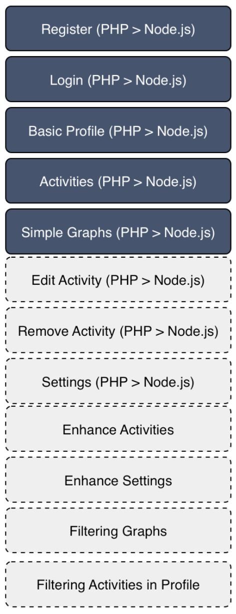
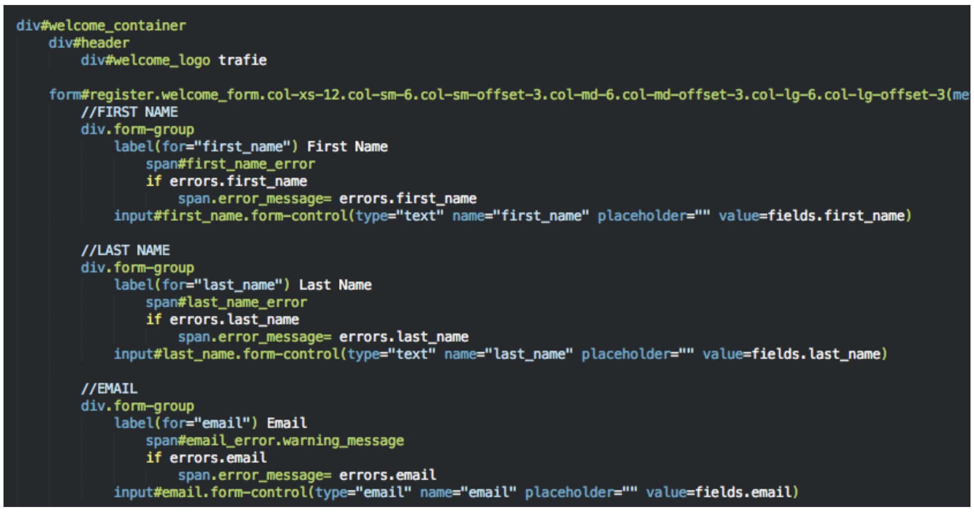

Register (PHP > Node.js)
Login (PHP > Node.js)
Basic Profile (PHP > Node.js)
Activities (PHP > Node.js)
Simple Graphs (PHP > Node.js)
Edit Activity (PHP > Node.js)
Remove Activity (PHP > Node.js)
Settings (PHP > Node.js)
Enhance Activities
Enhance Settings
Filtering Graphs
Filtering Activities in Profile

image 5.0: Backlog status for Sprint Planning of Sprint 5

6.5.3 Stand up

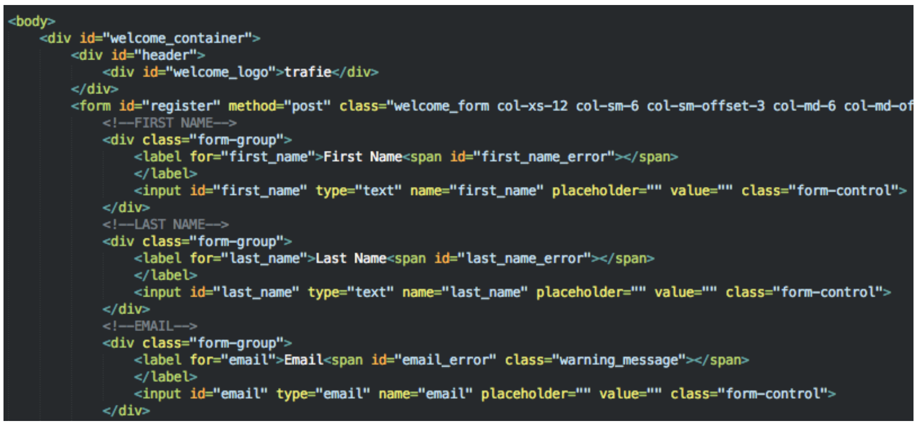
In JADE we create different types of document in comparison with HTML, which in continue are translated into HTML. What we have to do is to convert all pages in JADE's format. Our



```
div#welcome_container
    div#header
        div#welcome_logo trafie

    form#register.welcome_form.col-xs-12.col-sm-6.col-sm-offset-3.col-md-6.col-md-offset-3.col-lg-6.col-lg-offset-3(me
        //FIRST NAME
        div.form-group
            label(for="first_name") First Name
                span#first_name_error
                if errors.first_name
                    span.error_message= errors.first_name
            input#first_name.form-control(type="text" name="first_name" placeholder="" value=fields.first_name)

        //LAST NAME
        div.form-group
            label(for="last_name") Last Name
                span#last_name_error
                if errors.last_name
                    span.error_message= errors.last_name
            input#last_name.form-control(type="text" name="last_name" placeholder="" value=fields.last_name)

        //EMAIL
        div.form-group
            label(for="email") Email
                span#email_error.warning_message
                if errors.email
                    span.error_message= errors.email
            input#email.form-control(type="email" name="email" placeholder="" value=fields.email)
```

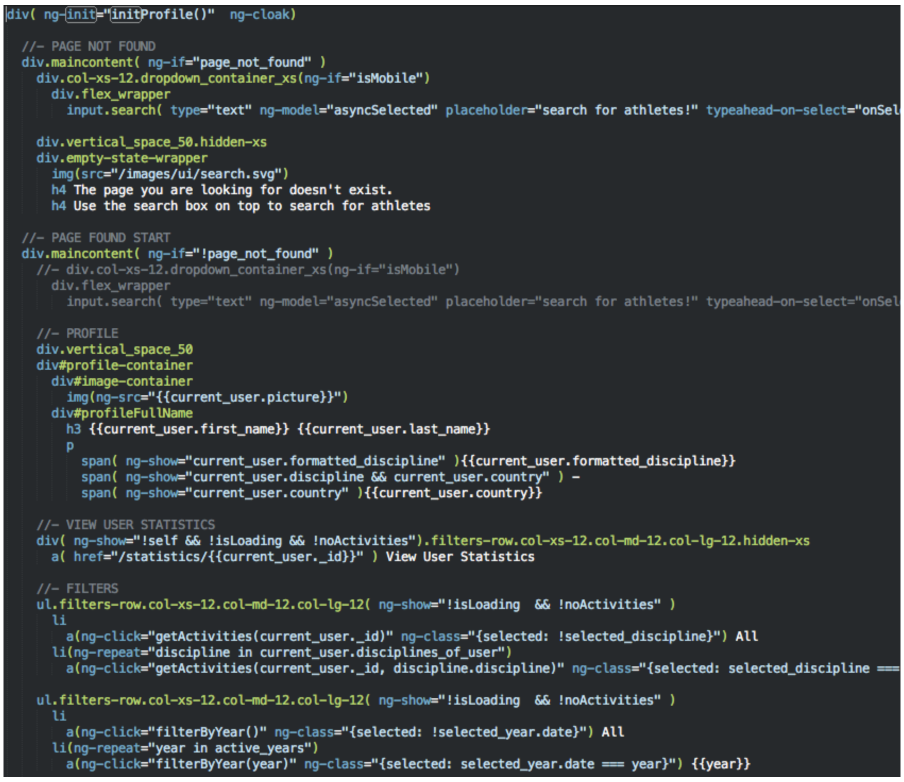
image 5.1: Register form in JADE template

templates are now easier to read and edit, without messing with all the HTML tags. Following two screenshots of the same part of User Interface in HTML and JADE.



```
<body>
    <div id="welcome_container">
        <div id="header">
            <div id="welcome_logo">trafie</div>
        </div>
        <form id="register" method="post" class="welcome_form col-xs-12 col-sm-6 col-sm-offset-3 col-md-6 col-md-of
            <!--FIRST NAME-->
            <div class="form-group">
                <label for="first_name">First Name<span id="first_name_error"></span>
                </label>
                <input id="first_name" type="text" name="first_name" placeholder="" value="" class="form-control">
            </div>
            <!--LAST NAME-->
            <div class="form-group">
                <label for="last_name">Last Name<span id="last_name_error"></span>
                </label>
                <input id="last_name" type="text" name="last_name" placeholder="" value="" class="form-control">
            </div>
            <!--EMAIL-->
            <div class="form-group">
                <label for="email">Email<span id="email_error" class="warning_message"></span>
                </label>
                <input id="email" type="email" name="email" placeholder="" value="" class="form-control">
            </div>
```

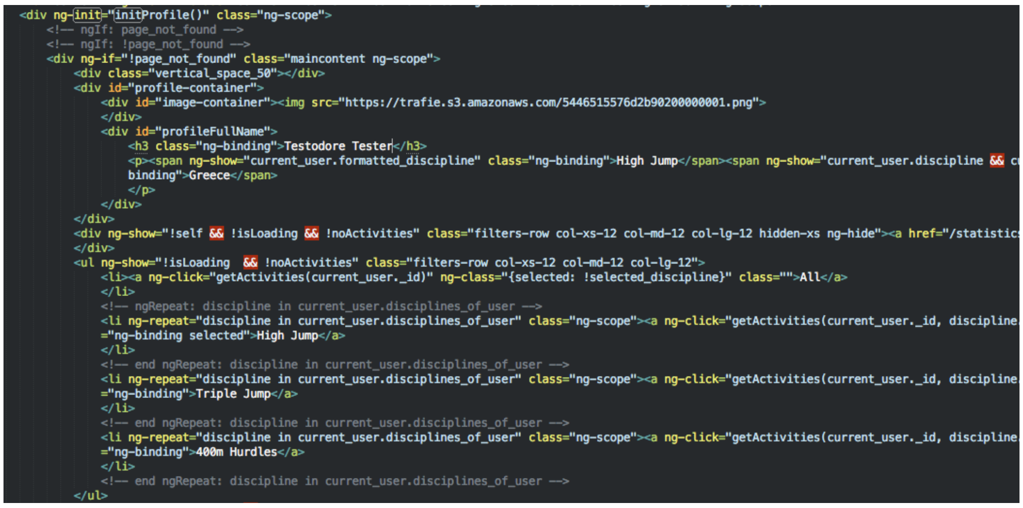
image 5.2: Register form in classic HTML



```
div( ng-init="initProfile()"  ng-cloak)

  //- PAGE NOT FOUND
  div.maincontent( ng-if="page_not_found" )
    div.col-xs-12.dropdown_container_xs(ng-if="isMobile")
      div.flex_wrapper
        input.search( type="text" ng-model="asyncSelected" placeholder="search for athletes!" typeahead-on-select="onSel

    div.vertical_space_50.hidden-xs
    div.empty-state-wrapper
      img(src="/images/ui/search.svg")
      h4 The page you are looking for doesn't exist.
      h4 Use the search box on top to search for athletes

  //- PAGE FOUND START
  div.maincontent( ng-if="!page_not_found" )
    //- div.col-xs-12.dropdown_container_xs(ng-if="isMobile")
      div.flex_wrapper
        input.search( type="text" ng-model="asyncSelected" placeholder="search for athletes!" typeahead-on-select="onSel

    //- PROFILE
    div.vertical_space_50
    div#profile-container
      div#image-container
        img(ng-src="{{current_user.picture}}")
      div#profileFullName
        h3 {{current_user.first_name}} {{current_user.last_name}}
        p
          span( ng-show="current_user.formatted_discipline" ){{current_user.formatted_discipline}}
          span( ng-show="current_user.discipline && current_user.country" ) -
          span( ng-show="current_user.country" ){{current_user.country}}

    //- VIEW USER STATISTICS
    div( ng-show="!self && !isLoading && !noActivities").filters-row.col-xs-12.col-md-12.col-lg-12.hidden-xs
      a( href="/statistics/{{current_user._id}}" ) View User Statistics

    //- FILTERS
    ul.filters-row.col-xs-12.col-md-12.col-lg-12( ng-show="!isLoading  && !noActivities" )
      li
        a(ng-click="getActivities(current_user._id)" ng-class="{selected: !selected_discipline}") All
      li(ng-repeat="discipline in current_user.disciplines_of_user")
        a(ng-click="getActivities(current_user._id, discipline.discipline)" ng-class="{selected: selected_discipline ===

    ul.filters-row.col-xs-12.col-md-12.col-lg-12( ng-show="!isLoading  && !noActivities" )
      li
        a(ng-click="filterByYear()" ng-class="{selected: !selected_year.date}") All
      li(ng-repeat="year in active_years")
        a(ng-click="filterByYear(year)" ng-class="{selected: selected_year.date === year}") {{year}}
```

image 5.3: Part of Profile page in JADE template

More specific, in this Sprint we tried to transfer our basic pages in our new logic. Our initial point is the registration and login page where our users have the first contact with our app. We also updated the basic profile and the simple graphs page. All these pages are the same as described in Sprint 2 and 3. Thus we don’t need to describe them again. Following one more example of our pages in JADE and HTML versions



```
<div ng-init="initProfile()" class="ng-scope">
    <!-- ngIf: page_not_found -->
    <!-- ngIf: !page_not_found -->
    <div ng-if="!page_not_found" class="maincontent ng-scope">
        <div class="vertical_space_50"></div>
        <div id="profile-container">
            <div id="image-container"><img src="https://trafie.s3.amazonaws.com/5446515576d2b90200000001.png">
            </div>
            <div id="profileFullName">
                <h3 class="ng-binding">Testodore Tester</h3>
                <p><span ng-show="current_user.formatted_discipline" class="ng-binding">High Jump</span><span ng-show="current_user.discipline && c
                binding">Greece</span>
                </p>
            </div>
        </div>
        <div ng-show="!self && !isLoading && !noActivities" class="filters-row col-xs-12 col-md-12 col-lg-12 hidden-xs ng-hide"><a href="/statistic
        </div>
        <ul ng-show="!isLoading  && !noActivities" class="filters-row col-xs-12 col-md-12 col-lg-12">
            <li><a ng-click="getActivities(current_user._id)" ng-class="{selected: !selected_discipline}" class="">All</a>
            </li>
            <!-- ngRepeat: discipline in current_user.disciplines_of_user -->
            <li ng-repeat="discipline in current_user.disciplines_of_user" class="ng-scope"><a ng-click="getActivities(current_user._id, discipline
            ="ng-binding selected">High Jump</a>
            </li>
            <!-- end ngRepeat: discipline in current_user.disciplines_of_user -->
            <li ng-repeat="discipline in current_user.disciplines_of_user" class="ng-scope"><a ng-click="getActivities(current_user._id, discipline
            ="ng-binding">Triple Jump</a>
            </li>
            <!-- end ngRepeat: discipline in current_user.disciplines_of_user -->
            <li ng-repeat="discipline in current_user.disciplines_of_user" class="ng-scope"><a ng-click="getActivities(current_user._id, discipline
            ="ng-binding">400m Hurdles</a>
            </li>
            <!-- end ngRepeat: discipline in current_user.disciplines_of_user -->
        </ul>
```

image 5.4: Part of Profile page in classic HTML

6.5.4 Diagrams

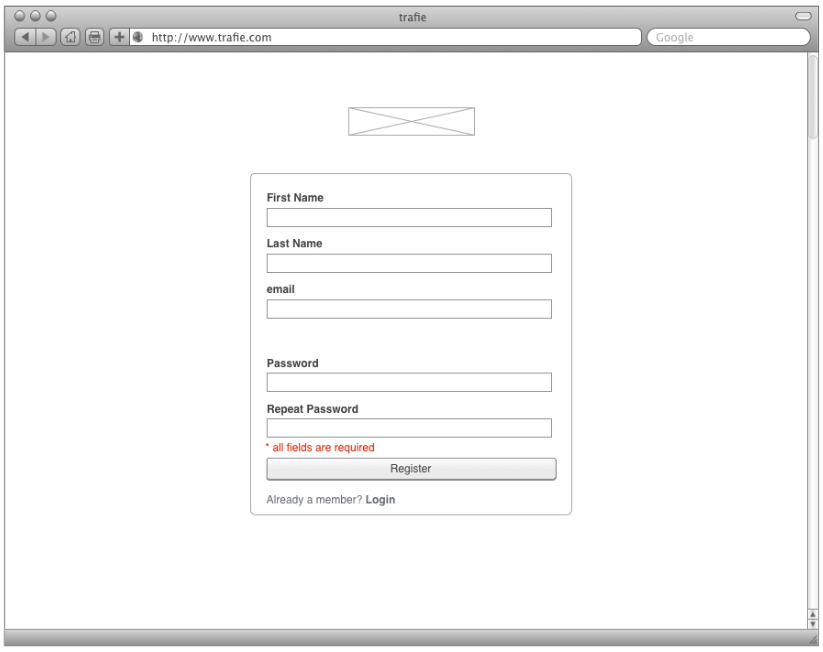


image 5.5: Register form User Interface

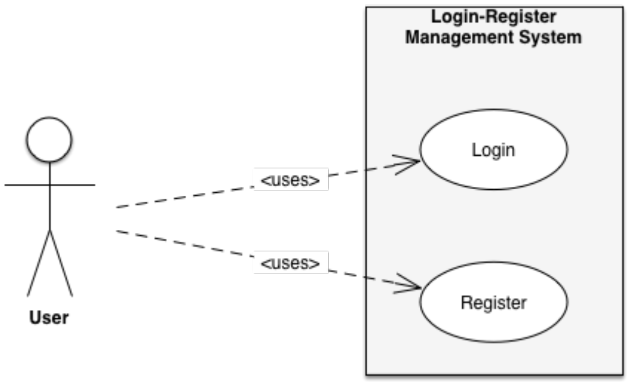


image 5.6: Register and Login Use Case

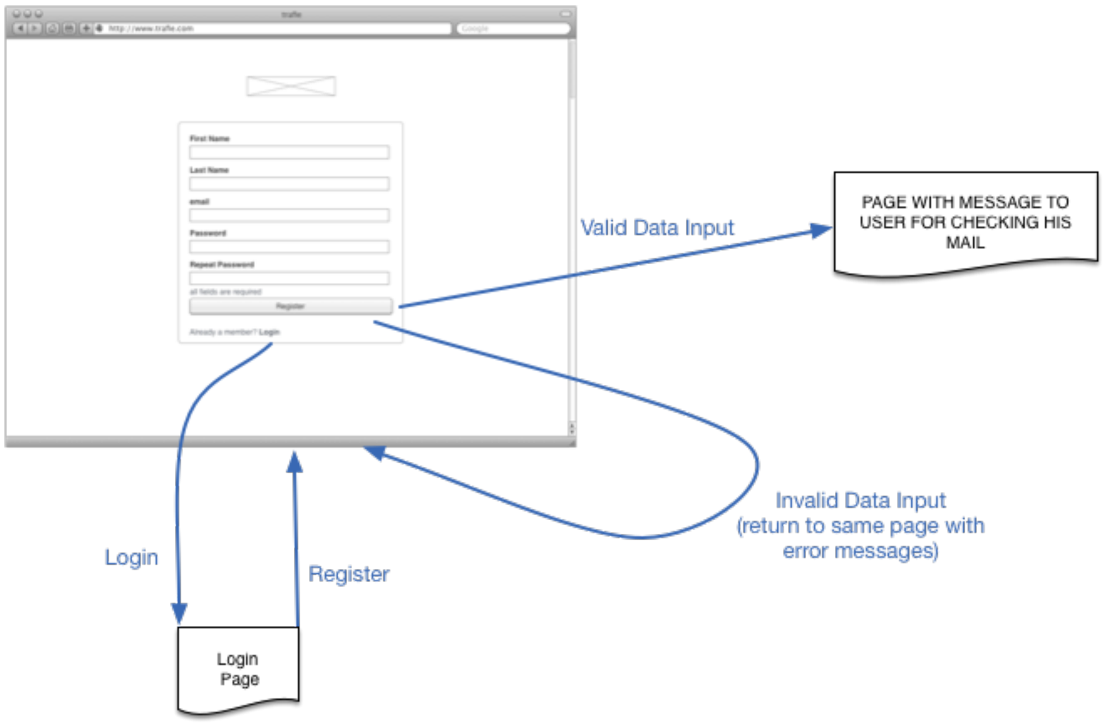


image 5.7: Register page Wireframe

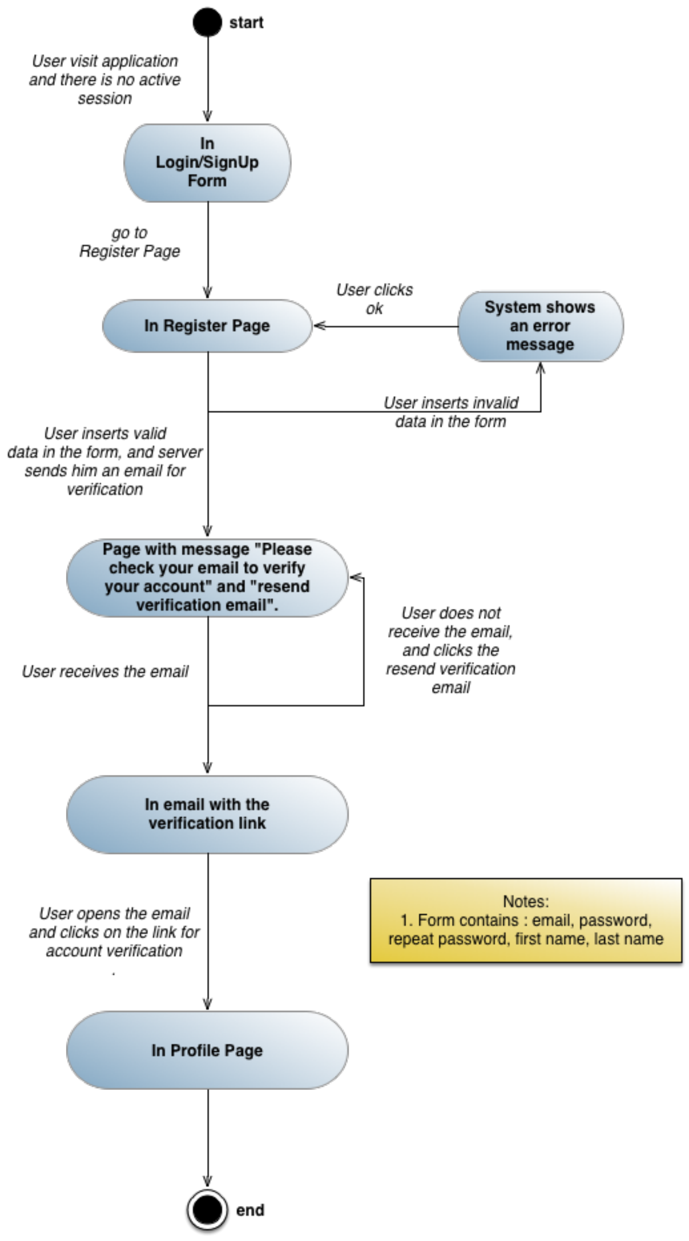

start
User visit application and there is no active session
In Login/SignUp Form
go to Register Page
In Register Page
User clicks ok
System shows an error message
User inserts invalid data in the form
User inserts valid data in the form, and server sends him an email for verification
Page with message "Please check your email to verify your account" and "resend verification email".
User does not receive the email, and clicks the resend verification email
User receives the email
In email with the verification link
User opens the email and clicks on the link for account verification
Notes:
1. Form contains : email, password, repeat password, first name, last name
In Profile Page
end

image 5.8: Register page State Diagram

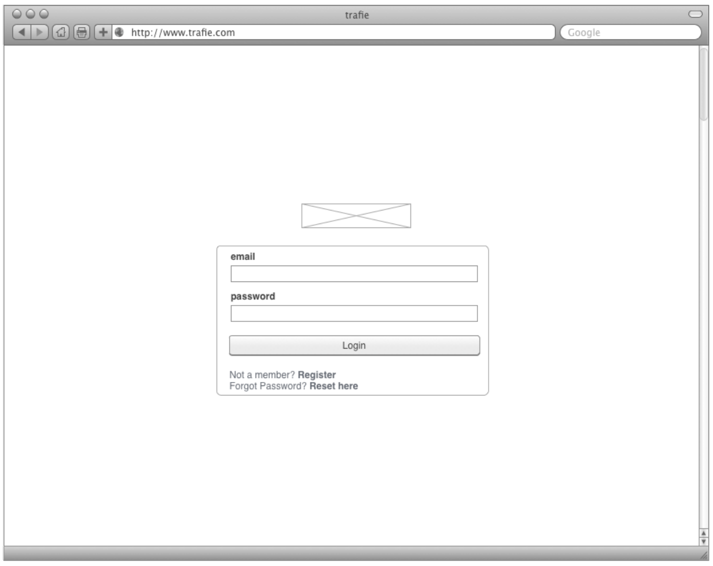


image 5.9: Login Page User Interface

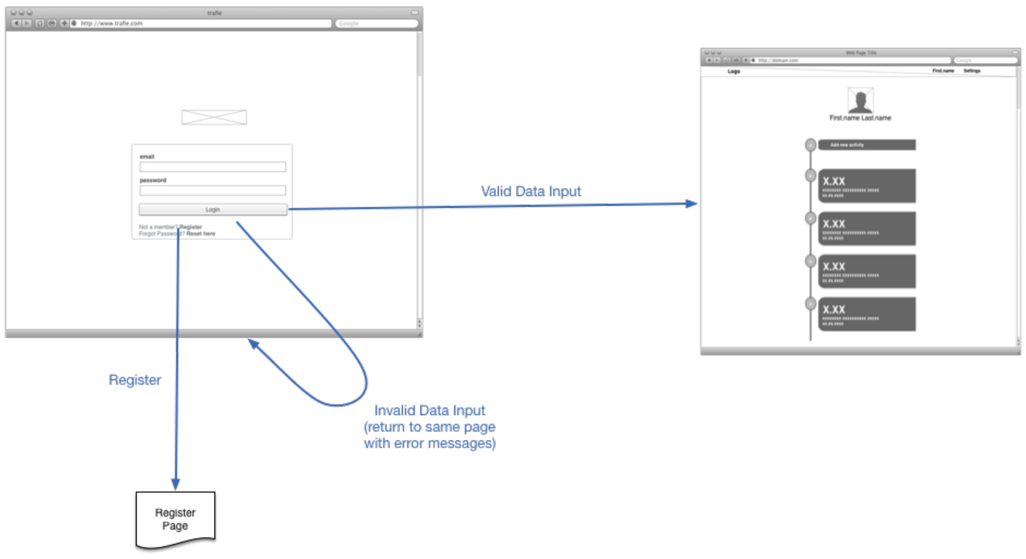


image 5.10: Login Wireframe

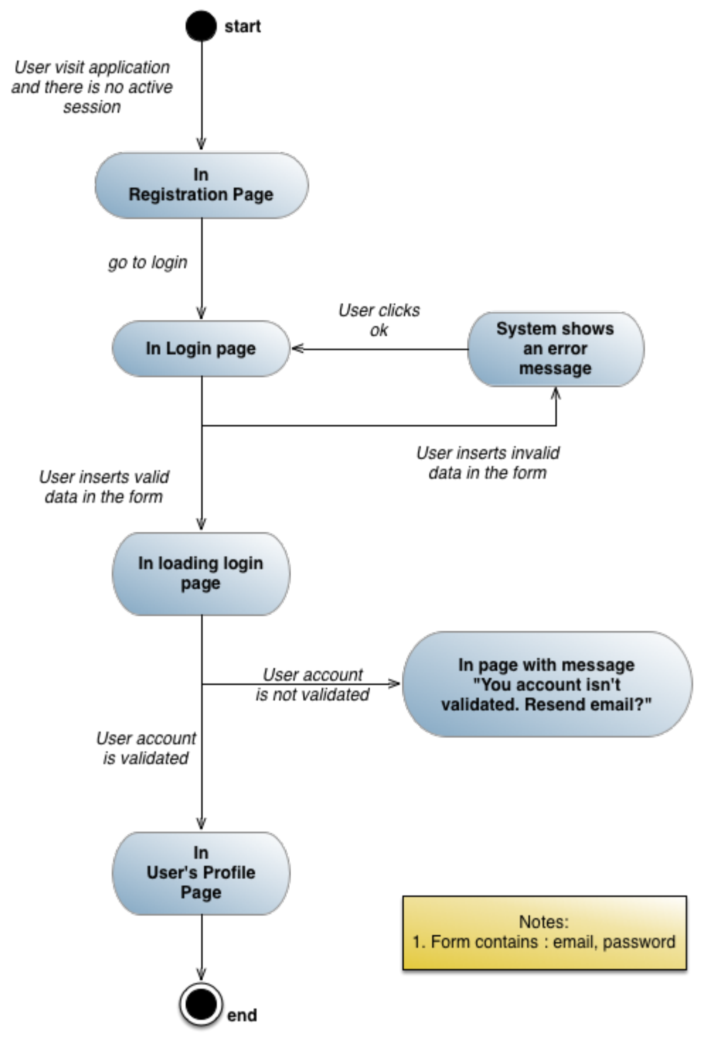

start
User visit application and there is no active session
In Registration Page
go to login
In Login page
User clicks ok
System shows an error message
User inserts invalid data in the form
User inserts valid data in the form
In loading login page
User account is not validated
In page with message "You account isn't validated. Resend email?"
User account is validated
In User's Profile Page
Notes:
1. Form contains : email, password
end

image 5.11: Login State Diagram

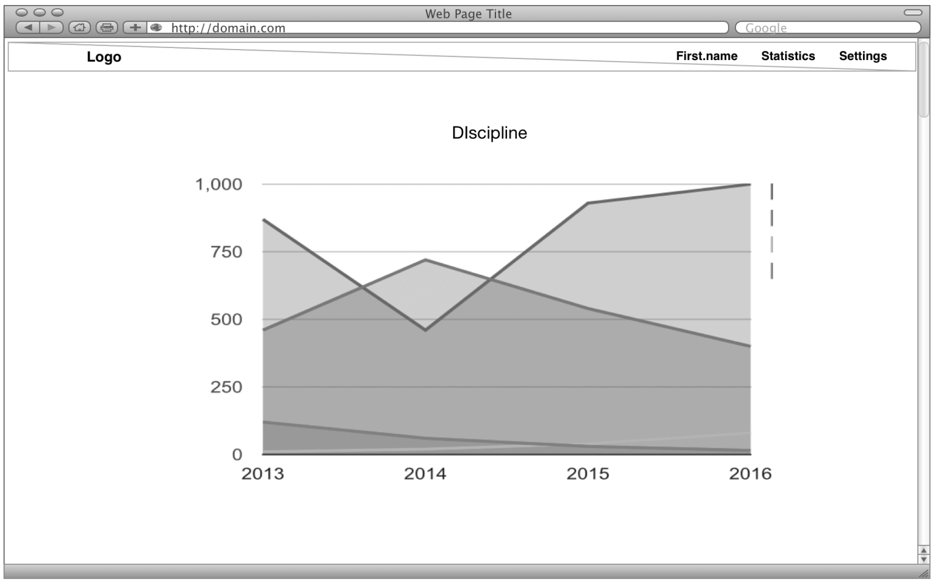


image 5.12: Statistics page User Interface

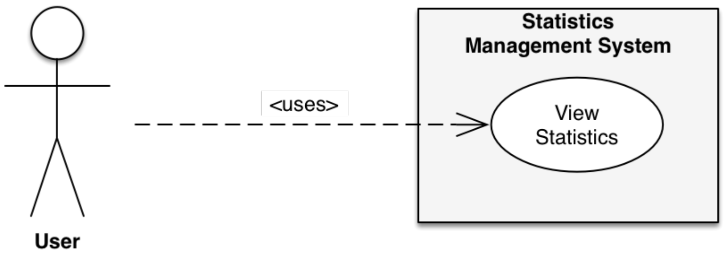


image 5.13: View Activities as graphs Use Case

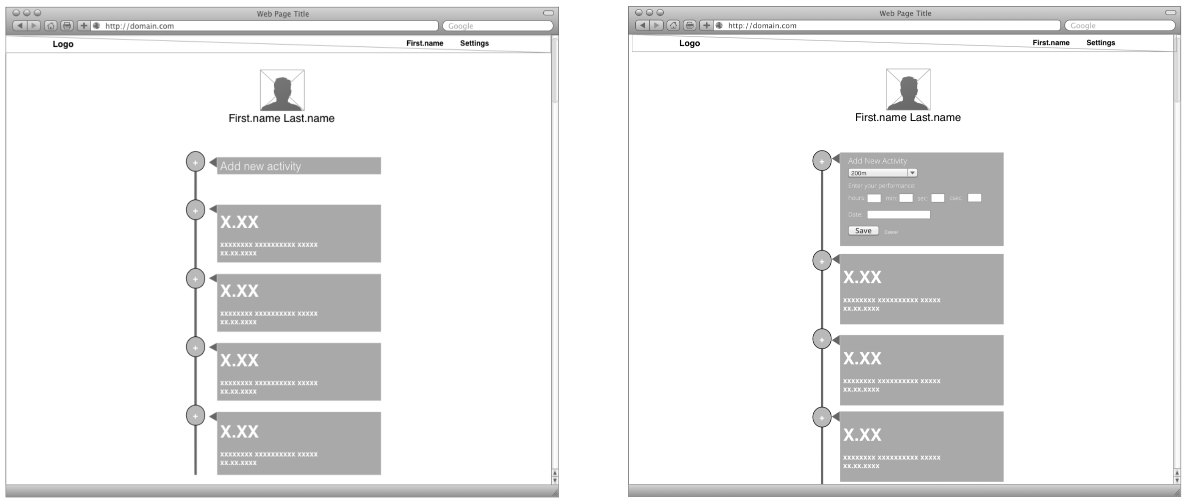


image 5.14: Profile Page in normal state and with add activity form open

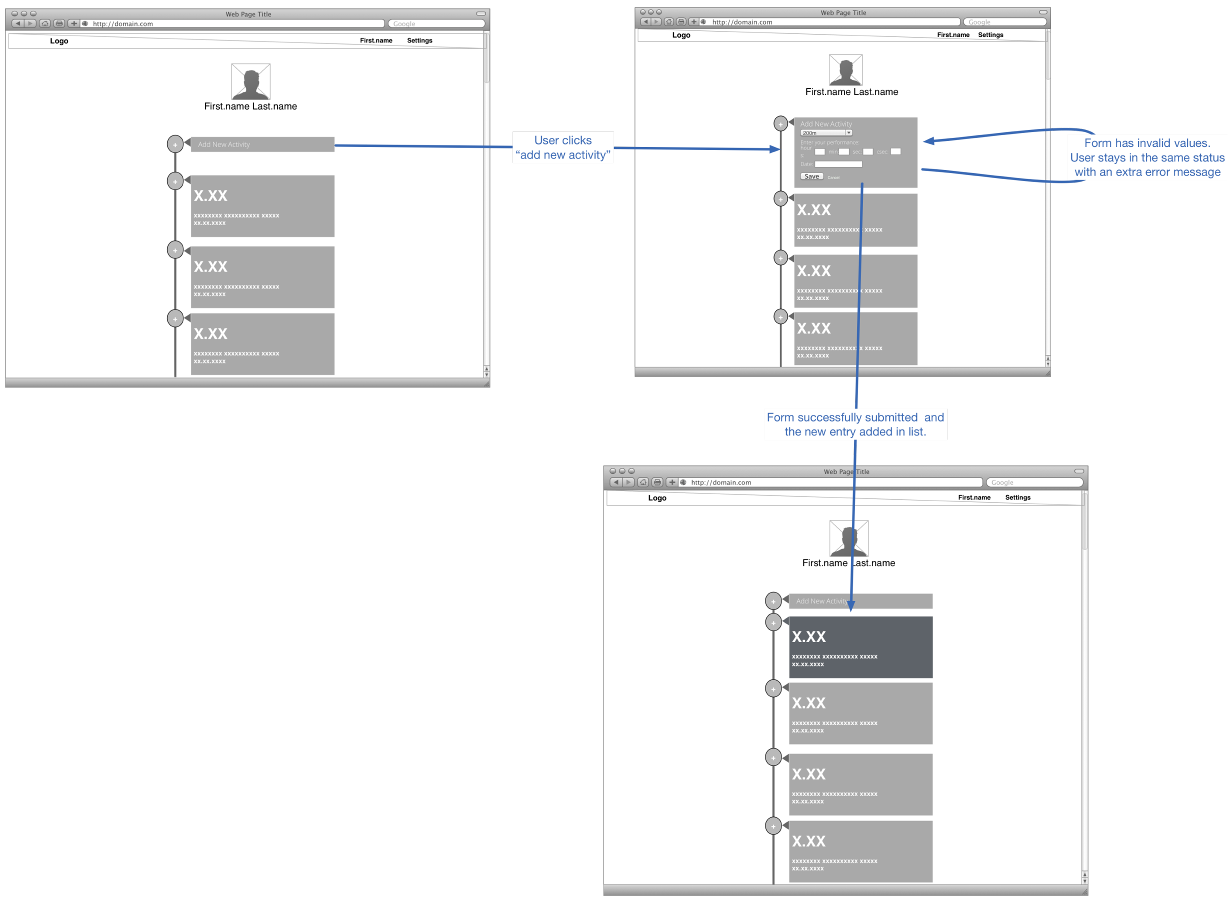


image 5.15: Add activity Wireframe

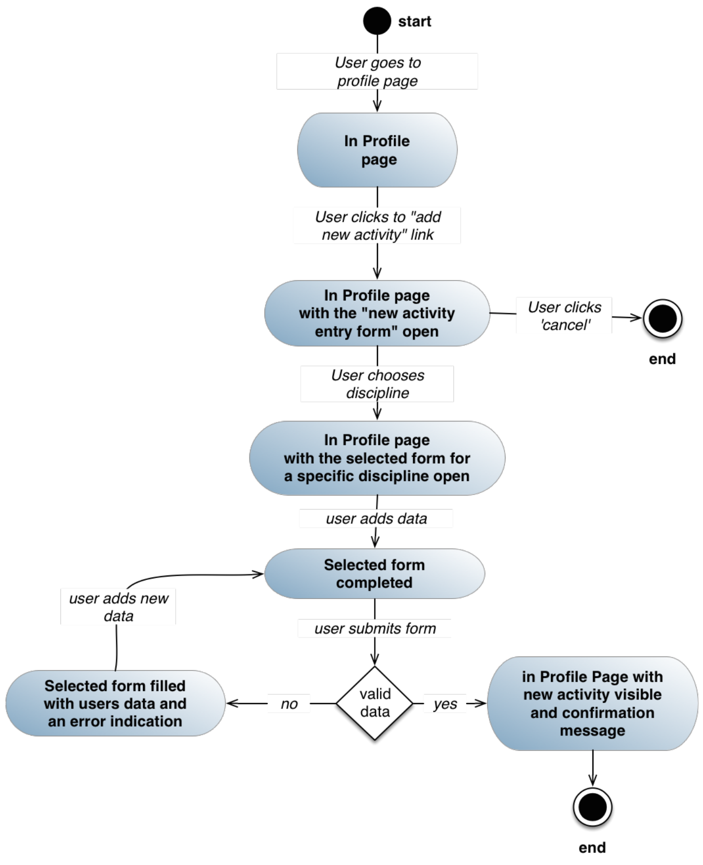

start
User goes to profile page
In Profile page
User clicks to "add new activity" link
In Profile page with the "new activity entry form" open
User clicks 'cancel'
end
User chooses discipline
In Profile page with the selected form for a specific discipline open
user adds data
Selected form completed
user adds new data
user submits form
Selected form filled with users data and an error indication
no
valid data
yes
in Profile Page with new activity visible and confirmation message
end

image 5.16: Add Activity State Diagram

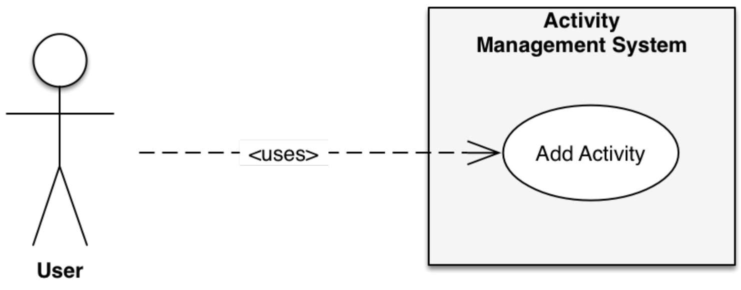


image 5.17: Add Activity Use Case

6.5.5 Sprint review

This sprint took more time than expected, because me and my colleague were learning new frameworks and libraries that used for the implementation. There were delays because we needed time to become familiar with a new end-to-end stack of software, with a non-blocking style on server-side. We weren't familiar with nodeJS, express, Google Charts and MongoDB database. In the end, all the required features were implemented. This is a good pivot point for the project, because there are a lot of advantages we gain from these changes.

6.6 Sprint 6

6.6.1 Sprint planning

In this Sprint we will try to complete the transformation of the server side. The functionality that is missing is edit and remove activity, and also the settings. We also decided to add more information for the user in this page. So now user's profile has first name, last name, biography , main discipline, country, profile pic and password. Each field should have its own validation rules as before.

The main discipline should have the same accepted values as the disciplines in the profile page. There should be a list of valid countries in the server, that will be used in order to validate the user's country choice. The password should be validated based on the validation rules that were used during the registration process and there should be a second field, ensuring that the user has entered the password that they intended to use.

Uploading a profile picture is a different process than posting the rest of the data in the form. There should be several steps that should be followed during the upload process. First the MIME type and the file extension should be checked, in order to verify the file type. Only jpeg and png files should be accepted. Then the file size should be checked. If the image is too big, it should be rejected. The user is allowed to have only one profile picture, so if there is another picture posted by them, it should be replaced. Then the new file should be renamed to be the user's id, for fast file retrieval. Then the image should be copied to the file system of the server, the path to the file should be stored in the database, in the profile table and a json object should be sent back to the client, stating the the process was completed successfully. If something went wrong, a detailed error message should be sent to the client. When a new profile picture is uploaded, the image preview and the image in the profile should be refreshed, and the new image should be shown.

There is no need to validate the user in the settings page, as the users can not view the settings page of the others users. The user_id of the values is automatically set to the id of the user in the session. After all the validations, messages should be sent stating if the process was completed successfully or not.

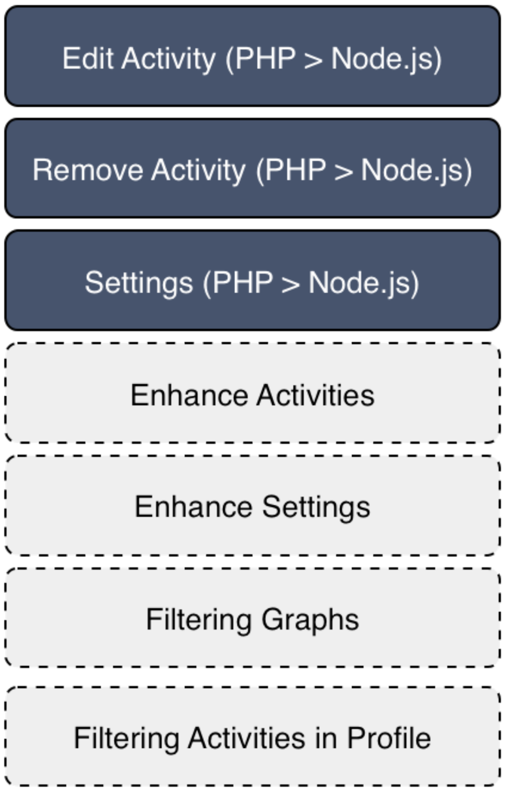

Edit Activity (PHP > Node.js)
Remove Activity (PHP > Node.js)
Settings (PHP > Node.js)
Enhance Activities
Enhance Settings
Filtering Graphs
Filtering Activities in Profile

image 6.0: Backlog status for Planning of Sprint 6

6.6.2 Stand up

There are not a lot of things to mention here because this Sprint didn't have special difficulties. After Sprint5 we already had overcome the problems and have the knowledge on managing these kind of issues.

6.6.3 Diagrams

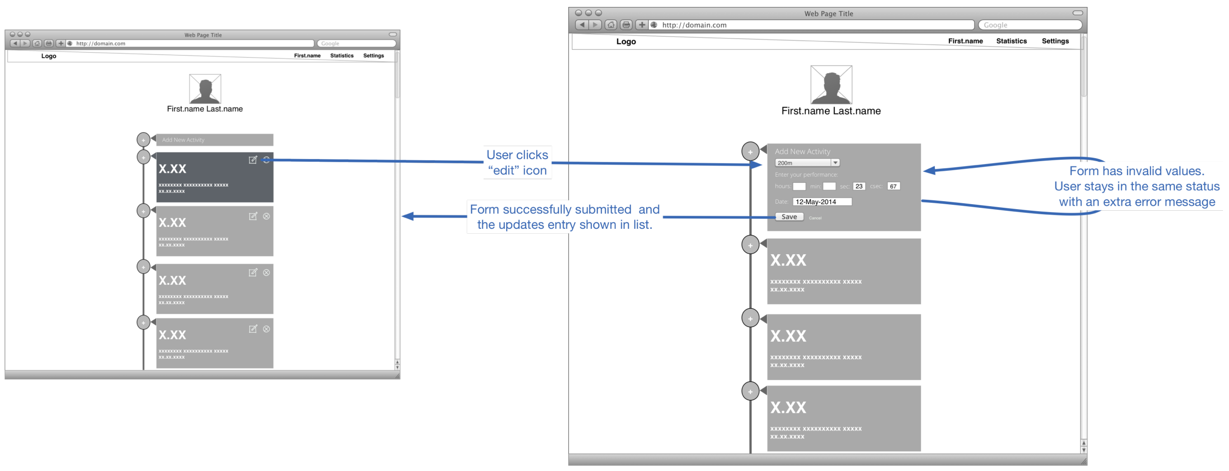


image 6.1: Edit Activity Wireframe

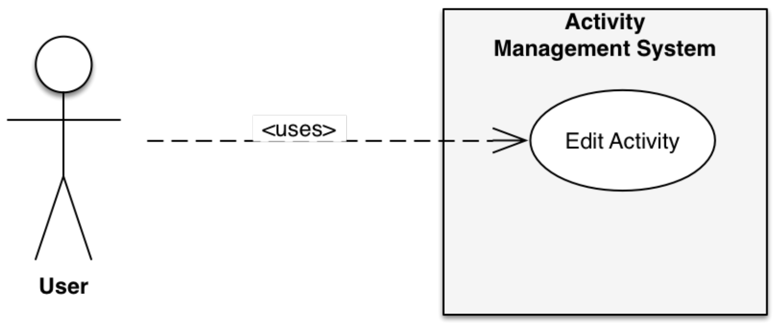


image 6.2: Edit Activity Use Case

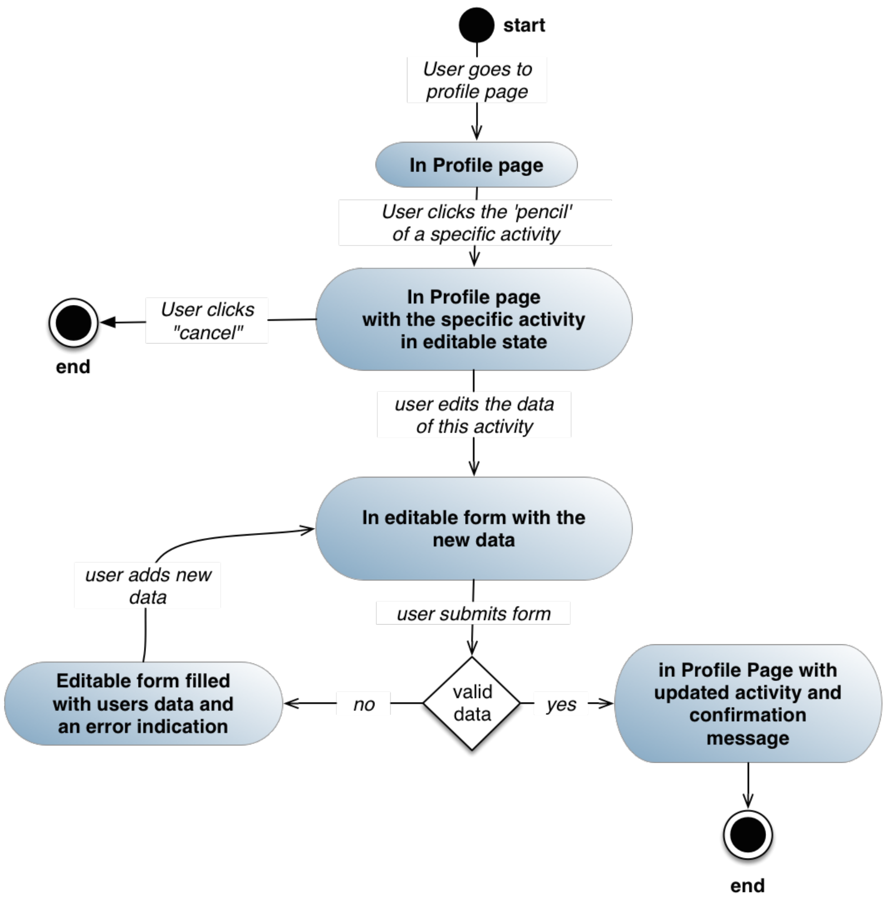

start
User goes to profile page
In Profile page
User clicks the 'pencil' of a specific activity
In Profile page with the specific activity in editable state
User clicks "cancel"
end
user edits the data of this activity
In editable form with the new data
user adds new data
user submits form
valid data
no
yes
Editable form filled with users data and an error indication
in Profile Page with updated activity and confirmation message
end

image 6.3: Edit Activity State Diagram

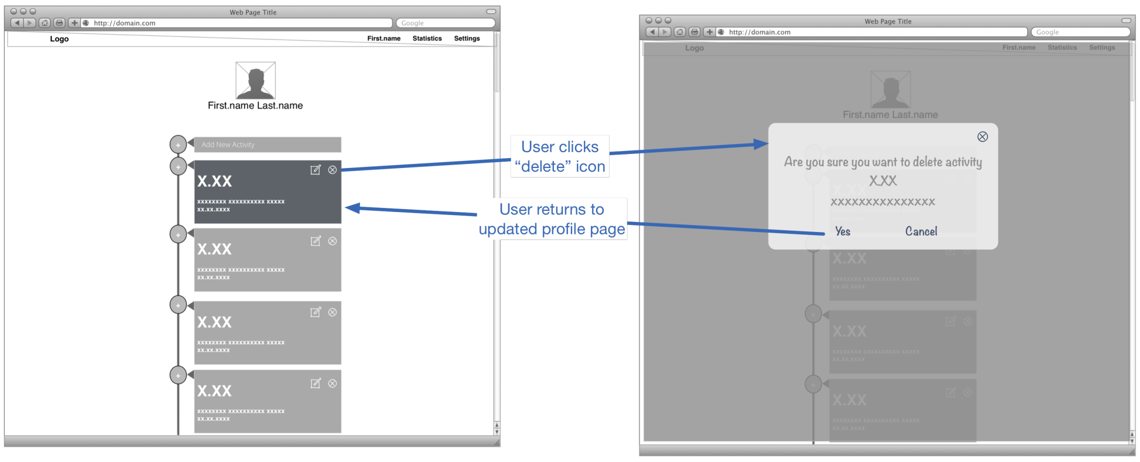


image 6.4: Delete Activity Wireframe

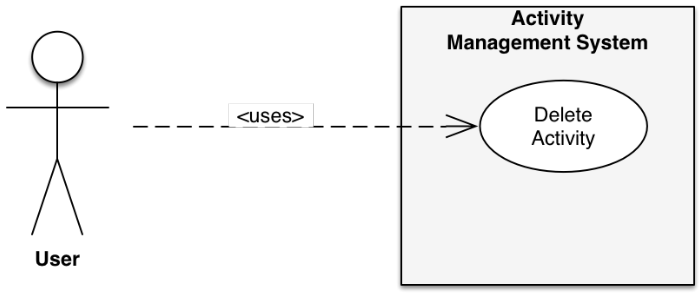


image 6.5: Delete Activity Use Case

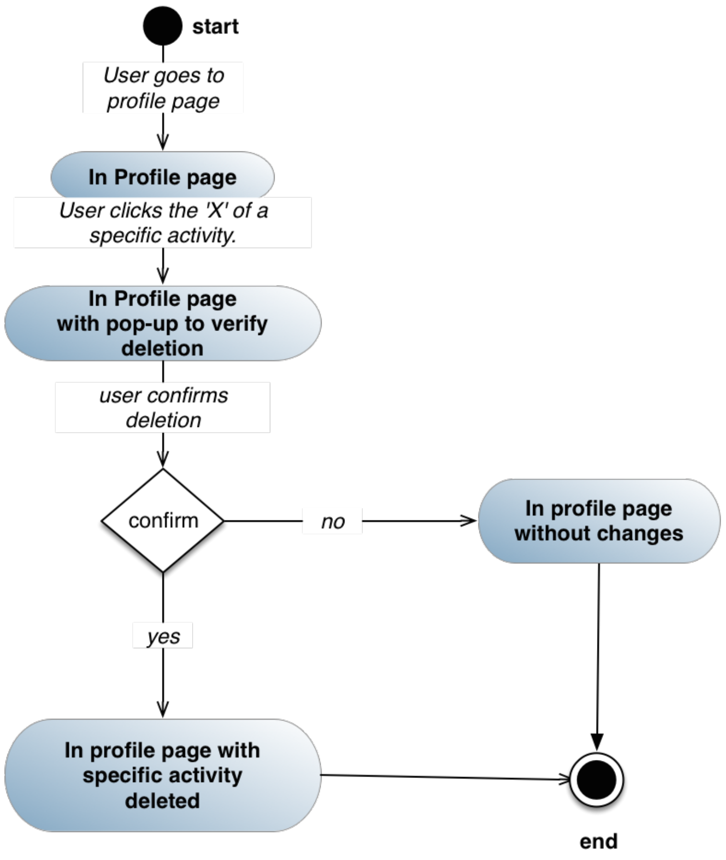
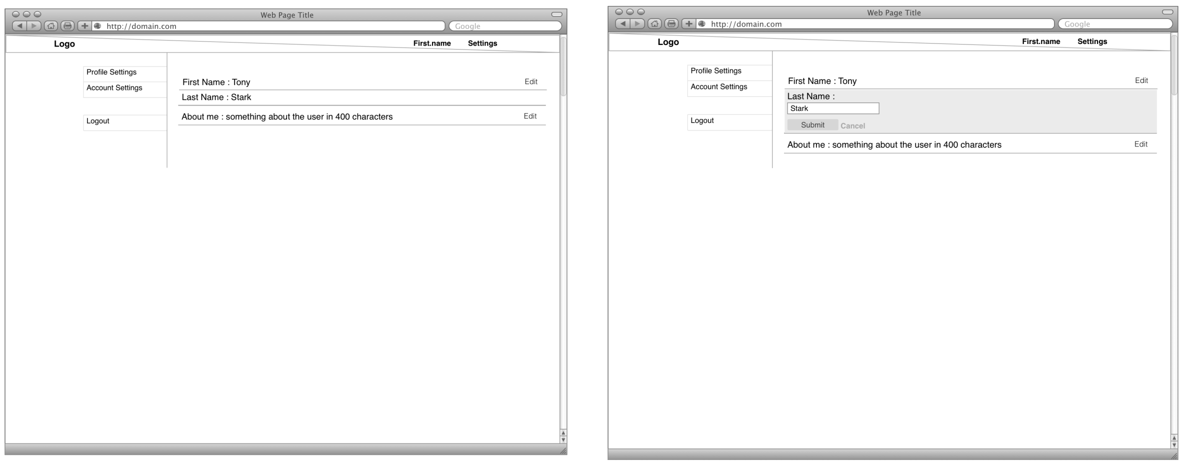

start
User goes to profile page
In Profile page
User clicks the 'X' of a specific activity.
In Profile page with pop-up to verify deletion
user confirms deletion
confirm
no
In profile page without changes
yes
In profile page with specific activity deleted
end

image 6.6: Delete Activity State Diagram



First Name : Tony
Last Name : Stark
About me : something about the user in 400 characters

First Name : Tony
Last Name :
Stark
About me : something about the user in 400 characters

image 6.7: Settings Page User Interface with and without editable field

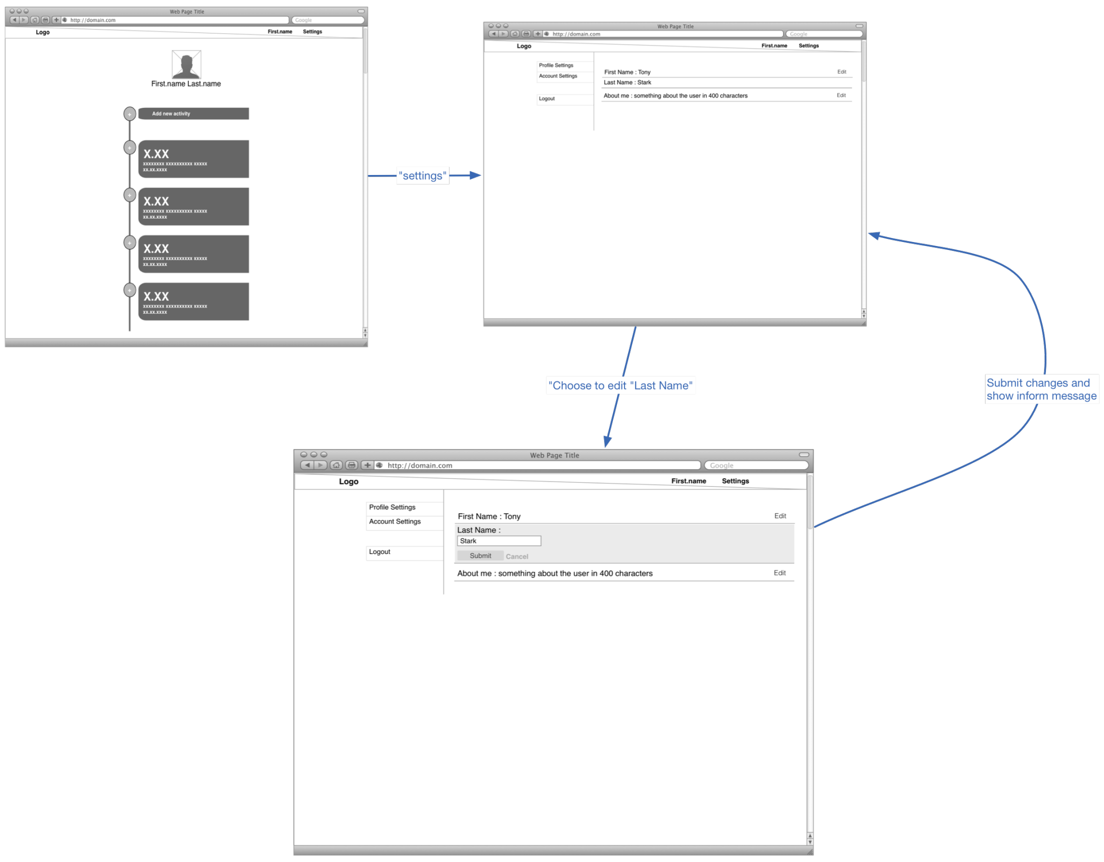


image 6.8: Edit Setting Wireframe

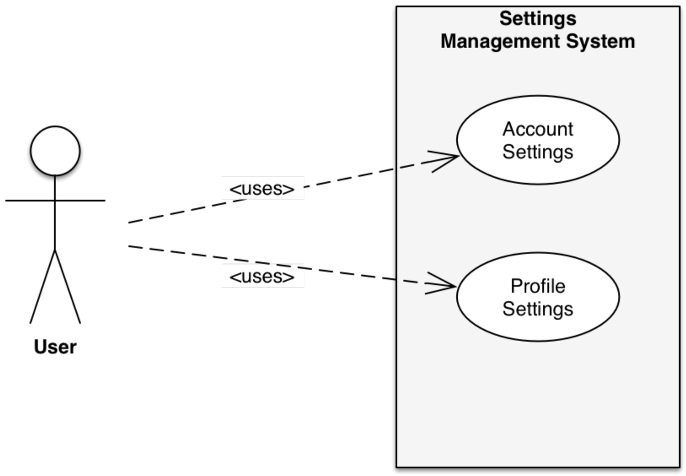


image 6.9: Edit Settings Use Case

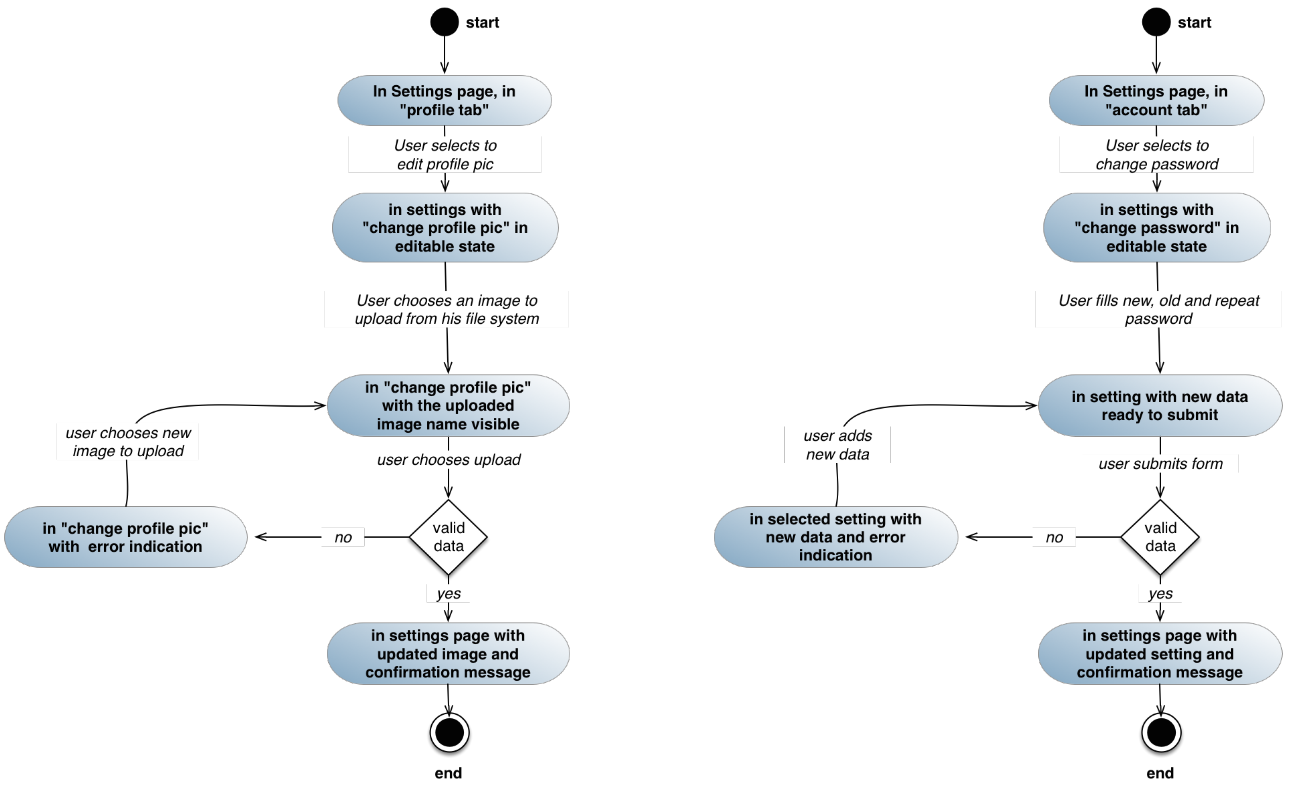

start
In Settings page, in "profile tab"
User selects to edit profile pic
in settings with "change profile pic" in editable state
User chooses an image to upload from his file system
in "change profile pic" with the uploaded image name visible
user chooses upload
valid data
no
in "change profile pic" with error indication
user chooses new image to upload
yes
in settings page with updated image and confirmation message
end
start
In Settings page, in "account tab"
User selects to change password
in settings with "change password" in editable state
User fills new, old and repeat password
in setting with new data ready to submit
user submits form
valid data
no
in selected setting with new data and error indication
user adds new data
yes
in settings page with updated setting and confirmation message
end

image 6.10: Edit Setting in Account and Profile tab State Diagram

6.6.4 Sprint review

This sprint was relatively easy to do, but there was a lot of content that should be written, so it was still time consuming. The only challenge was the file uploading process, for which, the fs library of node was needed. The page of the profile is almost complete. Now users can create and manage activities, so the application is functional and can be released to a close circle of users for testing and a first feedback. From this sprint and on, the application can be released in the end of each sprint, as it can help the users in some way, every time with more tools and functionality, which is the idea behind the agile methodology.

6.7 Sprint 7

6.7.1 Sprint planning

In this sprint we want to enhance the information that is stored for each activity. Till now activities only had the fields of discipline, performance and date. Now we are going to add also location, place, competition and notes. We will also need to update the functionality for edit activity in order to cover the new fields.

There will be need changes in profile page where the activities are shown, in add activity form and in edit activity form. New fields will be added and also all the validations that are required for them.

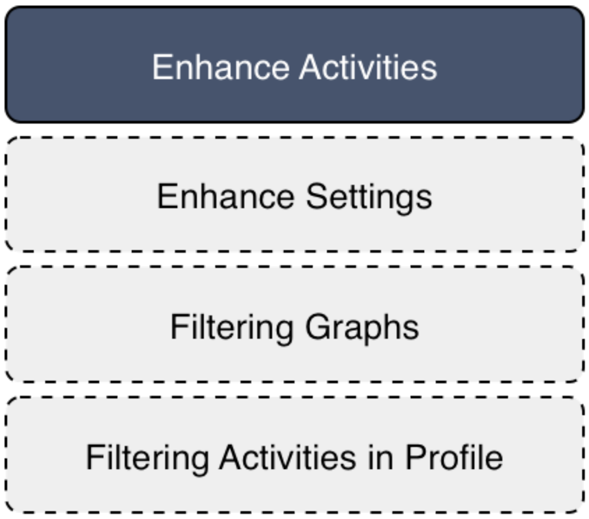


image 7.0: Backlog status for Planning of Sprint 7

6.7.2 Stand up

The input types we wanted to add didn't have any specialties. So we need to take some decisions about the length of input and the characters that are allowed on each of them. New fields will be added and also all the validations that are required for them. There will be a validation function for each one of them at javascript level, before the ajax request will be send for update or creation.

We had to create abstract validation functions in order to avoid writing multiple times the same code and increase the maintenance's difficulty. So we had to create validations in inputs for strings with limited set of characters, for numerical input with different and limited lengths and for dates.

6.7.3 Diagrams

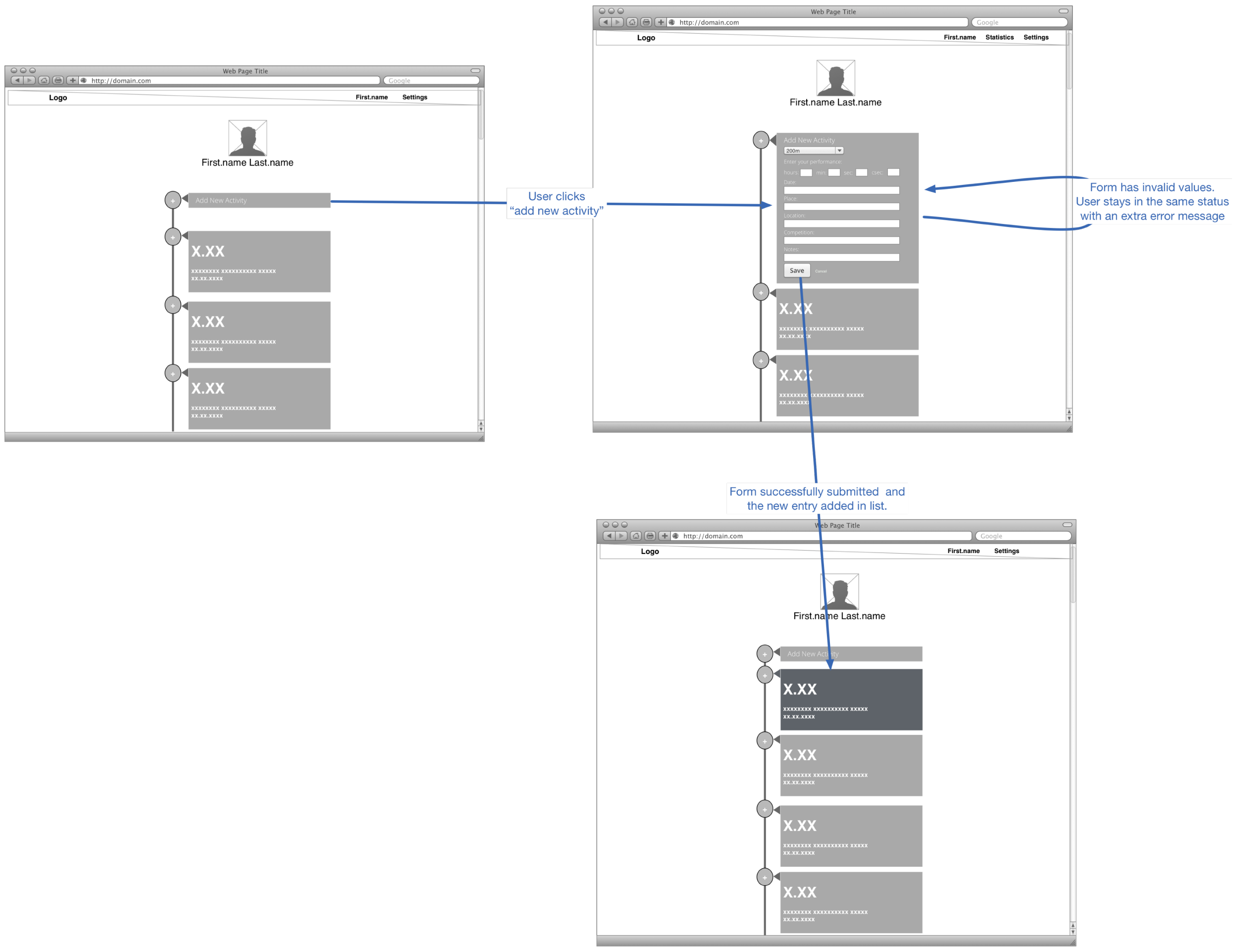


image 7.1: Add Activity Wireframe

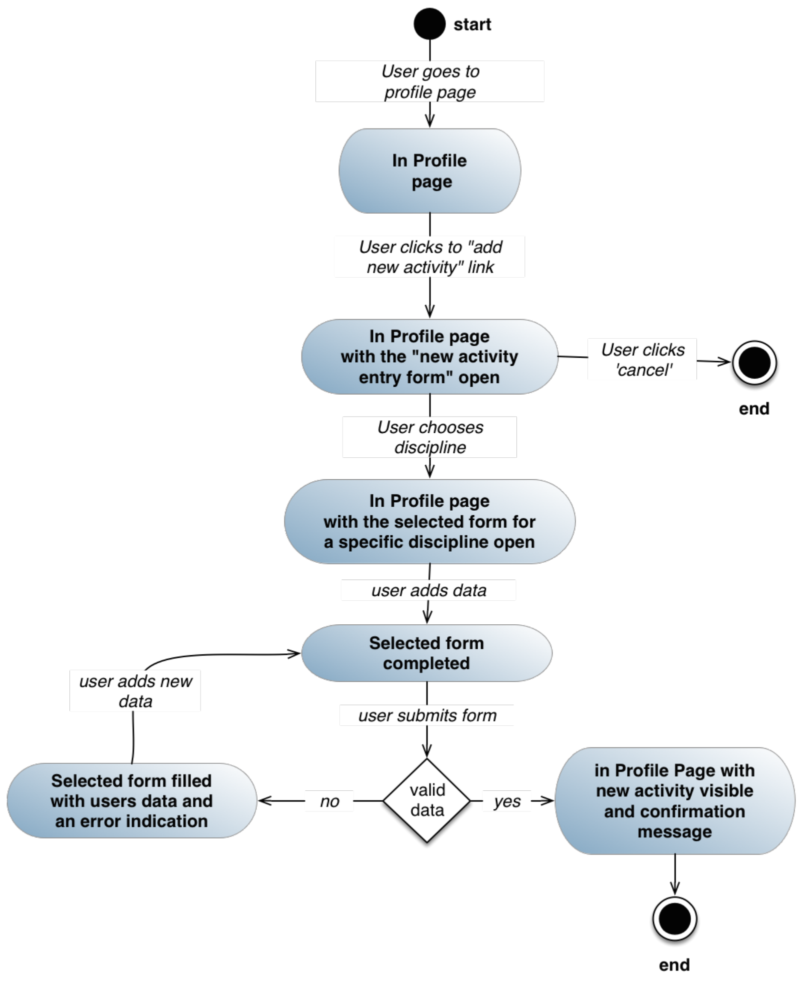

start
User goes to profile page
In Profile page
User clicks to "add new activity" link
In Profile page with the "new activity entry form" open
User clicks 'cancel'
end
User chooses discipline
In Profile page with the selected form for a specific discipline open
user adds data
Selected form completed
user adds new data
user submits form
Selected form filled with users data and an error indication
no
valid data
yes
in Profile Page with new activity visible and confirmation message
end

image 7.2: Add Activity State Diagram

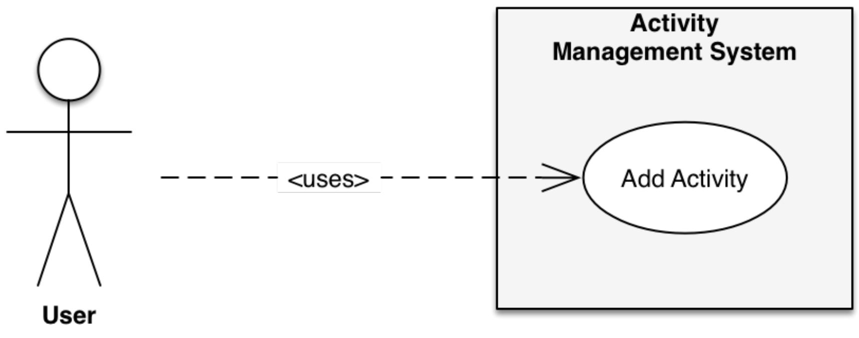


image 7.3: Add Activity Use Case

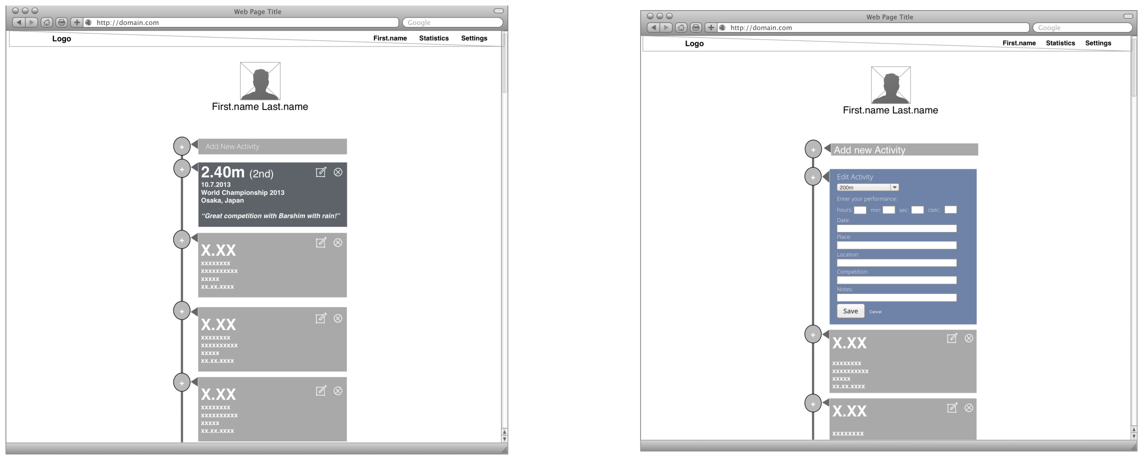


image 7.4: User Profile (with Edit and Delete options visible), and Edit Activity User Interface

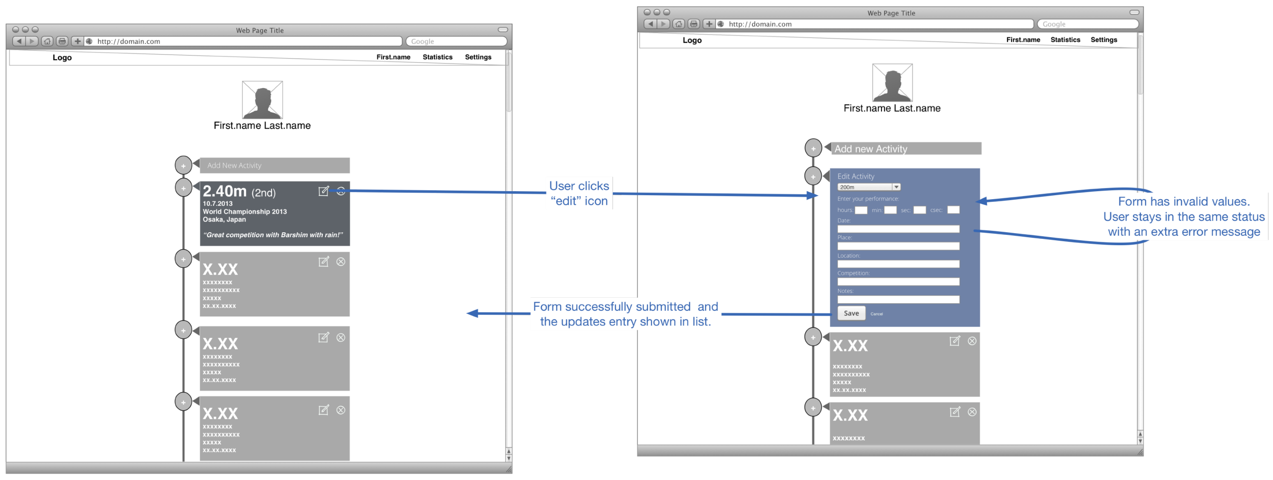


image 7.5: Edit Activity Wireframe

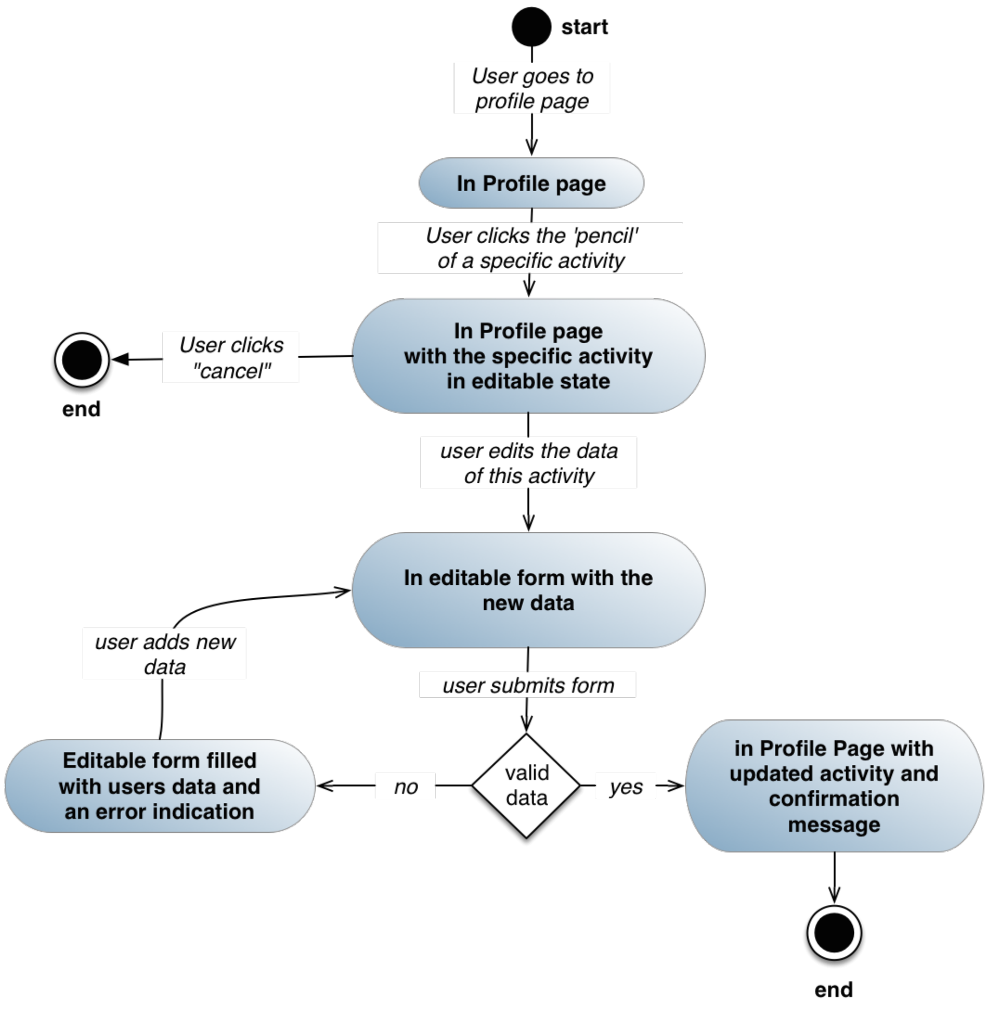

start
User goes to profile page
In Profile page
User clicks the 'pencil' of a specific activity
In Profile page with the specific activity in editable state
User clicks "cancel"
end
user edits the data of this activity
In editable form with the new data
user adds new data
user submits form
Editable form filled with users data and an error indication
no
valid data
yes
in Profile Page with updated activity and confirmation message
end

image 7.6: Edit Activity State Diagram

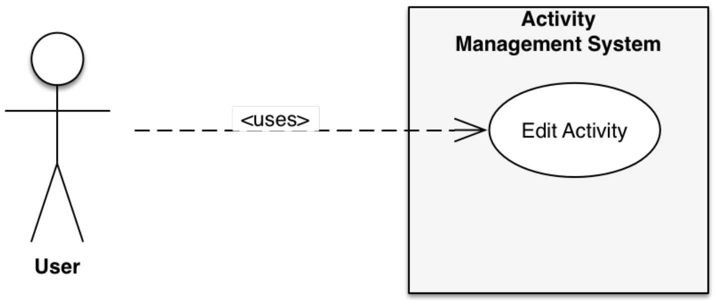


image 7.7: Edit Activity Use Case

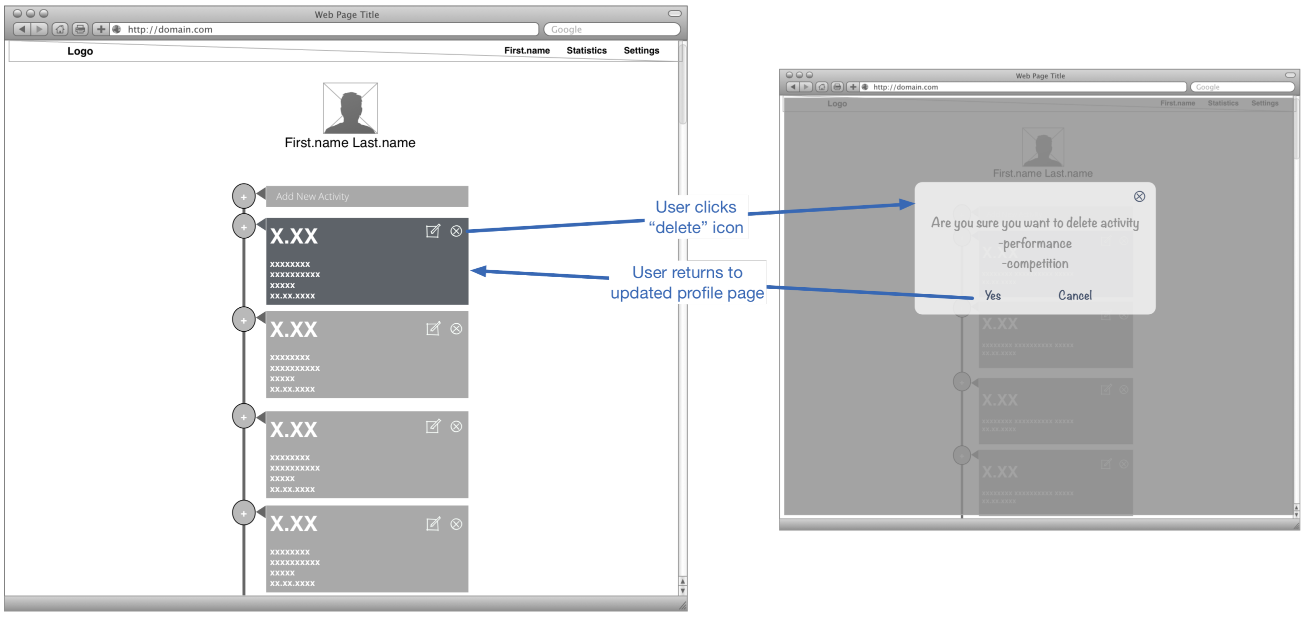


image 7.8: Delete Activity Wireframe

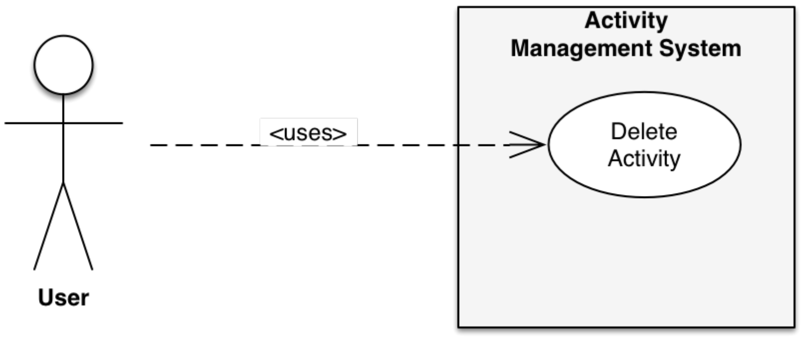


image 7.9: Delete Activity Use Case

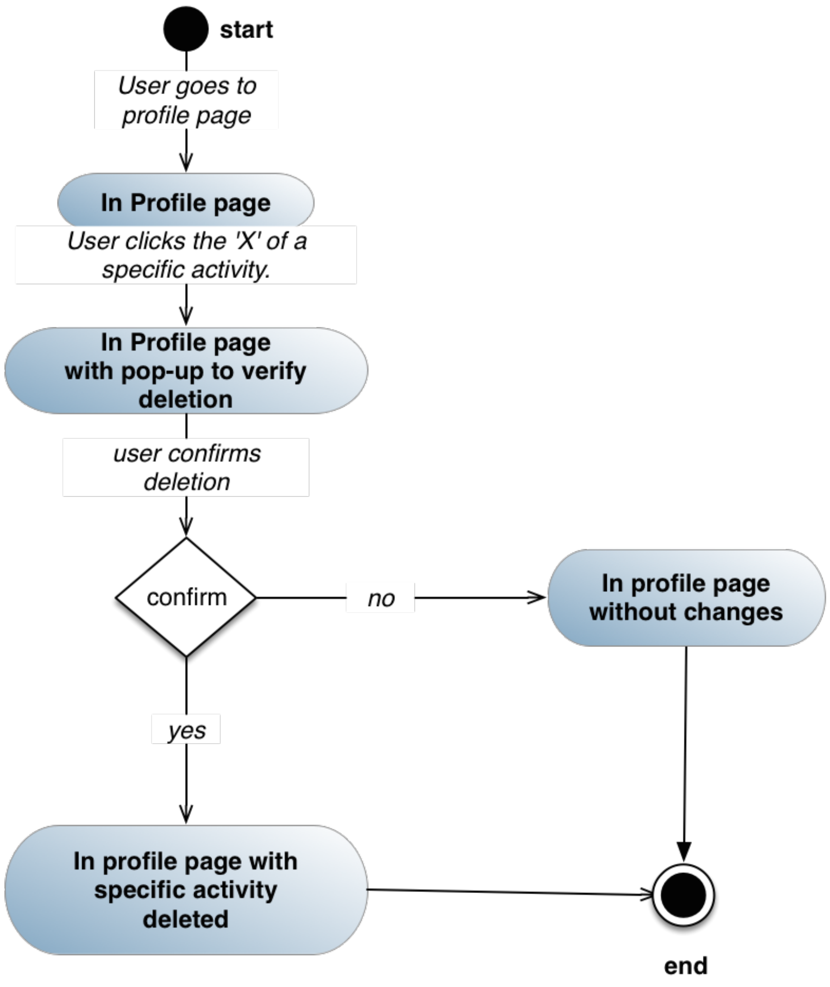

start
User goes to profile page
In Profile page
User clicks the 'X' of a specific activity.
In Profile page with pop-up to verify deletion
user confirms deletion
confirm
no
In profile page without changes
yes
In profile page with specific activity deleted
end

image 7.10: Delete Activity State Diagram

6.7.4 Sprint review

This sprint went as expected. We were on time, without any special difficulties. Working in parallel the client and the server allowed us to implement fast the functionality end-to-end. Also working both in the same programming language make all transactions easier, without heavy transformations and mapping between different formats. Having same validations in back-end and front-end saved us time and a lot of lines of code. But still the need for a more structured front-end seems very important.

6.8 Sprint 8

6.8.1 Sprint planning

In this sprint we want to enhance the information that is stored in settings page. Till now Profile Settings tab only had the fields of first and last name, and also the description. Now we are going to add also language, username, birthday, date format, gender. We will also need to update the functionality for edit these settings in order to cover the new fields. There will be need changes only in Profile Settings. New fields will be added and also all the validations that are required for them.

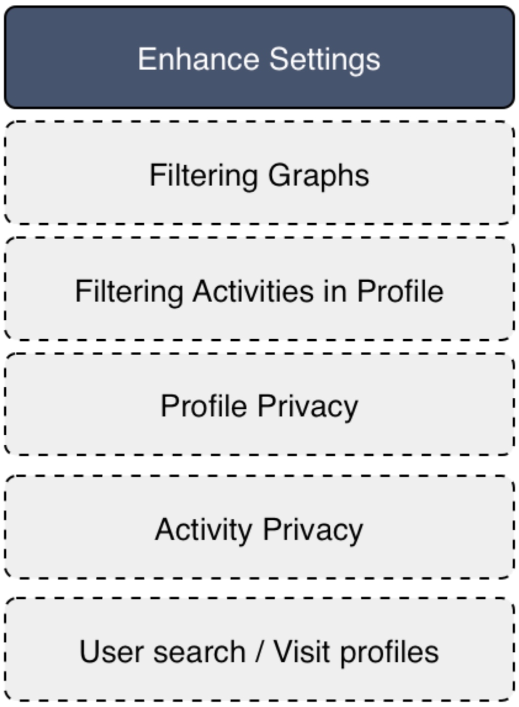


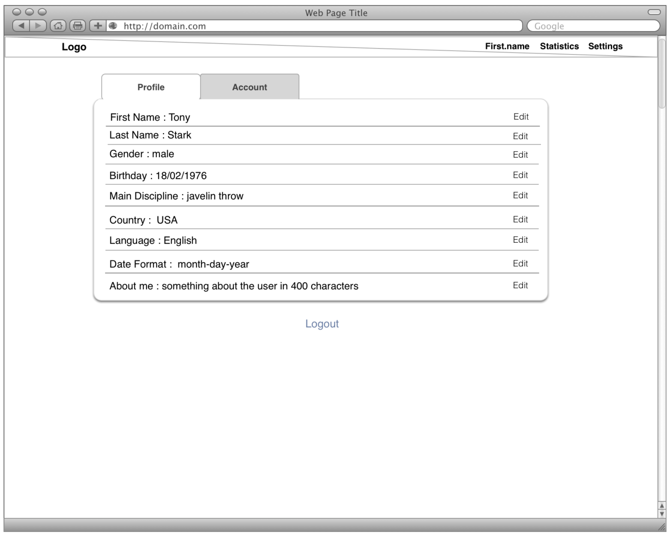
image 8.0: Backlog status for Planning of Sprint 8

6.8.2 Stand up

The input types we wanted to add had a lot of specialties. Language of the user should be selected from a list with existed languages. We also need to implement the logic for supporting multilingual User Interface and also the easy addition of new languages in the future. Thus we created some dictionaries. One for each language we want to support. This gives us the ability to easily duplicate and create new dictionaries for new languages. The username also had difficulties because we wanted to use it as an id in the url, for sharing purposes in the future. For example visit the profile with username ‘mathiou’ with the url : www.domain.com/mathiou. The birthday field was one of date type, so we reuse the date input we use in activity form. Here there was some different checks, because we were needed to add dates in different range. We allow users that were born after 1900 and before 2010. Gender was easy to implemented because it were needed just a boolean variable to store the two different values. We set male as true for this.

In this Sprint we decided to change our tabs from vertical to horizontal in order to have a better screen management, and also keep the UI consistent in the future between different devices. Having a vertical menu only in this page would create confusion, in some cases, to our users because there is no other vertical navigation menu or element in our app. Thus we change the design, to keep consistency in our designs.

6.8.3 Diagrams



Web Page Title

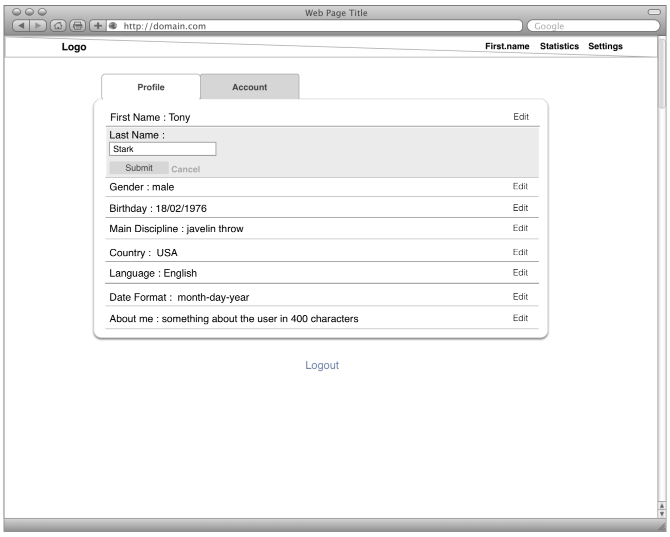
http://domain.com

Logo First.name Statistics Settings

Profile Account

First Name : Tony Edit
Last Name : Stark Edit
Gender : male Edit
Birthday : 18/02/1976 Edit
Main Discipline : javelin throw Edit
Country : USA Edit
Language : English Edit
Date Format : month-day-year Edit
About me : something about the user in 400 characters Edit

Logout



Web Page Title

http://domain.com

Logo First.name Statistics Settings

Profile Account

First Name : Tony Edit
Last Name :
Stark
Submit Cancel
Gender : male Edit
Birthday : 18/02/1976 Edit
Main Discipline : javelin throw Edit
Country : USA Edit
Language : English Edit
Date Format : month-day-year Edit
About me : something about the user in 400 characters Edit

Logout

image 8.1: Settings Page with and without editable field

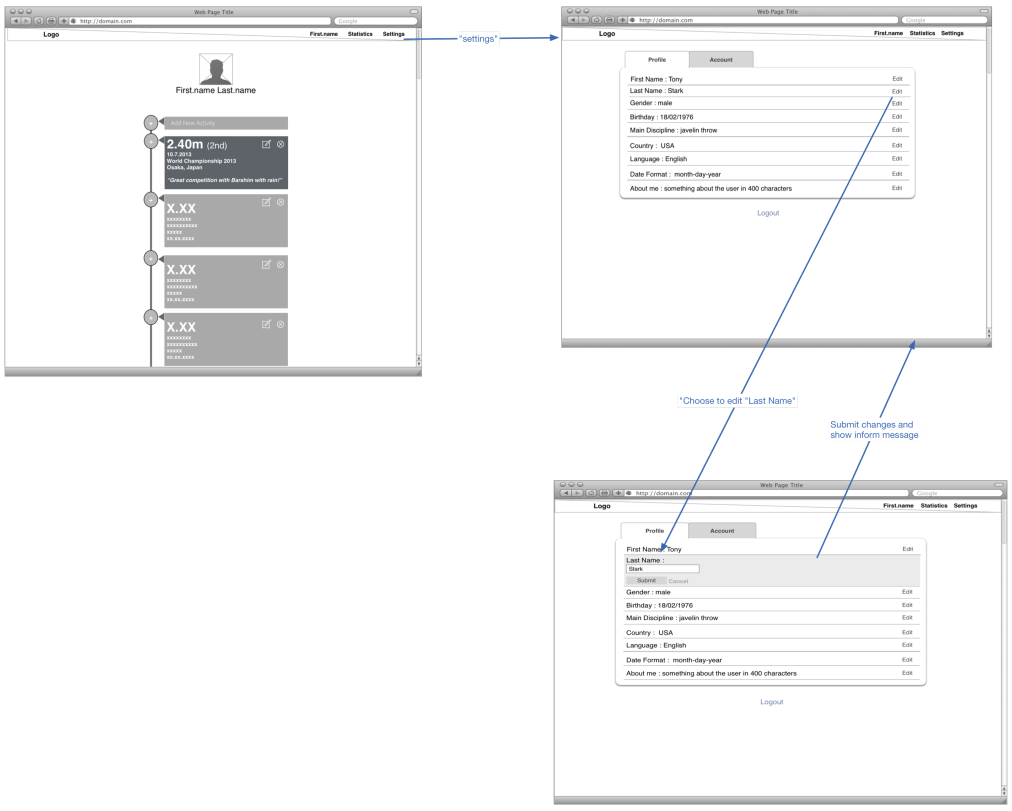


image 8.2: Edit Settings Wireframe

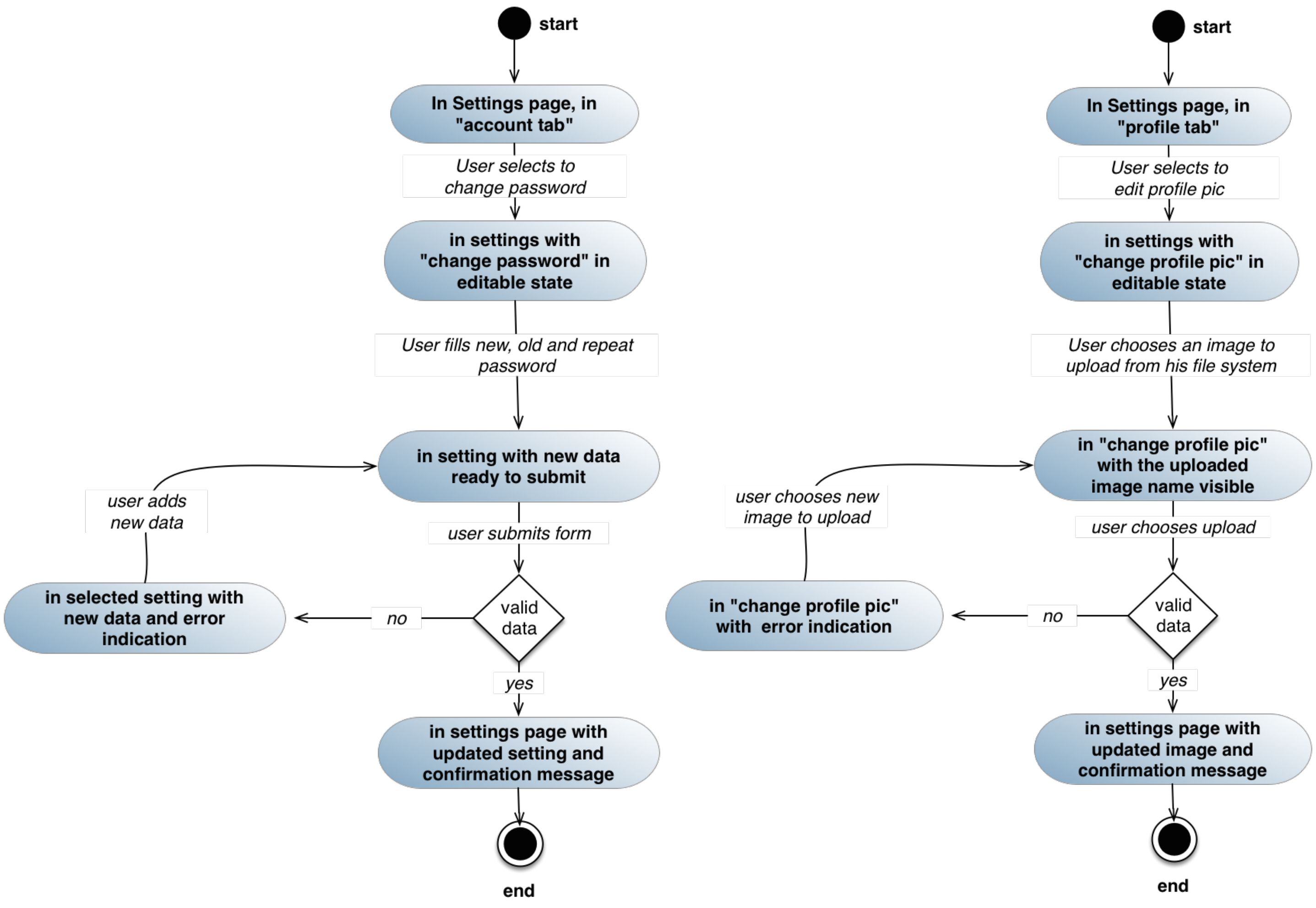

start
In Settings page, in "account tab"
User selects to change password
in settings with "change password" in editable state
User fills new, old and repeat password
in setting with new data ready to submit
user adds new data
user submits form
in selected setting with new data and error indication
no
valid data
yes
in settings page with updated setting and confirmation message
end
start
In Settings page, in "profile tab"
User selects to edit profile pic
in settings with "change profile pic" in editable state
User chooses an image to upload from his file system
in "change profile pic" with the uploaded image name visible
user chooses new image to upload
user chooses upload
in "change profile pic" with error indication
no
valid data
yes
in settings page with updated image and confirmation message
end

image 8.3: Change Password and Profile Picture State Diagram

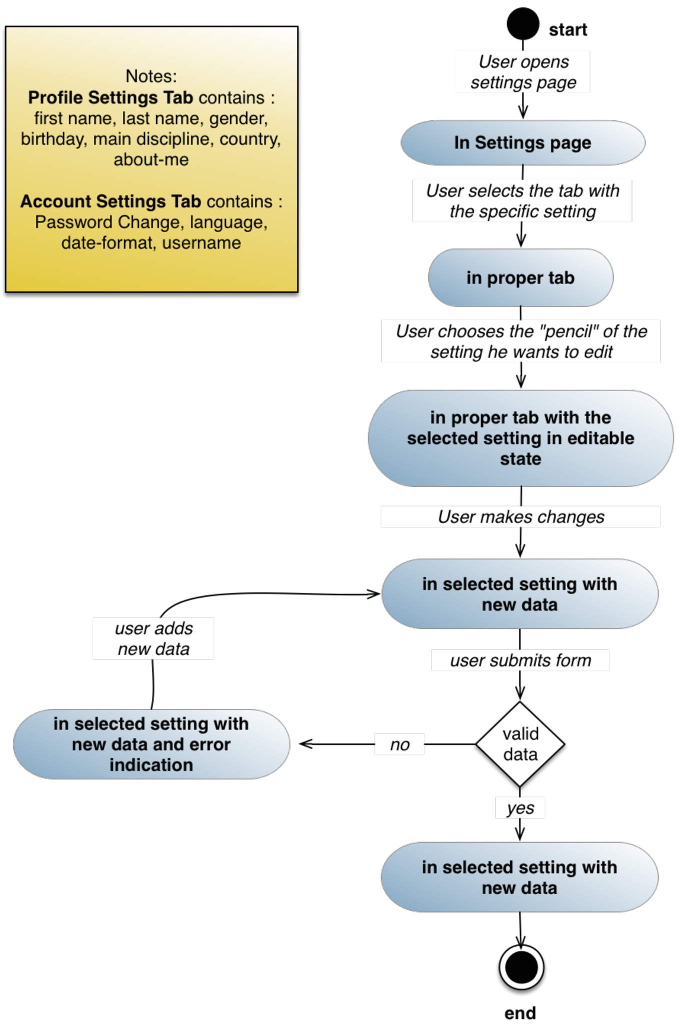

Notes:
Profile Settings Tab contains : first name, last name, gender, birthday, main discipline, country, about-me
Account Settings Tab contains : Password Change, language, date-format, username
start
User opens settings page
In Settings page
User selects the tab with the specific setting
in proper tab
User chooses the "pencil" of the setting he wants to edit
in proper tab with the selected setting in editable state
User makes changes
in selected setting with new data
user submits form
valid data
no
in selected setting with new data and error indication
user adds new data
yes
in selected setting with new data
end

image 8.4: Edit Setting in Account or Profile tab State Diagram

6.8.4 Sprint review

This Sprint had no special difficulties, but again the logic in the client side in order to make changes in UI without overloading the server with unnecessary work makes the code complex and hard to maintain. So we decided from the next Sprint to move the front-end implementation to AngularJS MVC framework. This will allow us to use two-way data binding between our User Interface and Business logic in client side.

6.9 Sprint 9

6.9.1 Sprint planning

In this Sprint we will make two great changes in web-client side of our app. The first one is that we will introduce AngularJS, an MVC framework that supports two-way data-binding, and also BootstrapJS, a javascript-css framework that will help us to design and implement, in an easier way, a responsive UI for different screen-size devices. Because these changes requires massive changes in client side we will spent all this Sprint in designing the MVC logic for AngularJS and prototyping our application in three different device categories. The categories we support are desktops, tablets and mobiles.

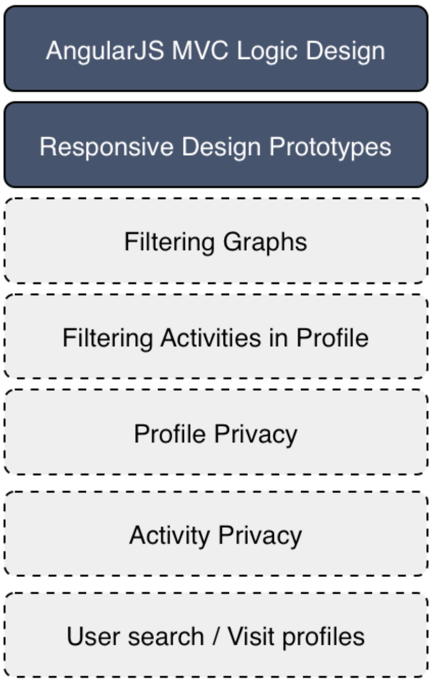

AngularJS MVC Logic Design
Responsive Design Prototypes
Filtering Graphs
Filtering Activities in Profile
Profile Privacy
Activity Privacy
User search / Visit profiles

image 9.0: Backlog status for Planning of Sprint 9

6.9.2 Stand up

We start form UI design because it will define, more or less, the whole information architecture of our application. Not all types of devices will support the same functionality, because there are restrictions in usage. We can separate our application in 4 main areas.

1. Login - Register
2. User Profile - Activities
3. Statistics
4. Settings

We decided to support the whole functionality for desktops and tablets, and only the Login-Register and User Profile - Activities in mobiles. There are two main reasons for this. The first one is the nature of mobiles. Mobiles are used mainly on the way and in short time blocks. So we need to support only the basic functionality which is the add and view activity. Obviously the login and register functionality are necessary. The second reason is that it is hard for our users to navigate and manipulate the data graphs, which are a different way to view your activities, in a small touch screen. Also Settings have information that is rarely change. In addition we wanted to avoid all the errors in the settings form, that are easier to happen in a mobile.

For supporting the same information architecture on business logic we need to create 4 different Controllers. The mainController which will handle all the logic that is shared along the whole application, and profileController, settingsController and statisticsController for handling all the logic that is required in each one page. The business logic for login and registration will not be implemented in AngularJS, since it is required before our user enter our application. This part of business logic will stay in server side and will be handled by nodeJS and vanilla Javascript on web client side.

6.9.3 Diagrams

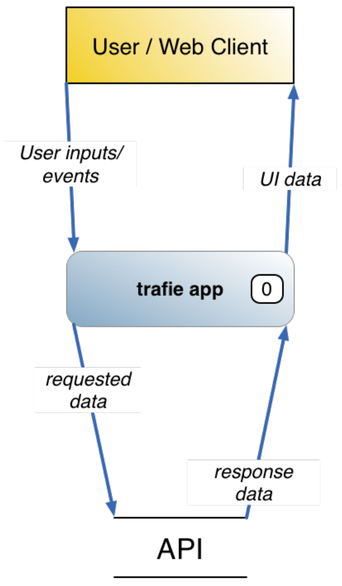


image 9.1: Application Data Flow Level 0

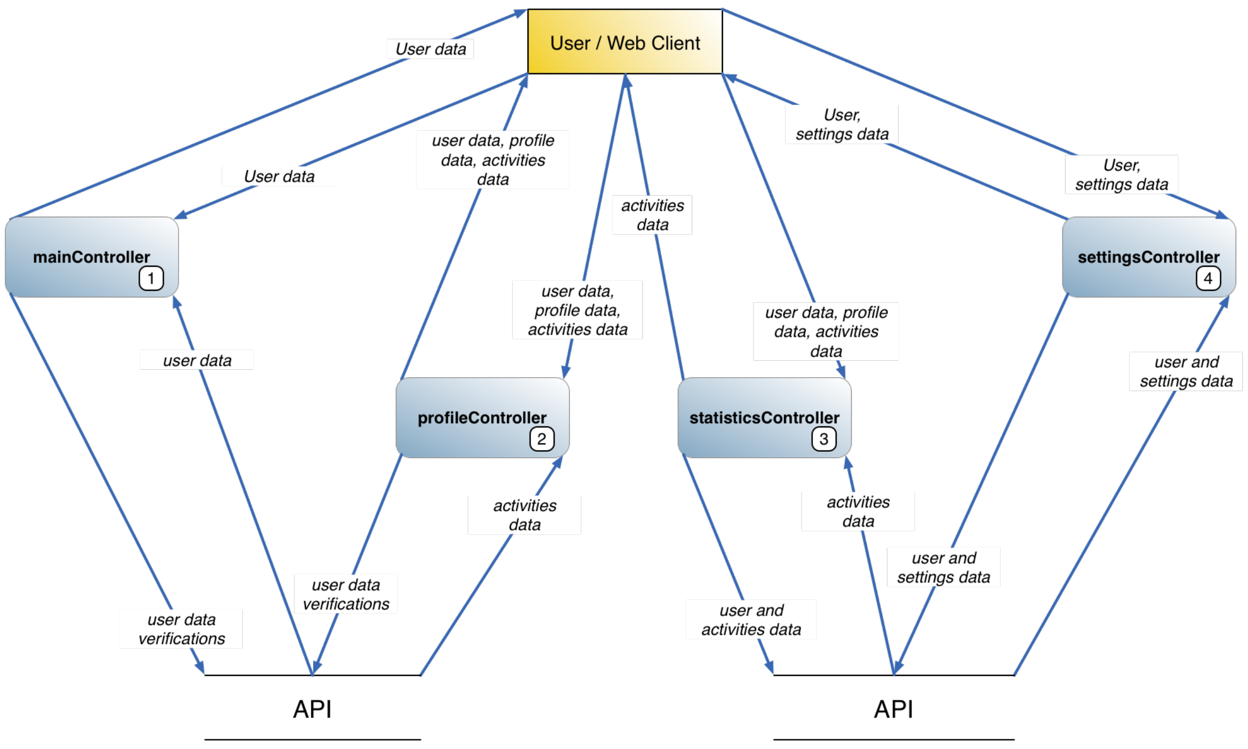
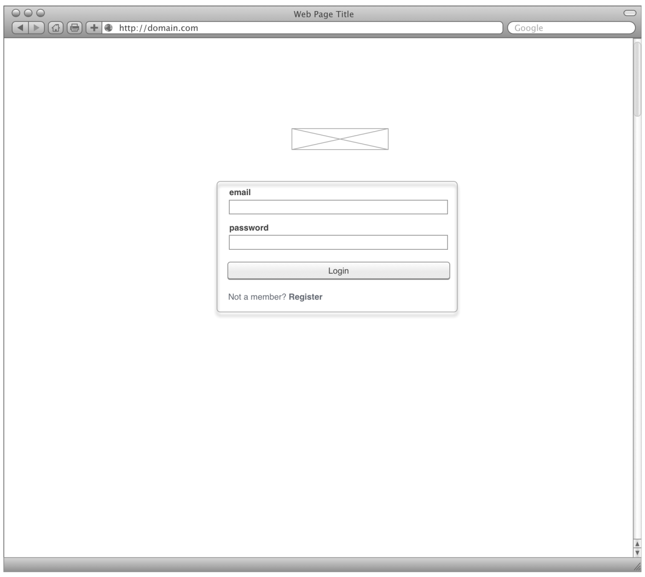
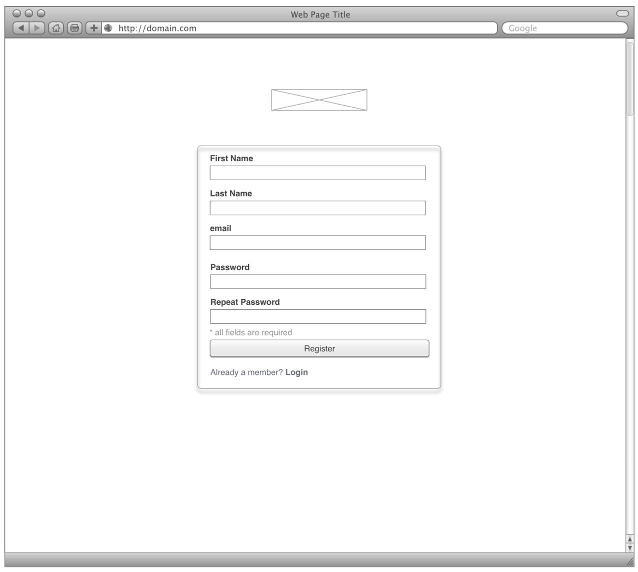
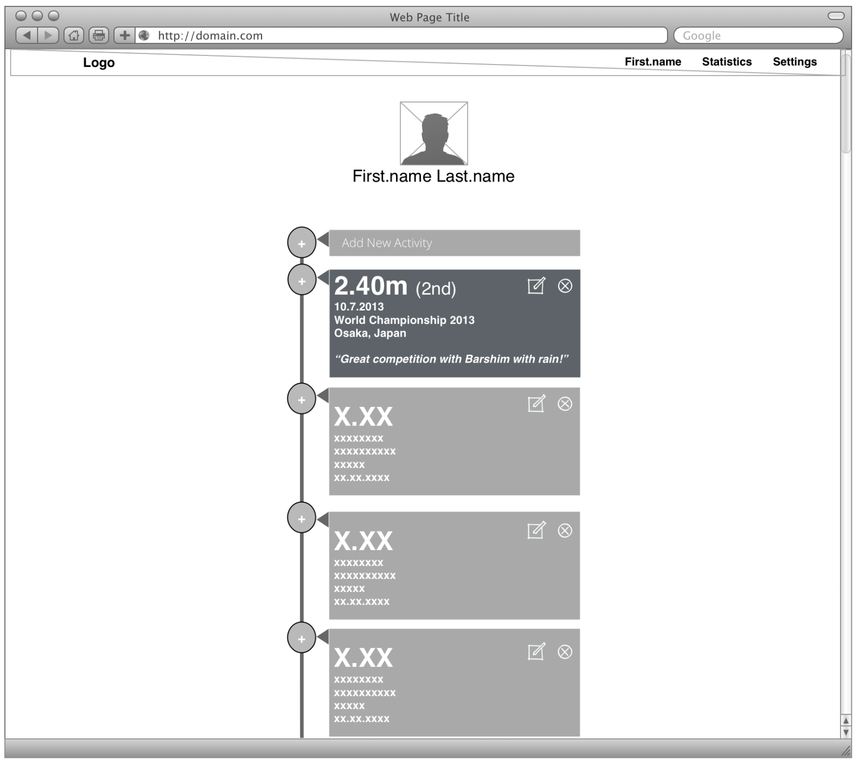


image 9.2: Application Data Flow Level 1



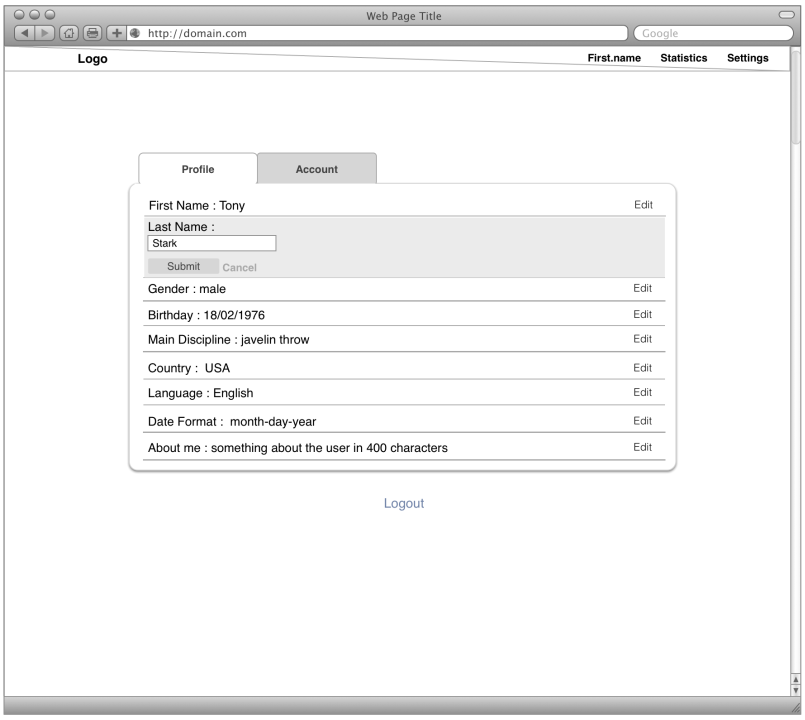




2.40m (2nd)
10.7.2013
World Championship 2013
Osaka, Japan

"Great competition with Barshim with rain!"

image 9.3: Desktop version of Login, Register and Profile page



First Name : Tony Edit

Last Name :

Stark

Gender : male Edit

Birthday : 18/02/1976 Edit

Main Discipline : javelin throw Edit

Country : USA Edit

Language : English Edit

Date Format : month-day-year Edit

About me : something about the user in 400 characters Edit

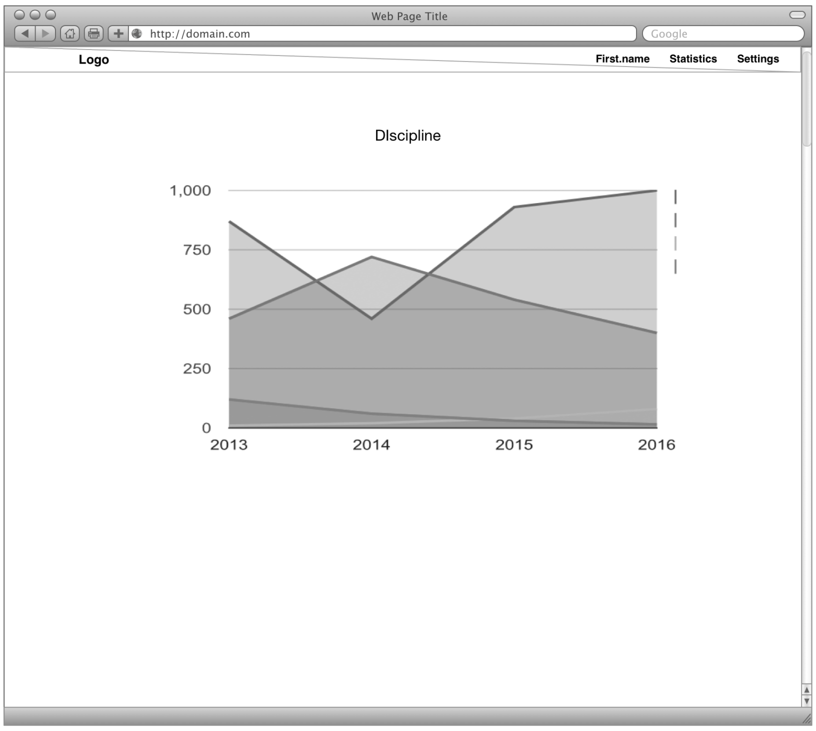


image 9.4: Desktop version of Settings and Statistics page

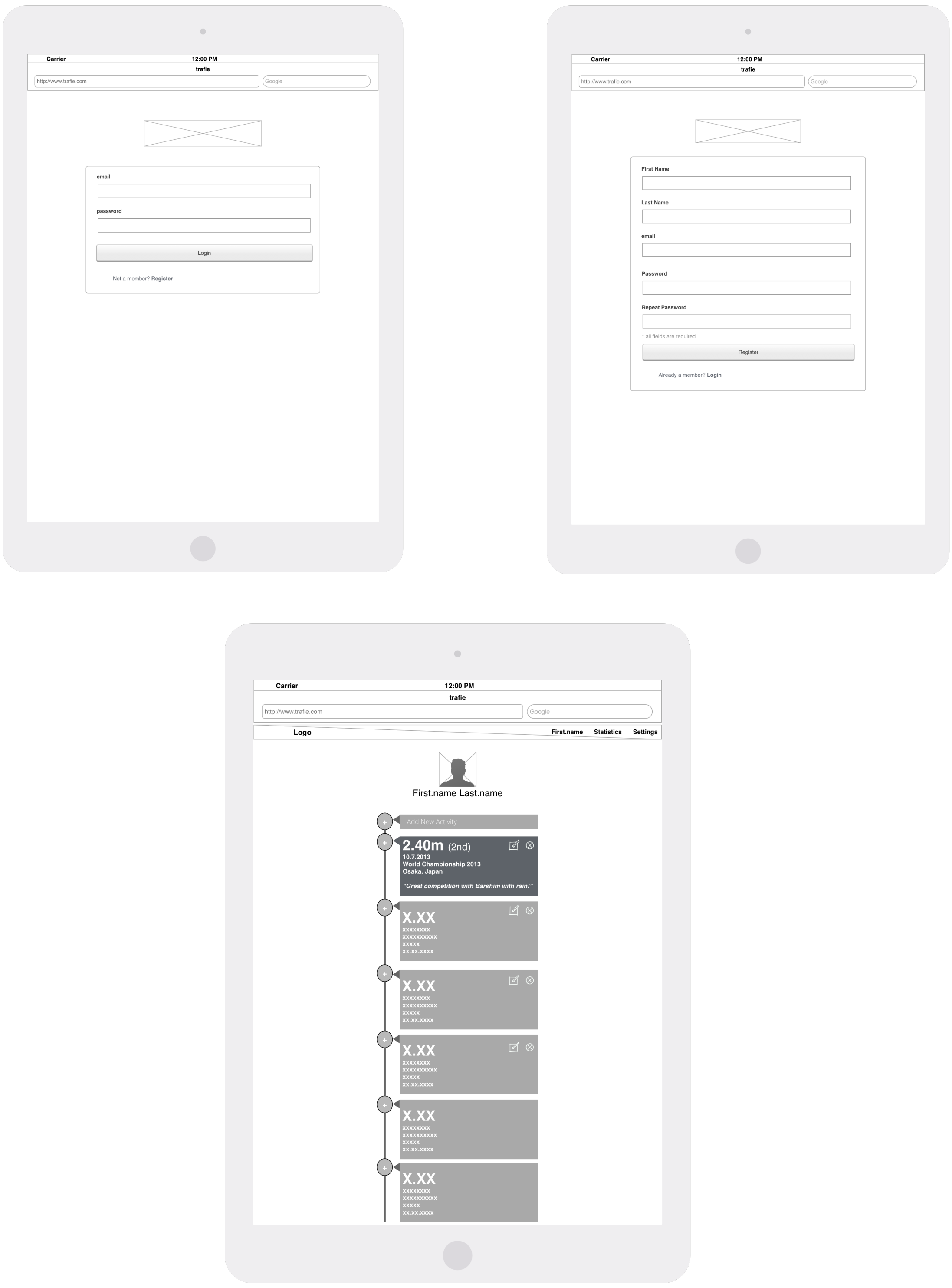
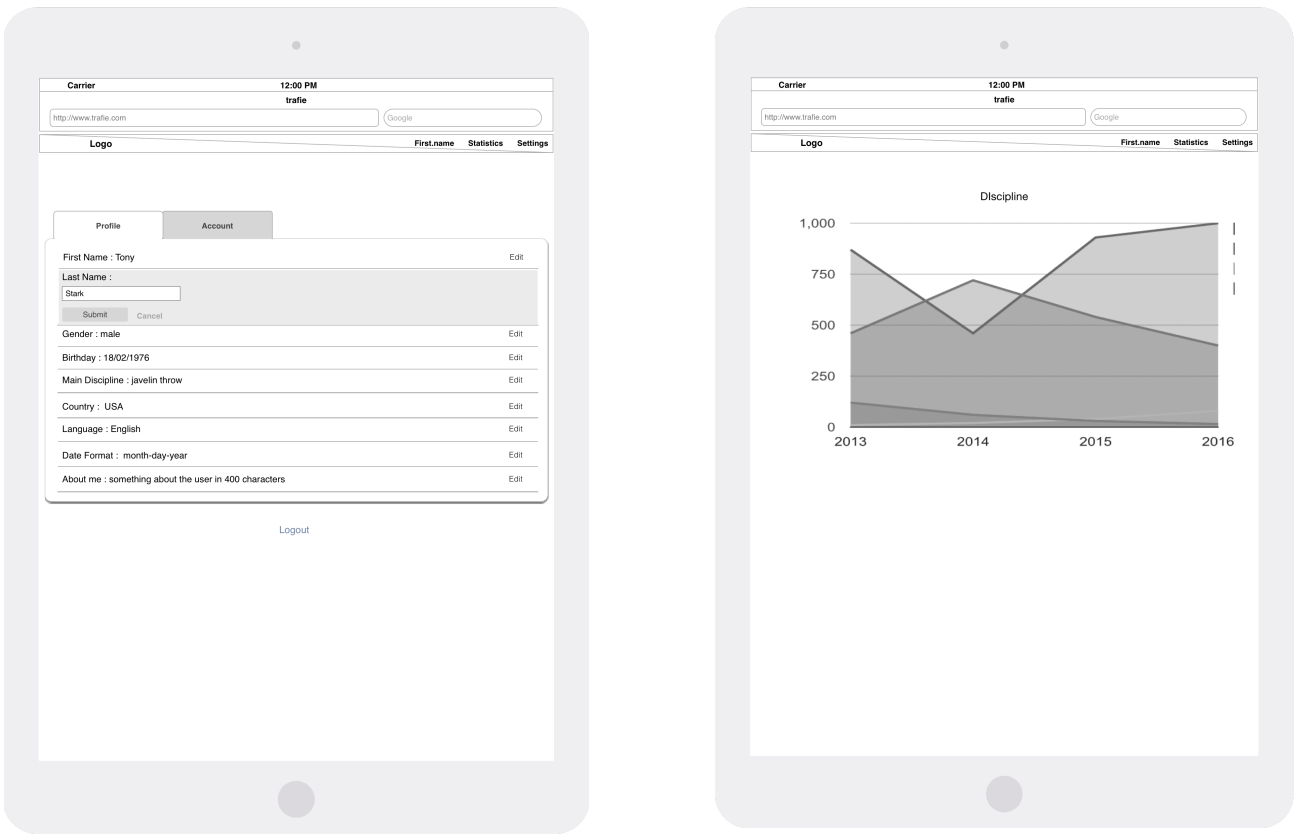


image 9.5: Tablet version of Login, Register and Profile page


Carrier 12:00 PM
trafie
http://www.trafie.com
Google
Logo First.name Statistics Settings

Profile | Account

First Name : Tony Edit
Last Name :
Stark
Submit Cancel
Gender : male Edit
Birthday : 18/02/1976 Edit
Main Discipline : javelin throw Edit
Country : USA Edit
Language : English Edit
Date Format : month-day-year Edit
About me : something about the user in 400 characters Edit

Logout

image 9.6: Tablet version of Settings and Statistics page

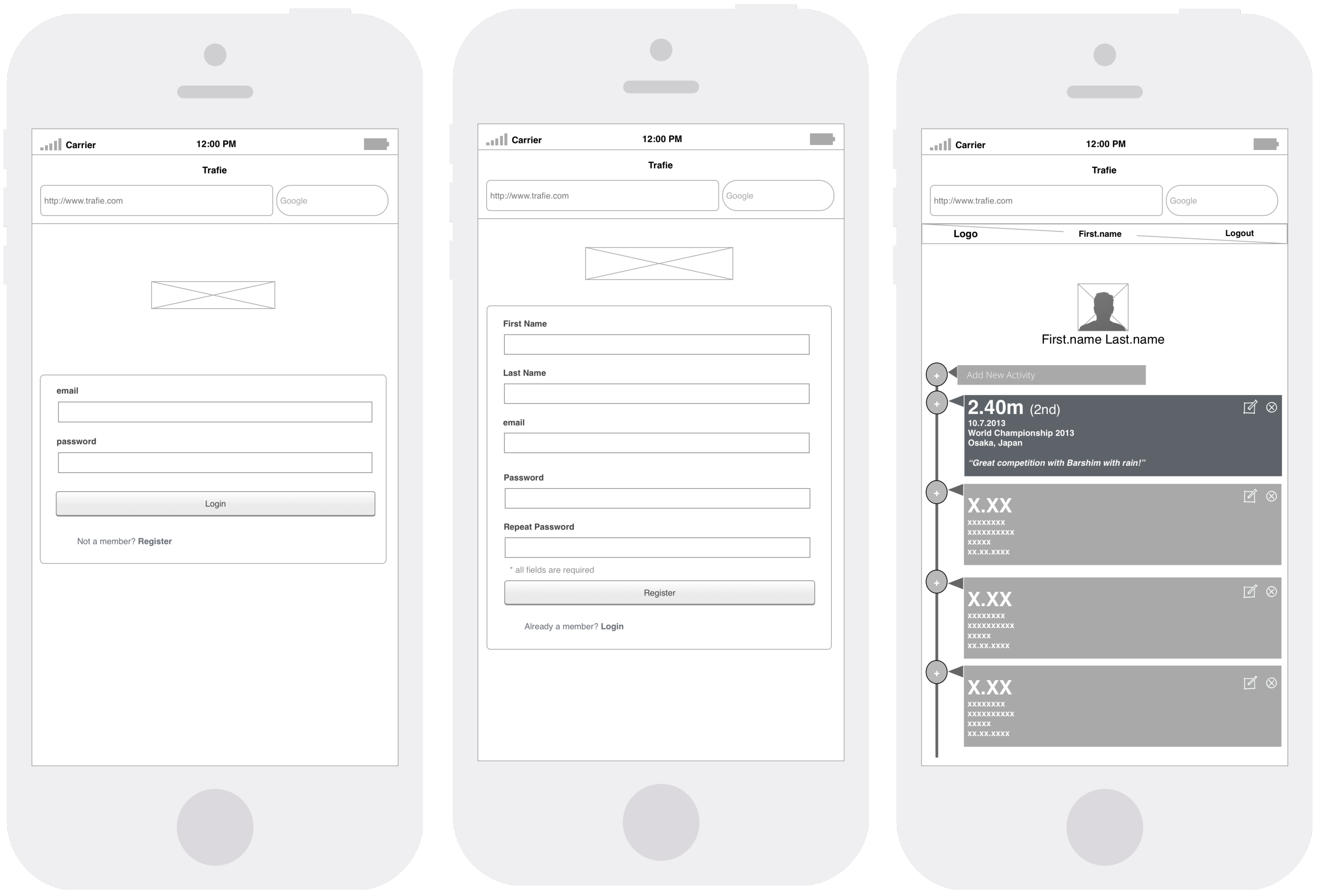


image 9.7: Mobile version of Login, Register and Profile page

6.9.4 Sprint review

The main difficulties on this Sprint was that we were needed to take some decisions without having exact knowledge of how the application will be in it's final version. Basically in this Sprint, we set an initial point and we will adapt all necessary changes on an agile way, as the implementation moves on.

6.10 Sprint 10

6.10.1 Sprint planning

In this Sprint we will transform the client-side implementation for create, edit and delete activities, replacing all the javascript logic and UI manipulation with AngularJS. This should not affect the User Interface and all these changes should be seamless to the end user.

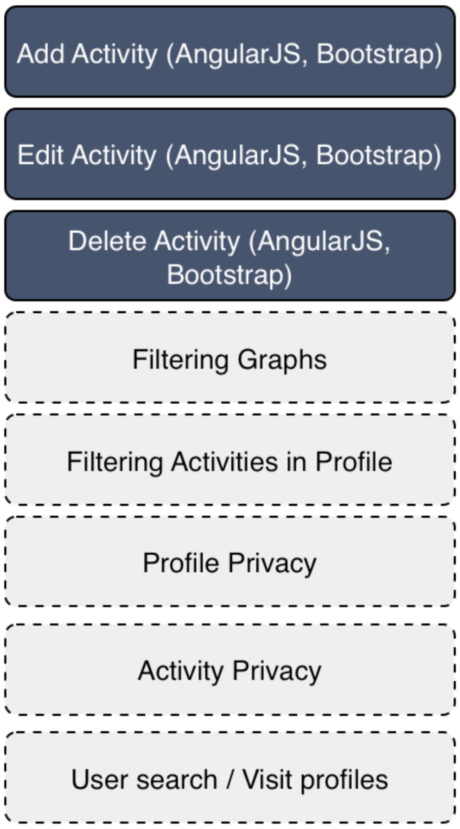

Add Activity (AngularJS, Bootstrap)
Edit Activity (AngularJS, Bootstrap)
Delete Activity (AngularJS, Bootstrap)
Filtering Graphs
Filtering Activities in Profile
Profile Privacy
Activity Privacy
User search / Visit profiles

image 10.0: Backlog status for Planning of Sprint 10

6.10.2 Stand up

The controllers we need to implement are mainController and profileController. Given that, this is the first time we create controllers for our app, we need to initialise the app and support some more generic actions.

In mainController we inject all the required services that will be available across the whole application, and also we will initialise and instantiate some variables and functions. We need a variable that will store the information about the type of our device, touch or not, in order to know how to interact with our user. For example in touch devices we should not have hover effects in our elements. In addition we need to store here all the validation patterns, which also will be used across our app. MainController initialise the controllers, thus must be the first controller that is loaded.

In profileController we initialise our user. On initProfile function we request from the server the user's profile data calling the '/users/userID' route, in getProfile function, with the specific userID. Using the same userID we request from the server all activities of the user with a GET request through '/users/userID/activities', in getActivities function. For all returned data, we have create bindings between controller and HTML page. So when all these data will be returned, they will automatically appeared in our page.

SubmitNewActivity gets the data inserted by user, make some transformations (i.e date) to specific formats and makes a POST request to route ''/users/userID/activities''on the server, in order to create a new activity. After a successful response the new activity will be created in UI and will be appeared in activities list without affecting the rest User Interface.

DeleteActivity will first call the ModalSvc in order to create a modal that will ask user to confirm the deletion of this specific activity. If user confirm then a DELETE request will be done to server through route ‘/users/userID/activities/activityID’, where userID is the ID of the specific user and activityID is the id of the specific activity. A confirmation message will be appeared after the deletion.

6.10.3 Diagrams

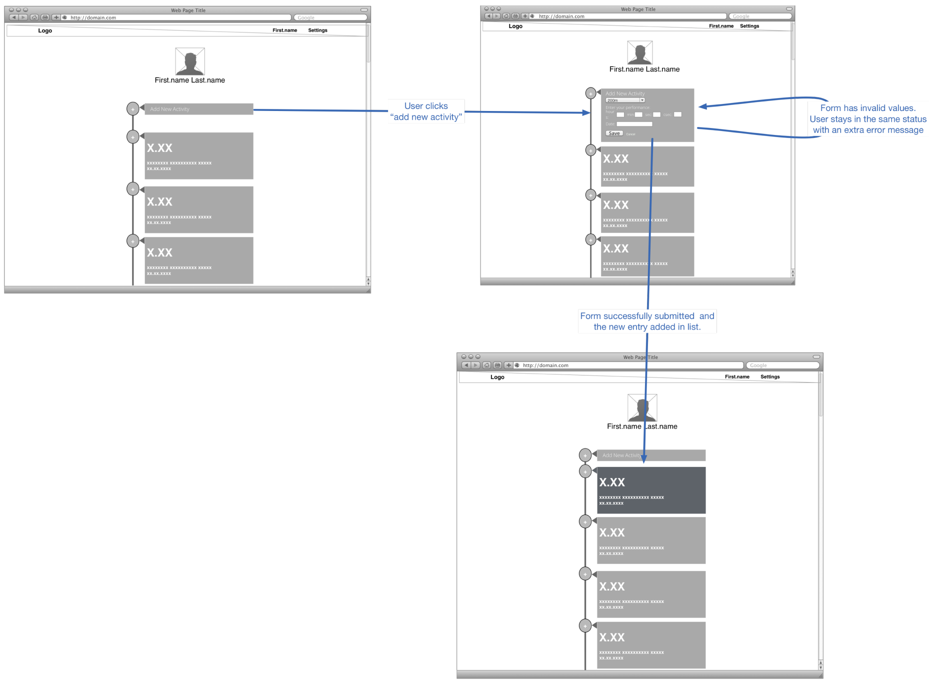

User clicks "add new activity"
Form has invalid values. User stays in the same status with an extra error message
Form successfully submitted and the new entry added in list.
Web Page Title
Logo
First.name Settings
First.name Last.name
Add New Activity
X.XX
Save

image 10.1: Add activity Wireframe

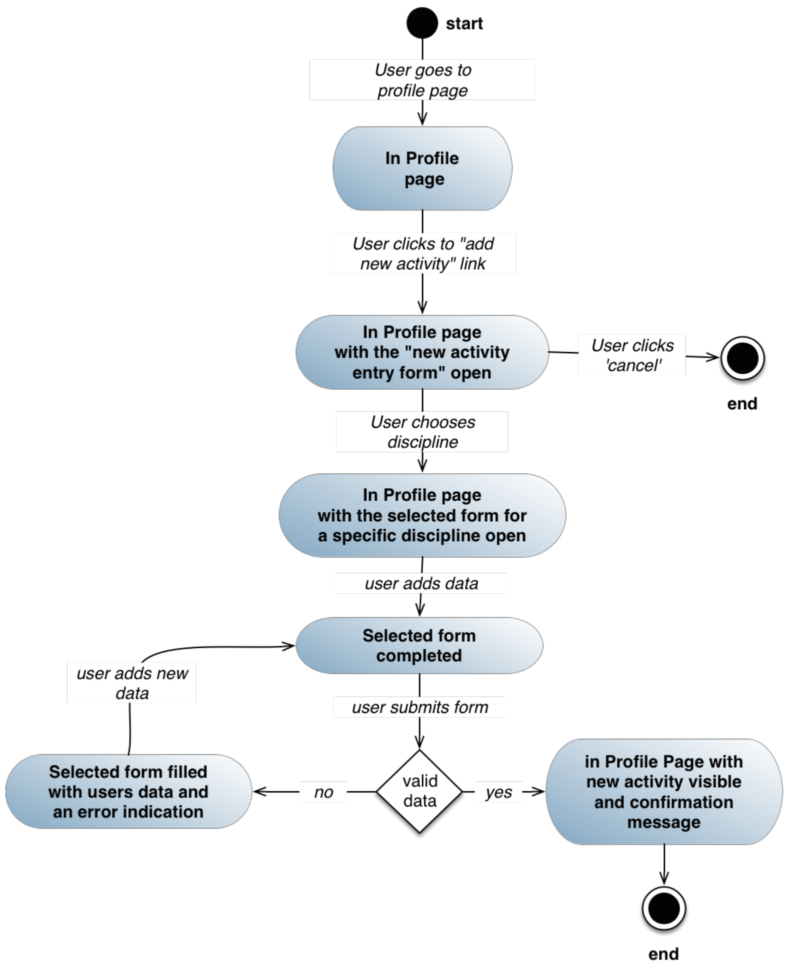

start
User goes to profile page
In Profile page
User clicks to "add new activity" link
In Profile page with the "new activity entry form" open
User clicks 'cancel'
end
User chooses discipline
In Profile page with the selected form for a specific discipline open
user adds data
Selected form completed
user adds new data
user submits form
Selected form filled with users data and an error indication
no
valid data
yes
in Profile Page with new activity visible and confirmation message
end

image 10.2: Add Activity State Diagram

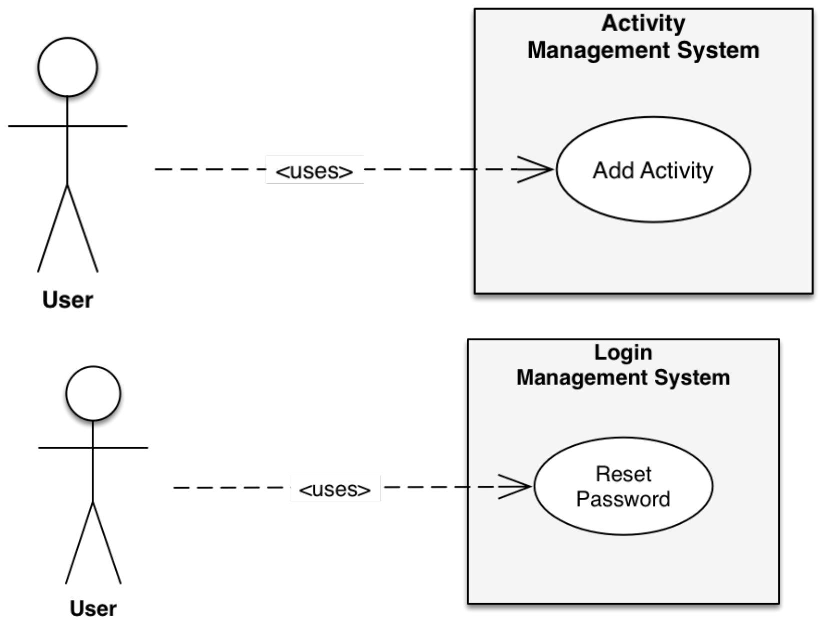


image 10.3: Add Activity Use Case

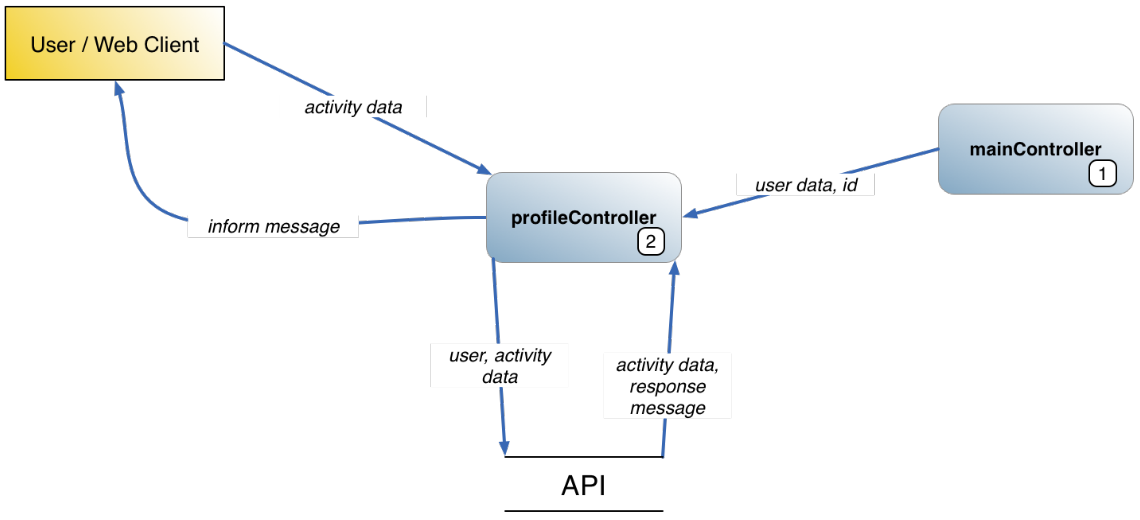


image 10.4: Add Activity Data Flow Diagram

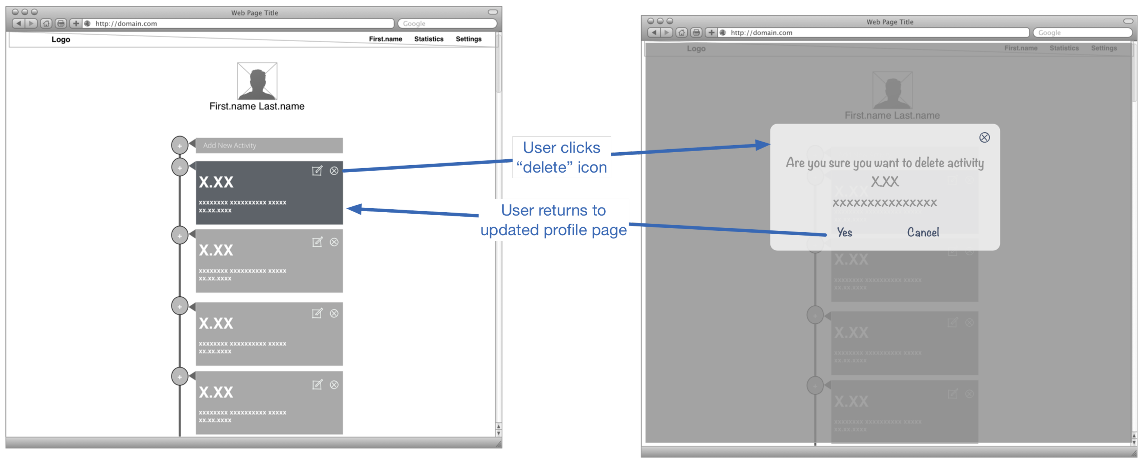


image 10.5: Delete Activity Wireframe

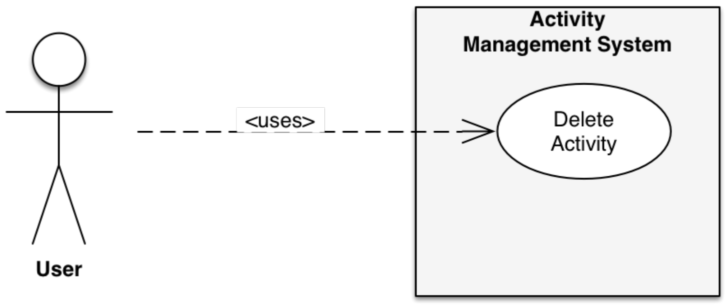


image 10.6: Delete Activity Use Case

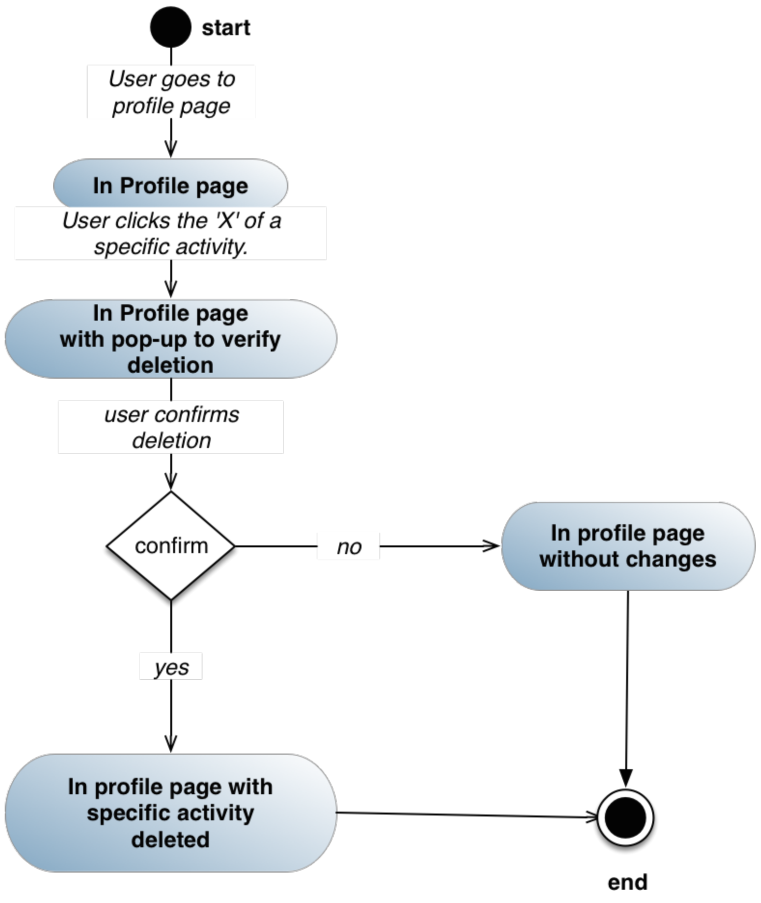

start
User goes to profile page
In Profile page
User clicks the 'X' of a specific activity.
In Profile page with pop-up to verify deletion
user confirms deletion
confirm
no
In profile page without changes
yes
In profile page with specific activity deleted
end

image 10.7: Delete Activity State Diagram

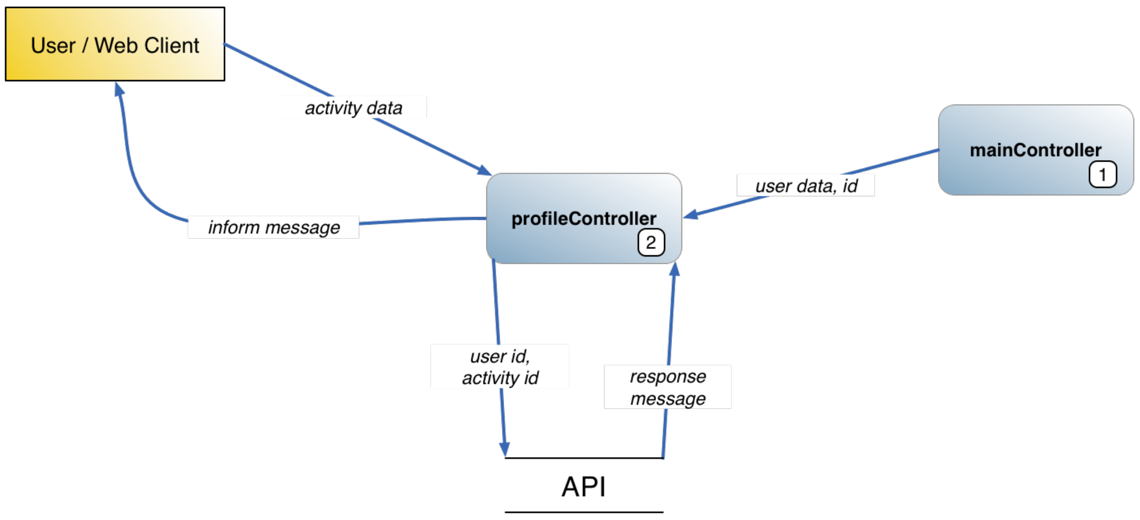


image 10.8: Delete Activity Data Flow Diagram

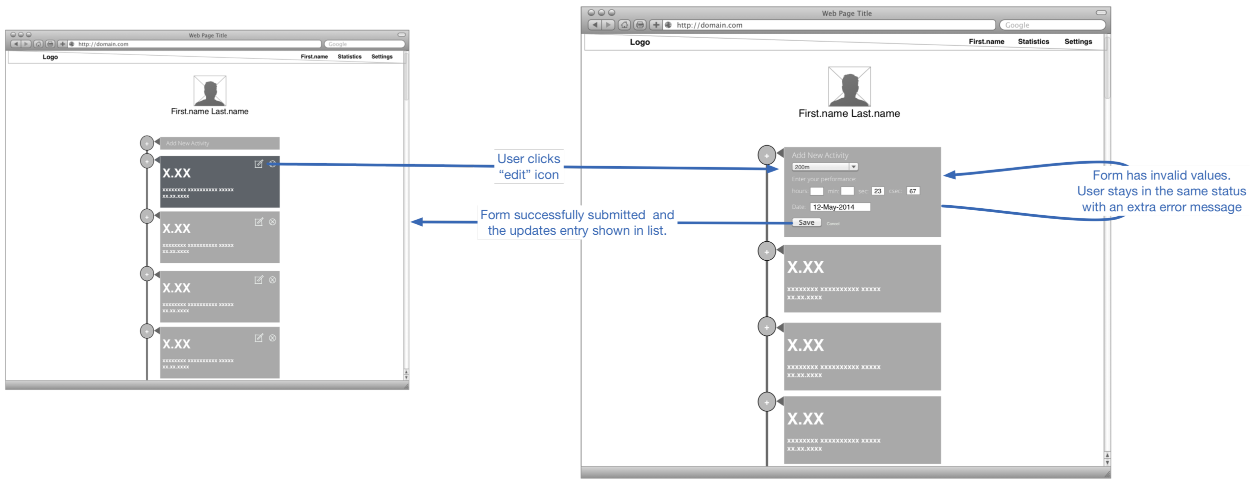

User clicks "edit" icon
Form successfully submitted and the updates entry shown in list.
Form has invalid values. User stays in the same status with an extra error message
First.name Last.name
Add New Activity
Save
X.XX

image 10.9: Edit Activity Wireframe

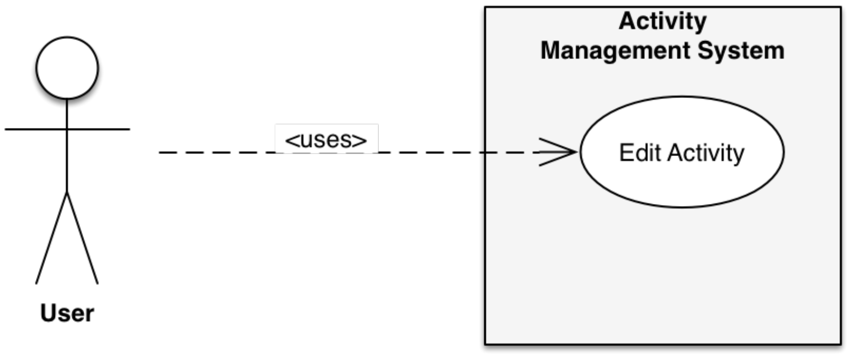


image 10.10: Edit Activity Use Case

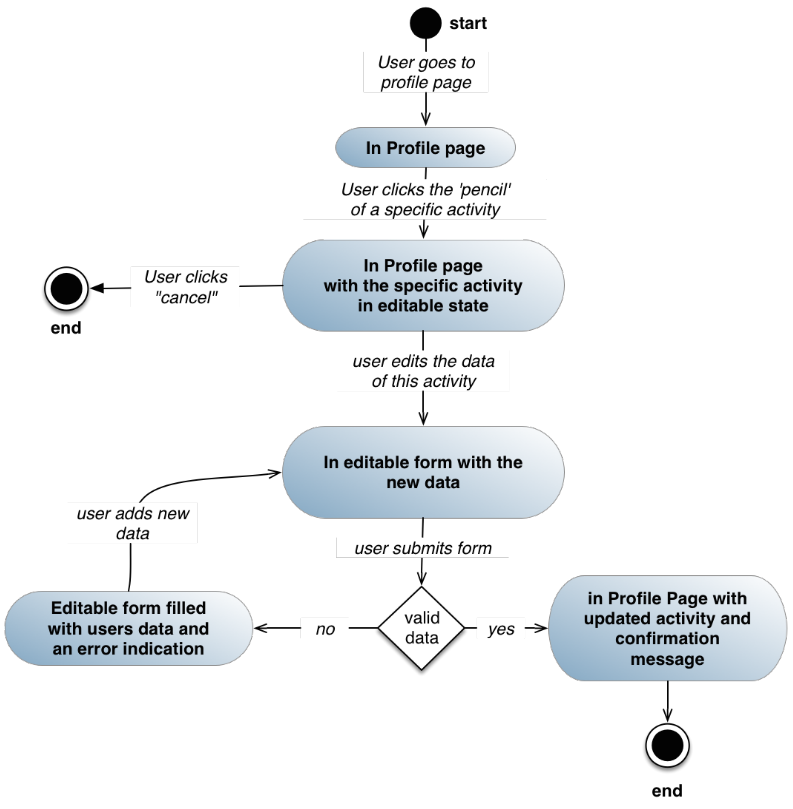

start
User goes to profile page
In Profile page
User clicks the 'pencil' of a specific activity
In Profile page with the specific activity in editable state
User clicks "cancel"
end
user edits the data of this activity
In editable form with the new data
user adds new data
user submits form
Editable form filled with users data and an error indication
no
valid data
yes
in Profile Page with updated activity and confirmation message
end

image 10.11: Edit Activity State Diagram

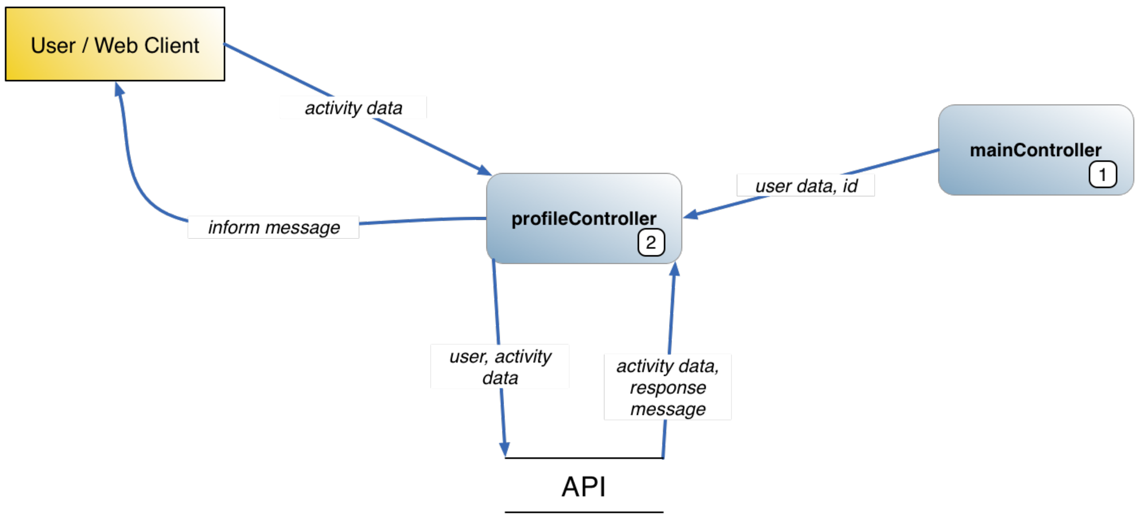


image 10.12: Edit Activity Data Flow Diagram

6.10.4 Sprint review

This Sprint had a lot of difficulties because it was our first hands on experience with AngularJS, two-way data binding and MVC logic in the web client. Except of that the benefits from this transition is visible already from this early point. We get rid a lot of code, the business logic is much clearer now, and the whole app is more maintainable and easy to understand.

6.11 Sprint 11

6.11.1 Sprint planning

In this Sprint we will transform the client-side implementation for Settings page, replacing all the javascript logic and UI manipulation with AngularJS. This should not affect the User Interface and all these changes should be seamless to the end user. In this Sprint we also implemented the change password functionality, which requires a confirmation step via email link that is send to the user.

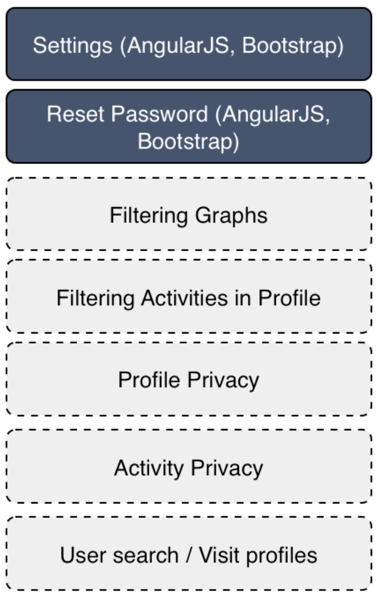

Settings (AngularJS, Bootstrap)
Reset Password (AngularJS, Bootstrap)
Filtering Graphs
Filtering Activities in Profile
Profile Privacy
Activity Privacy
User search / Visit profiles

image 11.0: Backlog status for Planning of Sprint 11

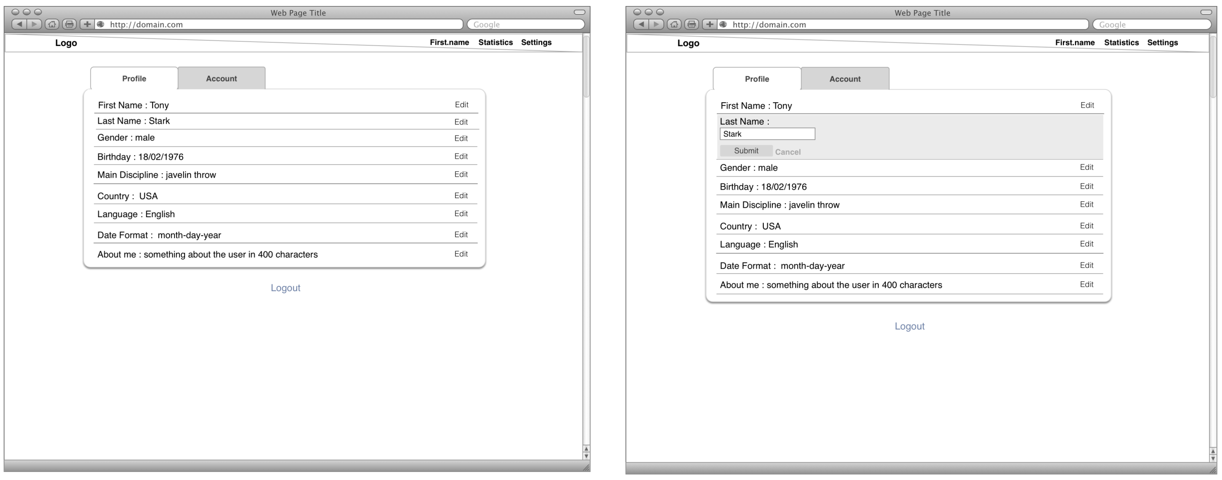
6.11.2 Stand up

We had to change two different tabs with all their functionality. All the given fields are editable and follow the same interaction design pattern. All of them become editable inside the area where there already exists, and user can insert new data, choose between options or upload files per case.

The first tab is the Profile Settings tab. The options that are given to user to edit, in this tab, are the profile picture, the first and the last name, the main discipline, the gender, the birthday, the country and ‘about me’ text. We are expecting to have difficulties on image upload through angularJS, but he rest of the fields are seems easy to implemented.

The second tab is the Account Settings tab. Here we have the option to change password, the language of our application, the format in which we, as users, want to see the dates the username. Here we expect to have difficulties with username in backend because it should be unique, and also be used in urls in user’s page. This is very handy for the user’s, in order to visit each other’s profiles, or share theirs in other social media, but we want to implement this functionality without compromise our system’s security.

6.11.3 Diagrams


Web Page Title

http://domain.com

Logo First.name Statistics Settings

Profile Account

First Name : Tony	Edit
Last Name : Stark	Edit
Gender : male	Edit
Birthday : 18/02/1976	Edit
Main Discipline : javelin throw	Edit
Country : USA	Edit
Language : English	Edit
Date Format : month-day-year	Edit
About me : something about the user in 400 characters	Edit

Logout

First Name : Tony	Edit
Last Name : Stark Submit Cancel	
Gender : male	Edit
Birthday : 18/02/1976	Edit
Main Discipline : javelin throw	Edit
Country : USA	Edit
Language : English	Edit
Date Format : month-day-year	Edit
About me : something about the user in 400 characters	Edit

Logout

image 11.1: Settings Page User Interface with and without editable form

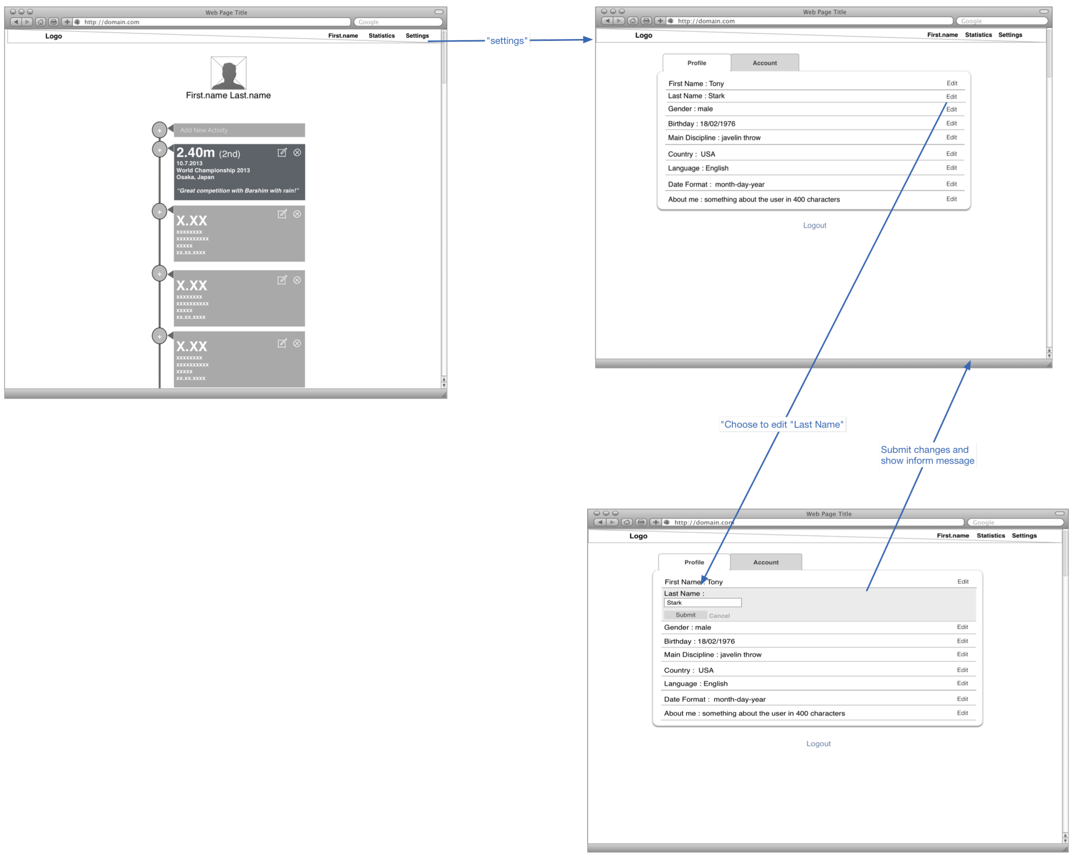


image 11.2: Edit Setting Wireframe

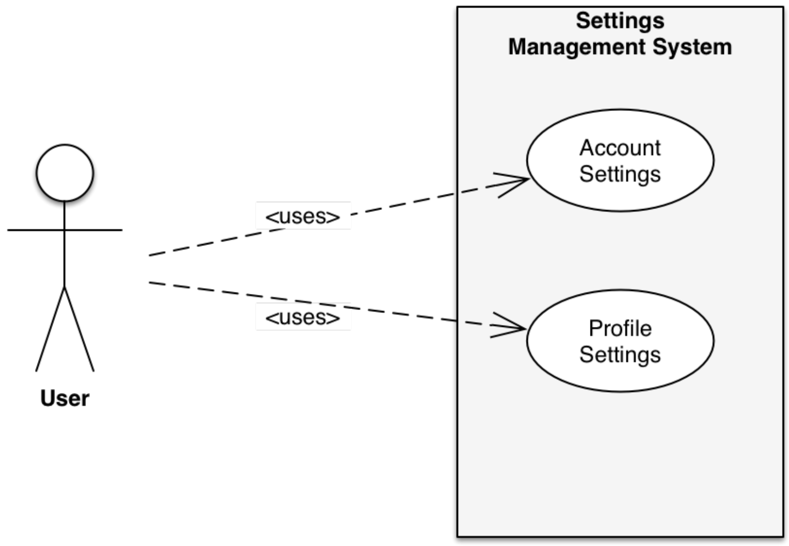


image 11.3: Edit Settings Use Case

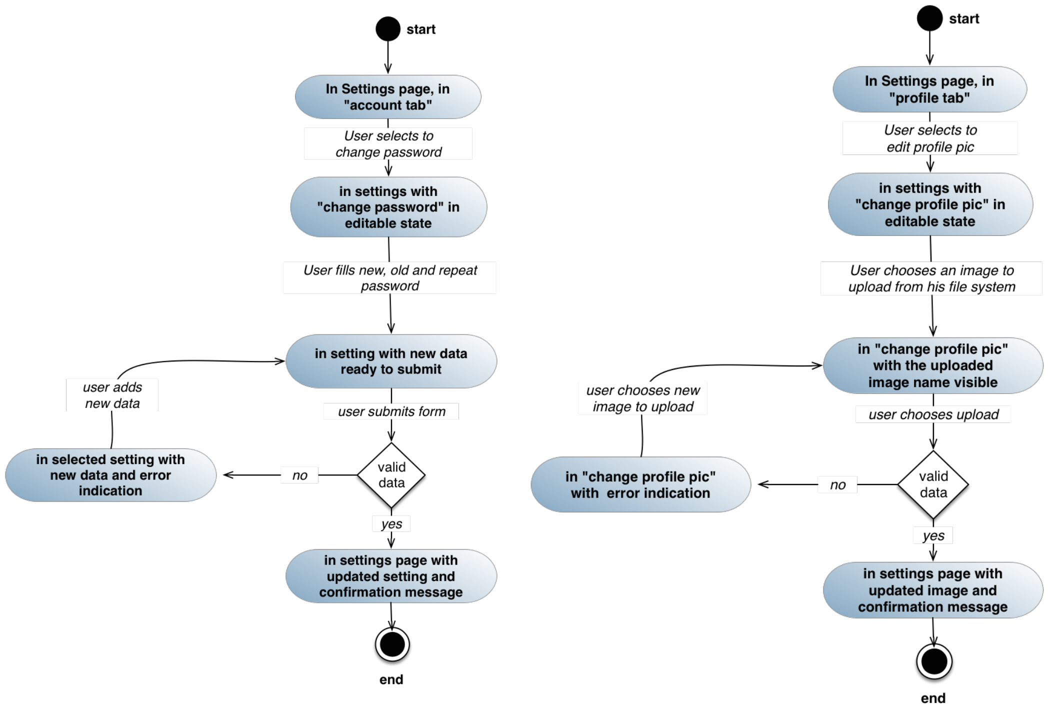

start
In Settings page, in "account tab"
User selects to change password
in settings with "change password" in editable state
User fills new, old and repeat password
in setting with new data ready to submit
user submits form
valid data
no
in selected setting with new data and error indication
user adds new data
yes
in settings page with updated setting and confirmation message
end
start
In Settings page, in "profile tab"
User selects to edit profile pic
in settings with "change profile pic" in editable state
User chooses an image to upload from his file system
in "change profile pic" with the uploaded image name visible
user chooses upload
valid data
no
in "change profile pic" with error indication
user chooses new image to upload
yes
in settings page with updated image and confirmation message
end

image 11.4: Change Password and Profile picture State Diagram

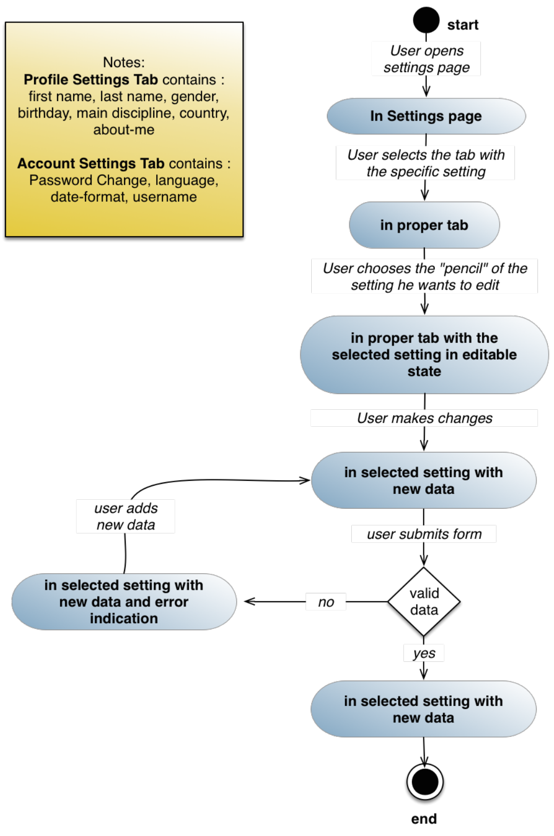

Notes:
Profile Settings Tab contains : first name, last name, gender, birthday, main discipline, country, about-me
Account Settings Tab contains : Password Change, language, date-format, username
start
User opens settings page
In Settings page
User selects the tab with the specific setting
in proper tab
User chooses the "pencil" of the setting he wants to edit
in proper tab with the selected setting in editable state
User makes changes
in selected setting with new data
user submits form
valid data
no
in selected setting with new data and error indication
user adds new data
yes
in selected setting with new data
end

image 11.5: Change Setting in Profile or Account tab State Diagram

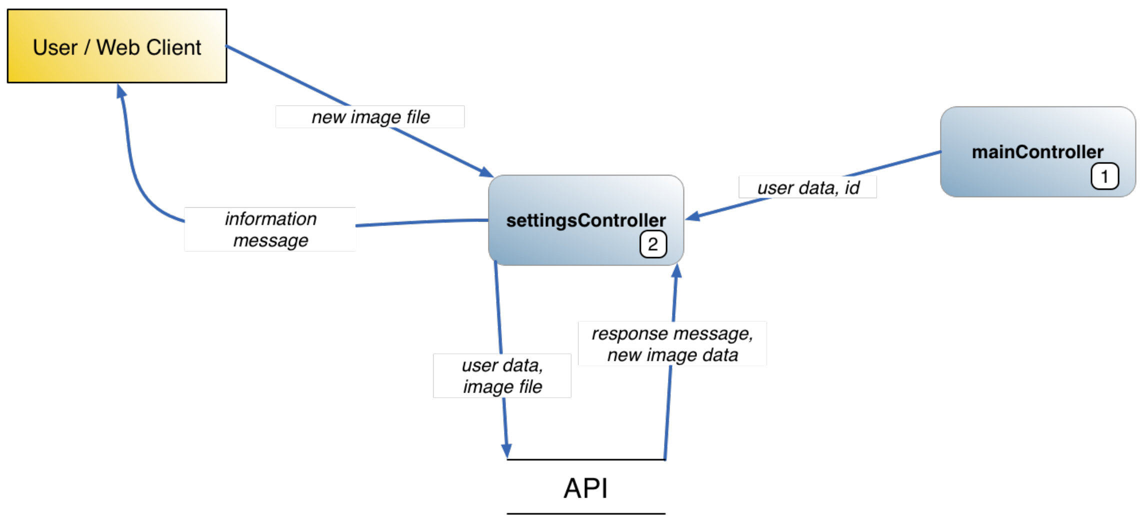


image 11.6: Change Image Data Flow Diagram

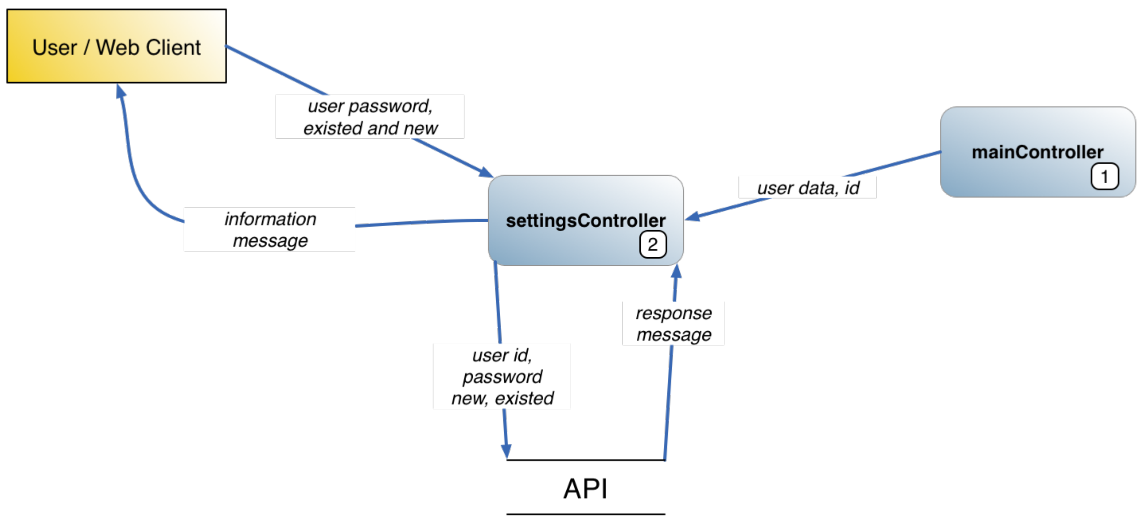
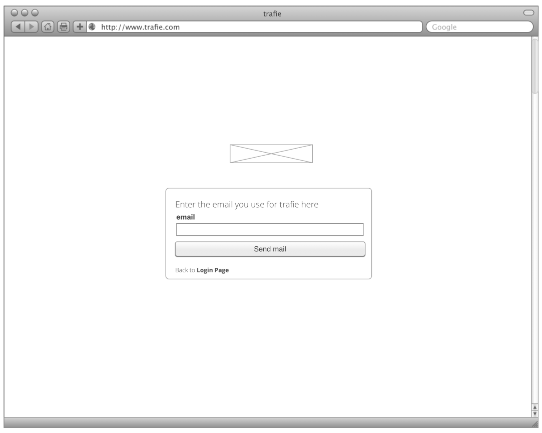


image 11.7: Change Password Data Flow Diagram



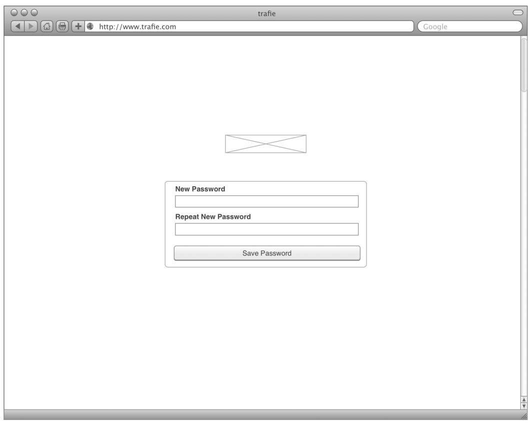


image 11.8: User Interface of Request and Change password

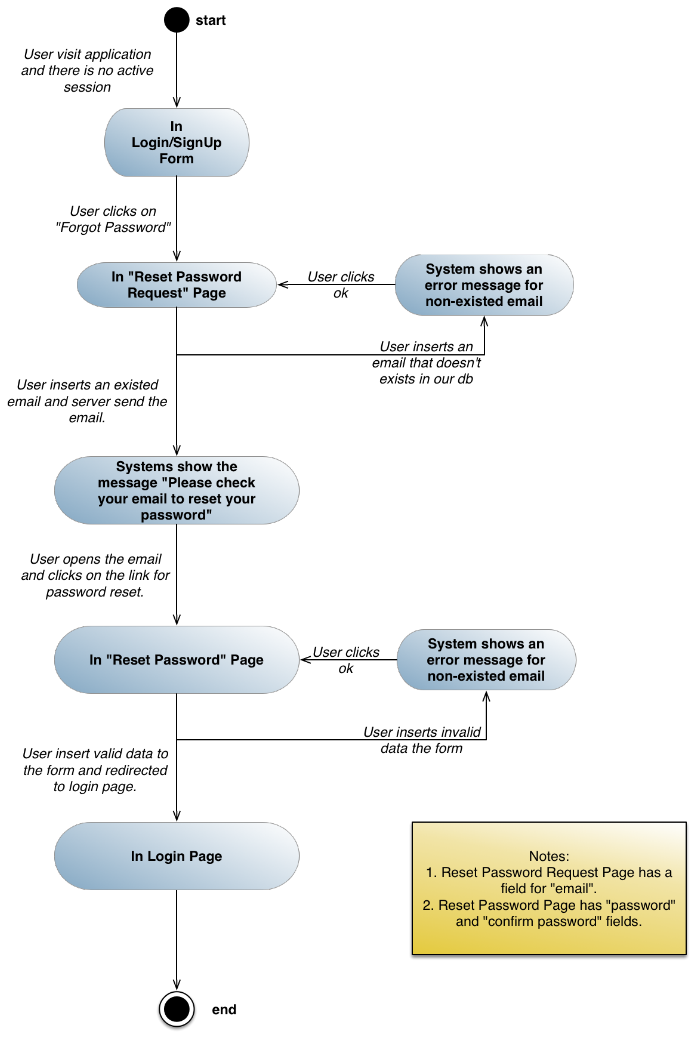

start
User visit application and there is no active session
In Login/SignUp Form
User clicks on "Forgot Password"
In "Reset Password Request" Page
User clicks ok
System shows an error message for non-existed email
User inserts an email that doesn't exists in our db
User inserts an existed email and server send the email.
Systems show the message "Please check your email to reset your password"
User opens the email and clicks on the link for password reset.
In "Reset Password" Page
User clicks ok
System shows an error message for non-existed email
User inserts invalid data the form
User insert valid data to the form and redirected to login page.
In Login Page
Notes:
1. Reset Password Request Page has a field for "email".
2. Reset Password Page has "password" and "confirm password" fields.
end

image 11.9: Reset password State diagram

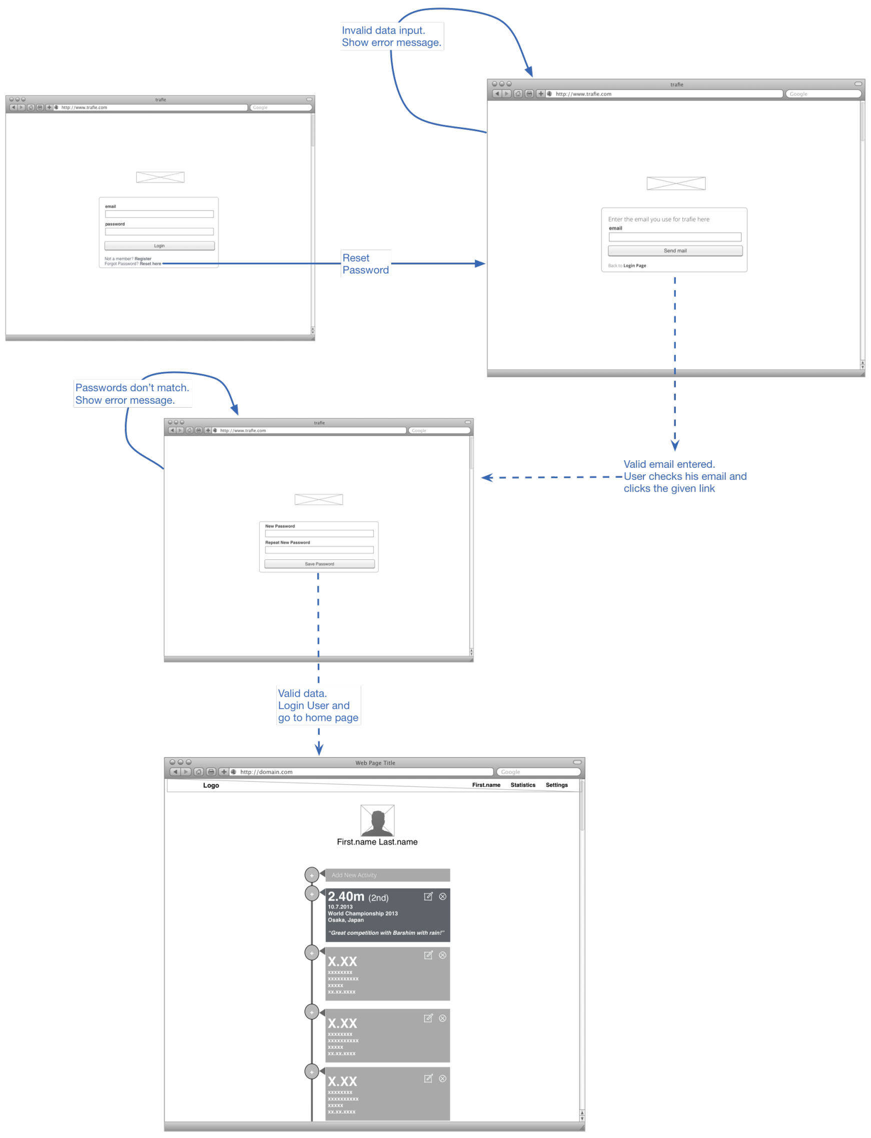


image 11.10: Reset Password Wireframe

6.11.4 Sprint review

Although the difficulties with special options such as profile picture update and username, we completed this Sprint on time. Now we have almost implement our whole application with the software stack we want, completing and the second important pivot, and for the next two Sprints we are going to implement new functionality and make some optimisations.

6.12 Sprint 12

6.12.1 Sprint planning

In this Sprint we will implement two very important features that will enhance the usability of our app. The first one is the graphs, which already existed but know we need to transfer their logic in AngularJS implementation. The second one is the filtering of our data. Our data will be filtered by discipline and year, and filtering will apply in profile page's activities but also in graphs.

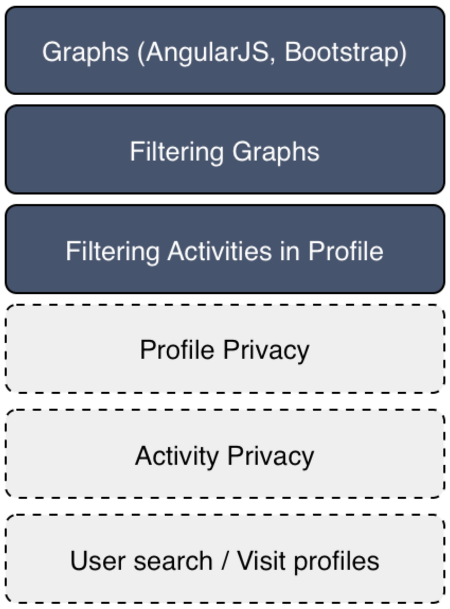

Graphs (AngularJS, Bootstrap)
Filtering Graphs
Filtering Activities in Profile
Profile Privacy
Activity Privacy
User search / Visit profiles

image 12.0: Backlog status for Planning of Sprint 12

6.12.2 Stand up

For graphs charts we will replace Google Charts with C3.js library which is based on D3.js library and is more lightweight, flexible and has more options which is a great thing for future use and feature additions. Our Statistics controller will request all activities' data exactly in the same way we do in Profile controller.

After we get the response from the server we will group them in arrays, creating in parallel the required variables for filtering. Then we show them using our charts library. The filtering variables are the user's disciplines and the year in which the user has saved activities. We can also filter our data chronically from a sub-chart which visualise all our data. We can select a time period on them just by click and drag our cursor on it. This functionality also support touch events and can be used by our users in a tablet.

We follow the same logic in Profile controller in order to create the filters for the activities. After fetching all the data we algorithmically get the dates and the disciplines that appeared in them and create the variables that are going to filter our activities list.

6.12.3 Diagrams

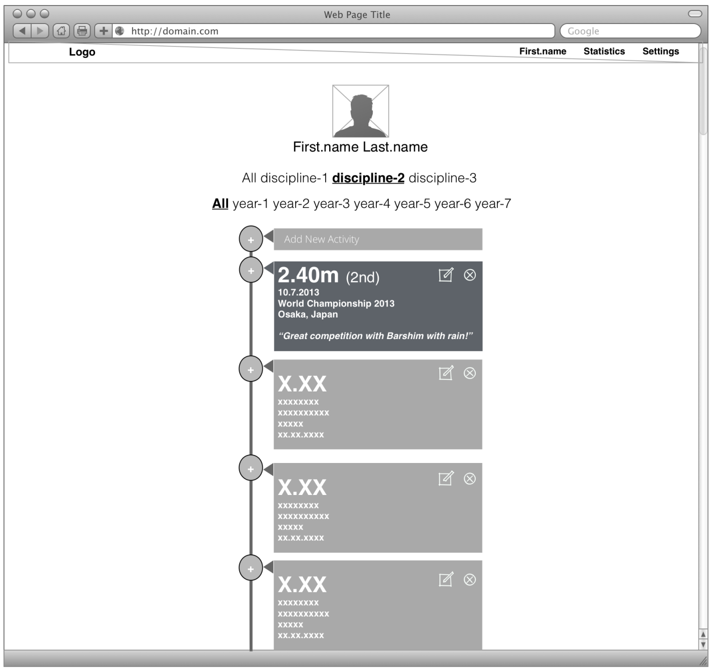


image 12.1: User Interface of Profile Page with filtering options in discipline and year

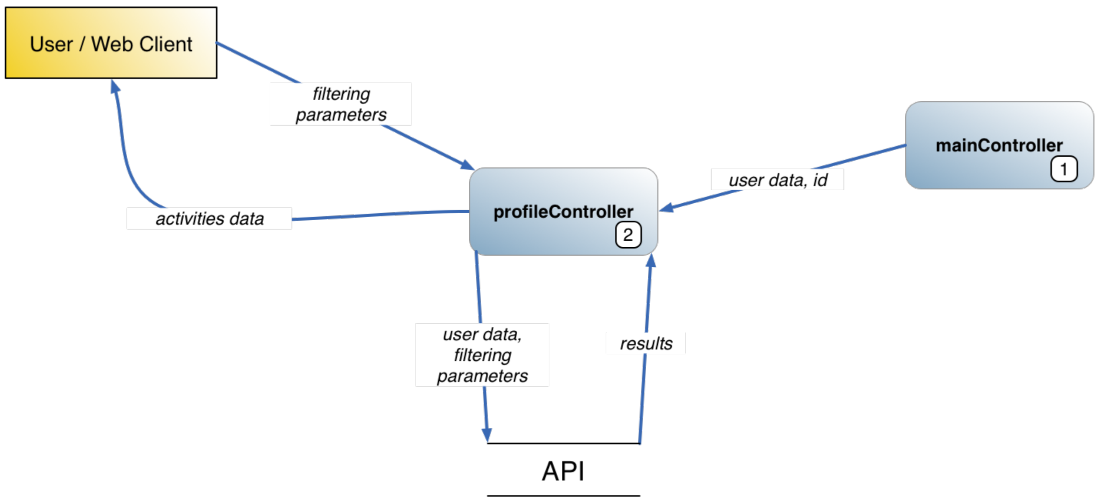


image 12.2: Filtering activities in profile page Data Flow Diagram

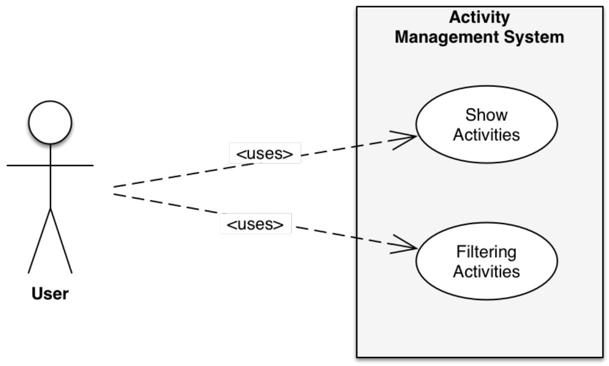


image 12.3: Show and filtering activities Use Case

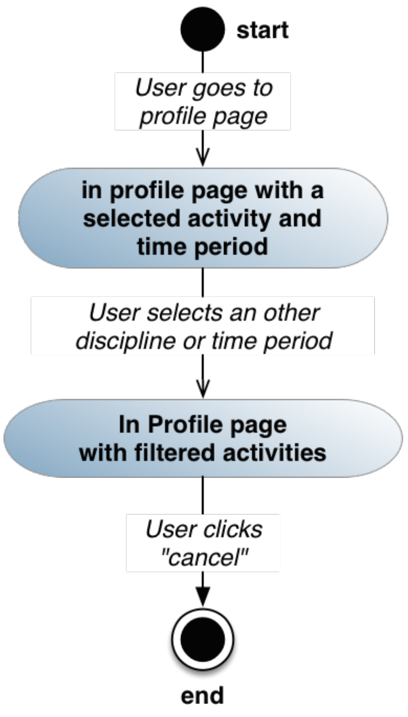

start
User goes to profile page
in profile page with a selected activity and time period
User selects an other discipline or time period
In Profile page with filtered activities
User clicks "cancel"
end

image 12.4: Filtering activities in Profile page Data Flow Diagram

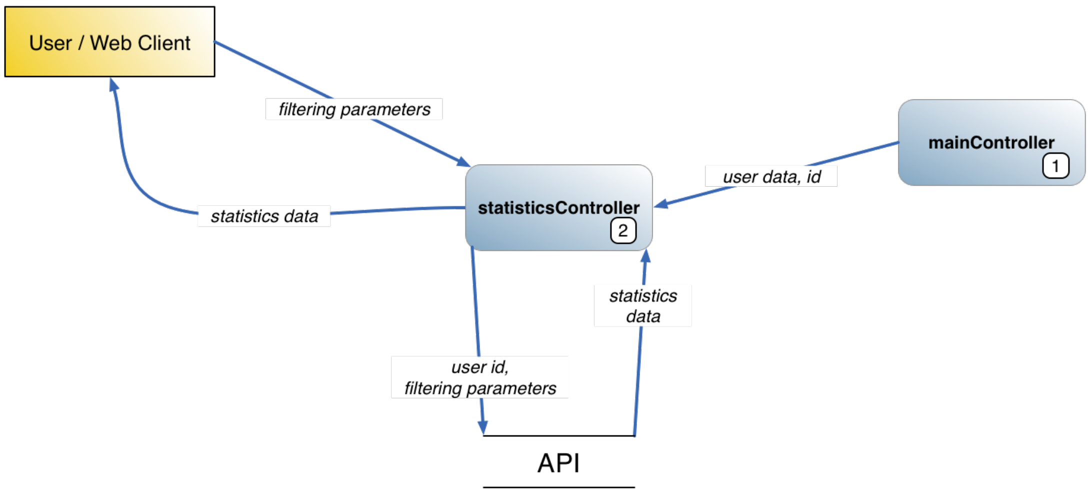


image 12.5: Filtering statistics Data Flow Diagram

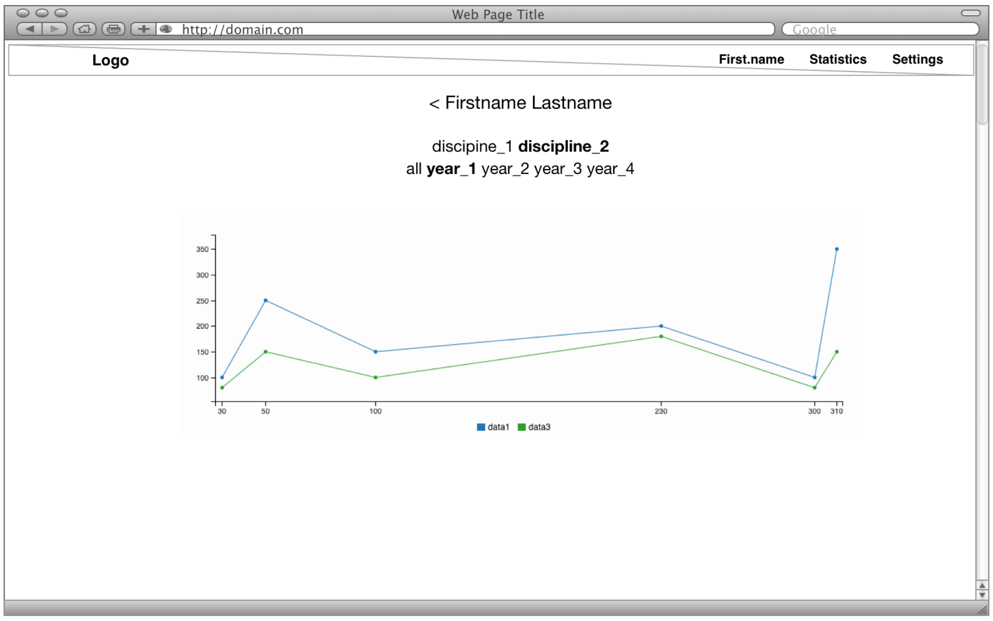


image 12.6: User Interface of Statistics page with filtering discipline and year option

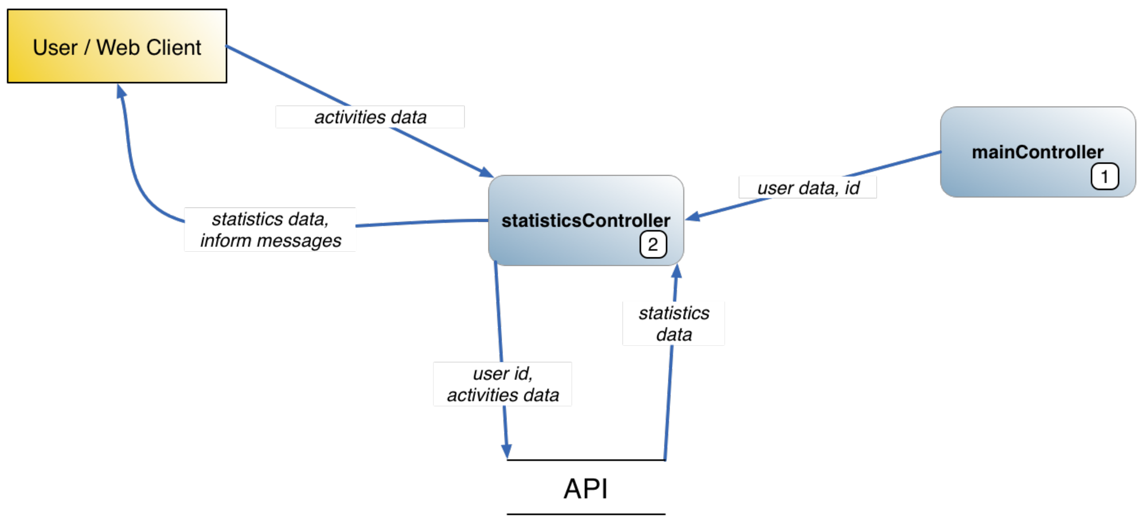


image 12.7: Show activities as graphs Data Flow Diagram

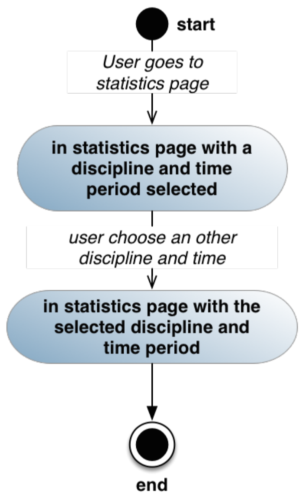


image 12.8: Filtering statistics page State Diagram

6.12.4 Sprint review

This was a very useful and important Sprint. We finalise the implementation of our app with the software stack and also add filtering which is a very important feature that enhances the overall user experience. Although we had to use one more time a new library, we deliver on time. This happened because with AngularJS the data manipulation has become very easy and let us time to work on more demanding task.

6.13 Sprint 13

6.13.1 Sprint planning

In this Sprint we are going to implement a kind of social feature. This is the public pages, which means that a user can search and visit some other's user profile page and see it's activities and charts. This functionality rise the need to define some privacy options and also create a search field in order to allow users to search each other. We will implement privacy settings for user profile and also for each activity. So a user can have a public profile but not show all of it's activities.

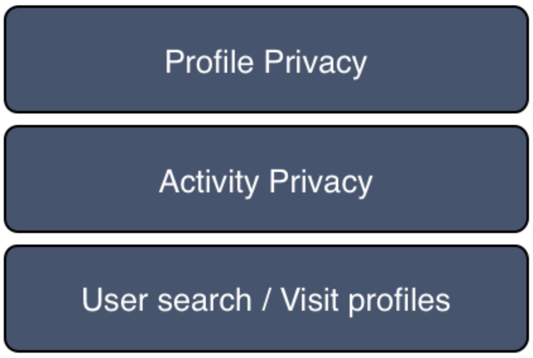


image 13.0: Backlog status for Planning of Sprint 13

6.13.2 Stand up

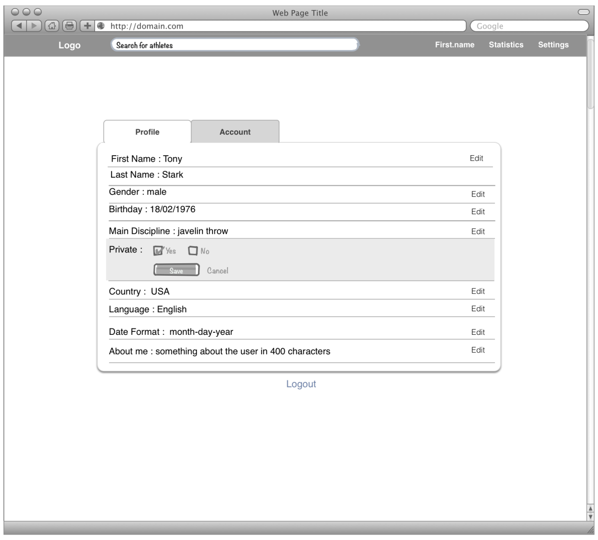
Firstly we created a search field that will search users, as we type, and brings us the top 10 results that matches their search. The search parameters is the name and last name, but in the results' list we also see the users' main discipline and country. This makes easier for the user who makes the search to recognise the athlete whom actually search for.

Public pages is a feature that allows user page to be visible to every one, even if he is not a registered member of our app. So if the user allows to make his profile public, everyone can visit it via a specific url like 'www.domain.com/username'. If a username hasn't been defined the profile is still visible using the user's id instead of it. If user chooses to have a private profile, it isn't visible neither by application's visitors, nor by registered users. Also if the user has a private profile, he doesn't appeared in the search results.

Each activity has a boolean field which defines the privacy of the specific activity. In this way even if a profile is public, each activity is visible to public only if the user chooses that.

The privacy option for the profile can be changed from Settings page, in Profile Settings tab. The privacy option for each activity can changed inside each activity in profile page. We think that this setting is very import, so we allow users to change them without edit the whole activity.

6.13.3 Diagrams



First Name : Tony Edit
Last Name : Stark
Gender : male Edit
Birthday : 18/02/1976 Edit
Main Discipline : javelin throw Edit
Private : [✓ Yes] No
Country : USA Edit
Language : English Edit
Date Format : month-day-year Edit
About me : something about the user in 400 characters Edit

image 13.1:User Interface for profile privacy in settings page

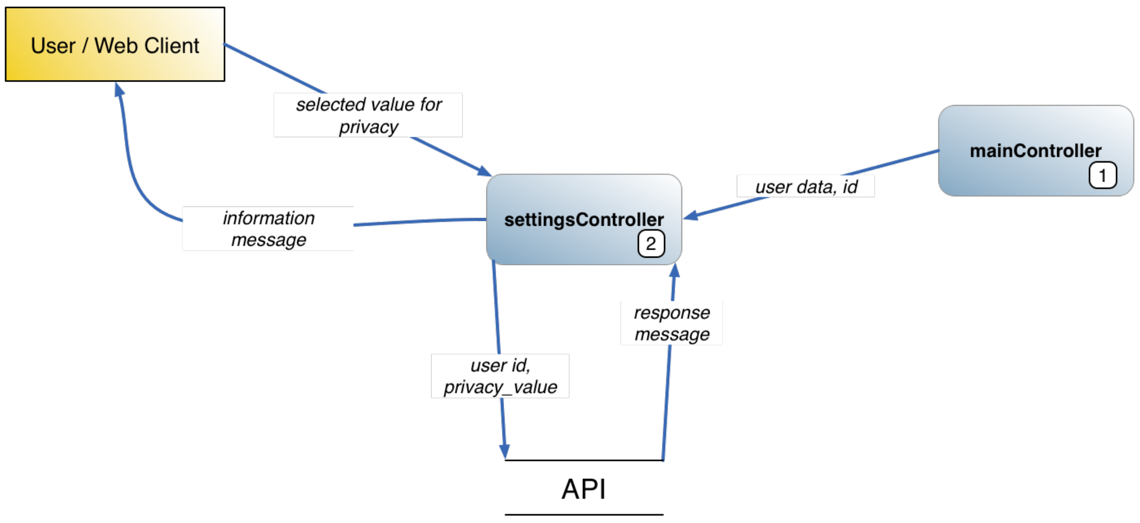


image 13.2: Change privacy for profile Data Flow Diagram

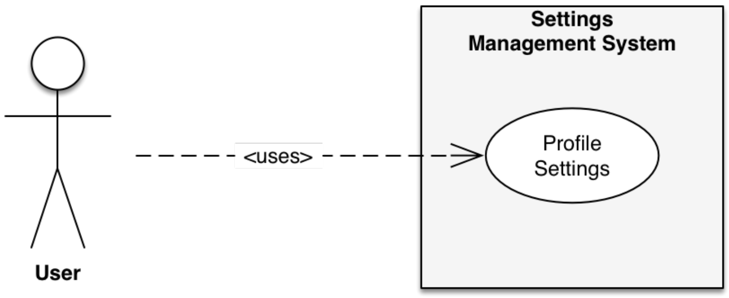


image 13.3: Change privacy in settings Use Case

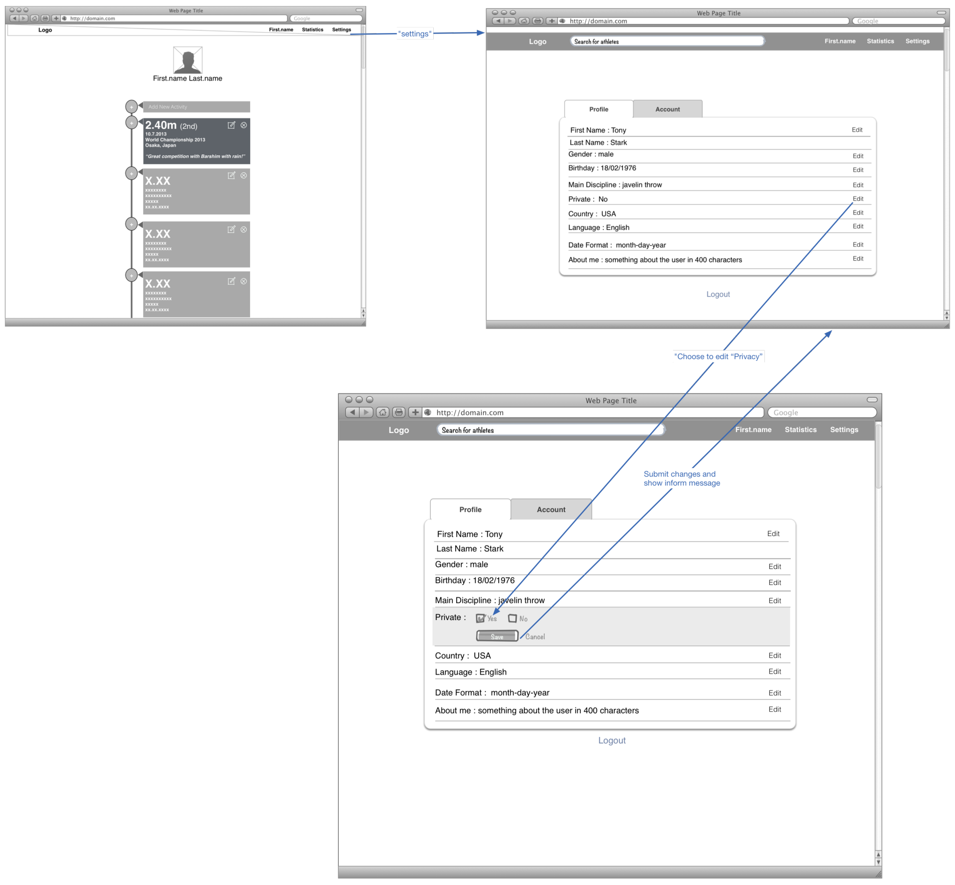


image 13.4: Change Privacy for profile Wireframe

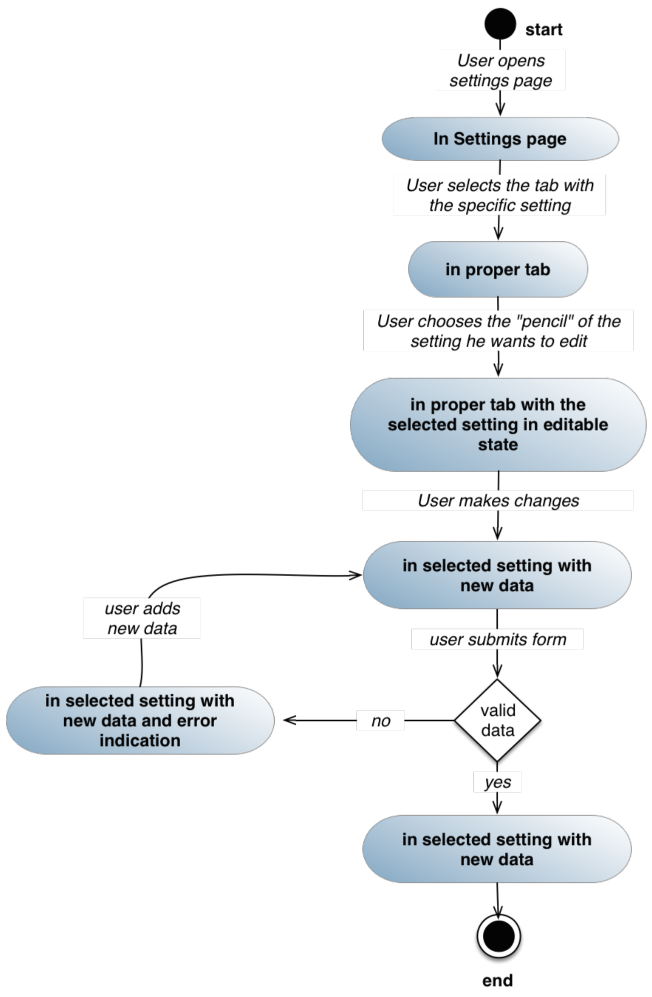

start
User opens settings page
In Settings page
User selects the tab with the specific setting
in proper tab
User chooses the "pencil" of the setting he wants to edit
in proper tab with the selected setting in editable state
User makes changes
in selected setting with new data
user adds new data
user submits form
valid data
no
in selected setting with new data and error indication
yes
in selected setting with new data
end

image 13.5: Change Profile Privacy State Diagram

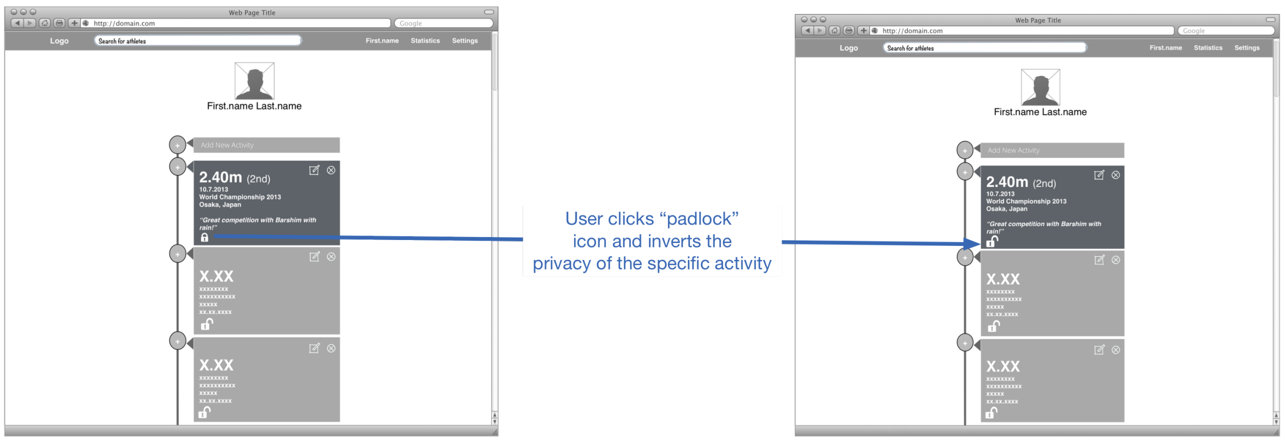


image 13.6: Change privacy in activity Wireframe

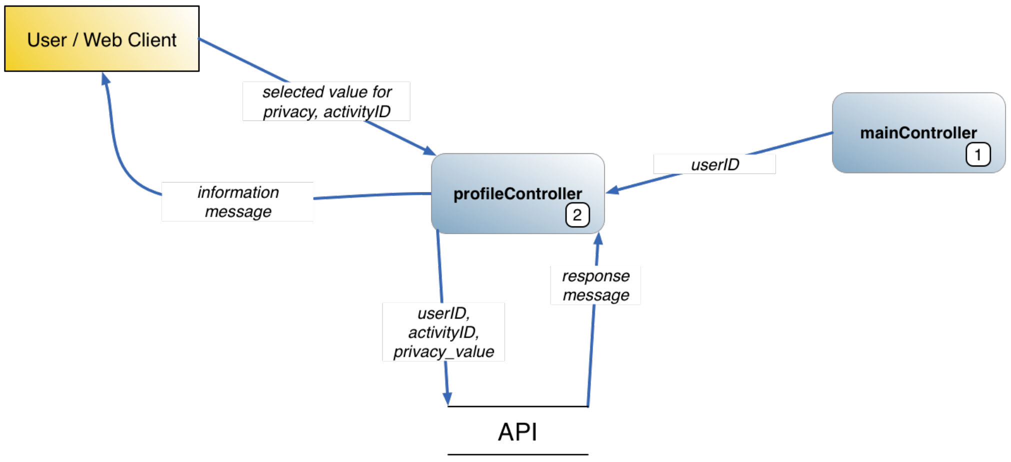


image 13.7: Change privacy for activity Data Flow Diagram

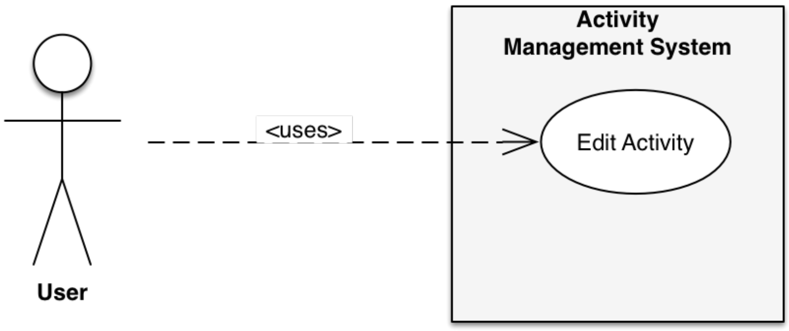
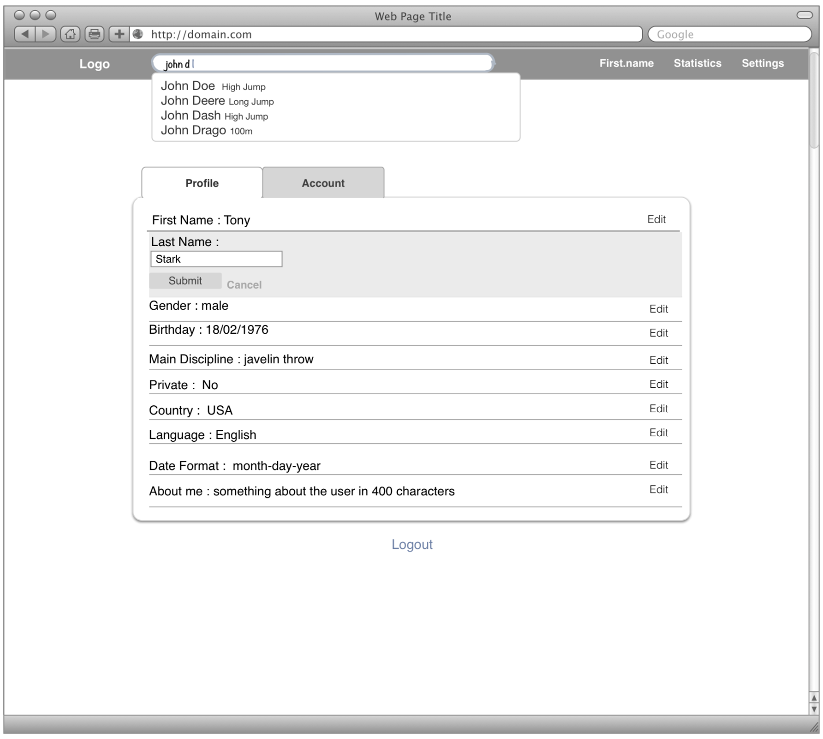


image 13.8: Change privacy in activity Use Case



john d

John Doe High Jump
John Deere Long Jump
John Dash High Jump
John Drago 100m

First Name : Tony
Last Name : Stark
Gender : male
Birthday : 18/02/1976
Main Discipline : javelin throw
Private : No
Country : USA
Language : English
Date Format : month-day-year
About me : something about the user in 400 characters

image 13.9: User Interface of search User

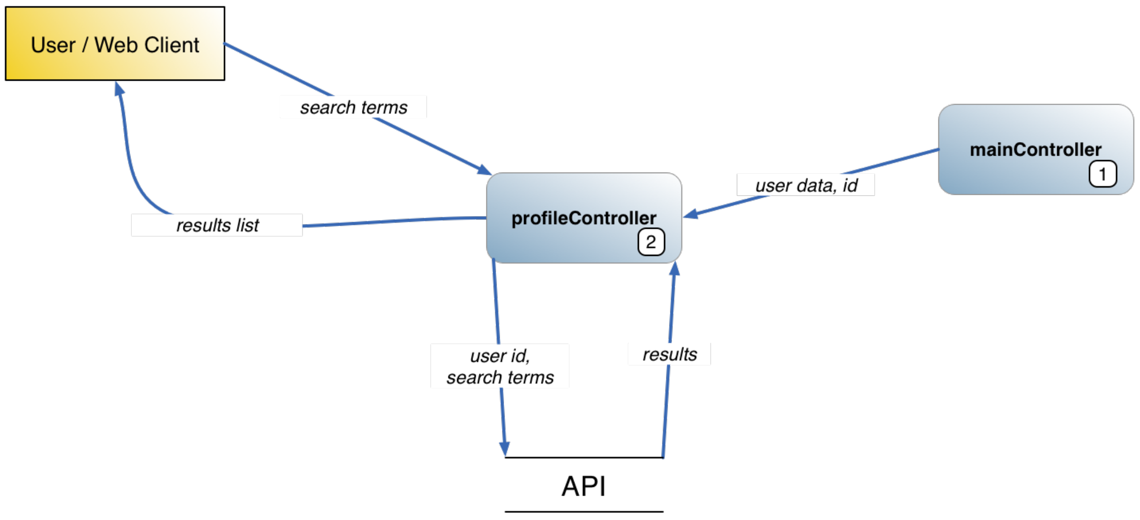


image 13.10: Search User Data Flow

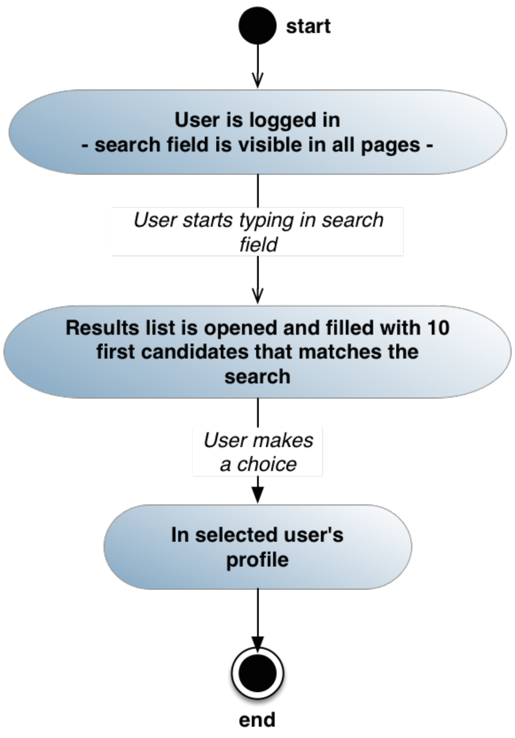

start
User is logged in
- search field is visible in all pages -
User starts typing in search field
Results list is opened and filled with 10 first candidates that matches the search
User makes a choice
In selected user's profile
end

image 13.11: Search user State diagram

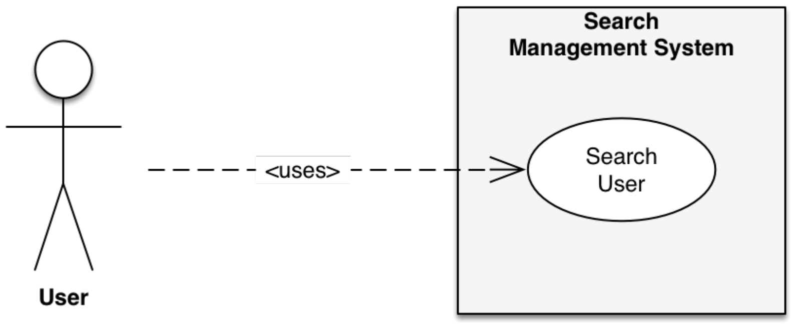


image 13.12: Search User State Diagram

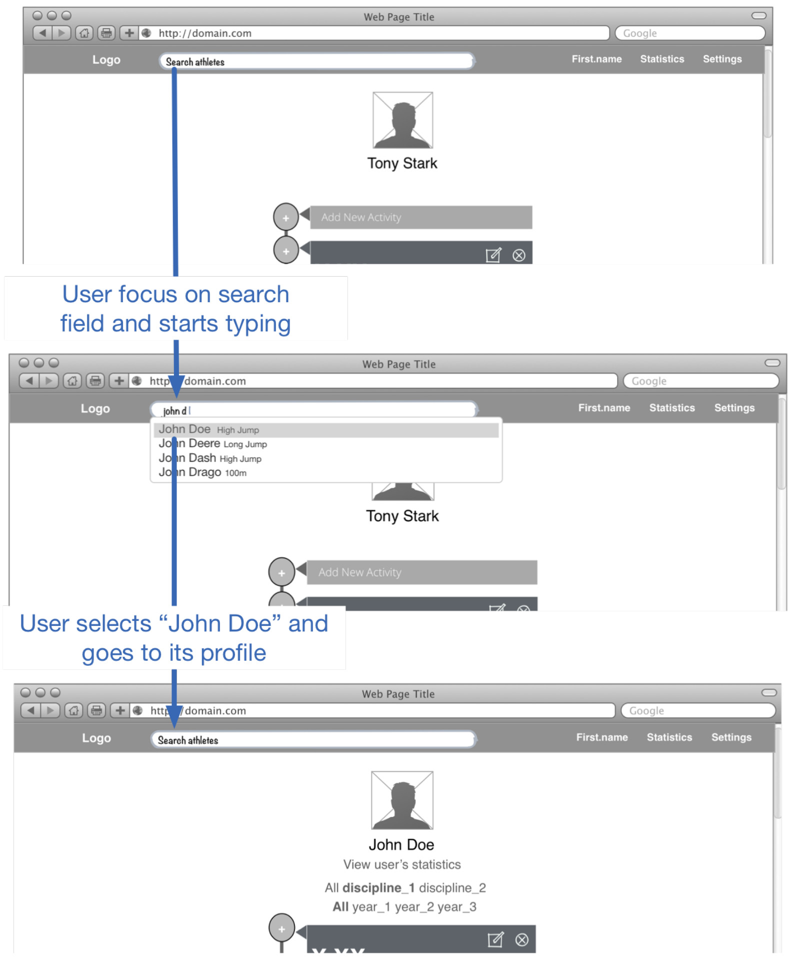


image 13.13: Search User Wireframe

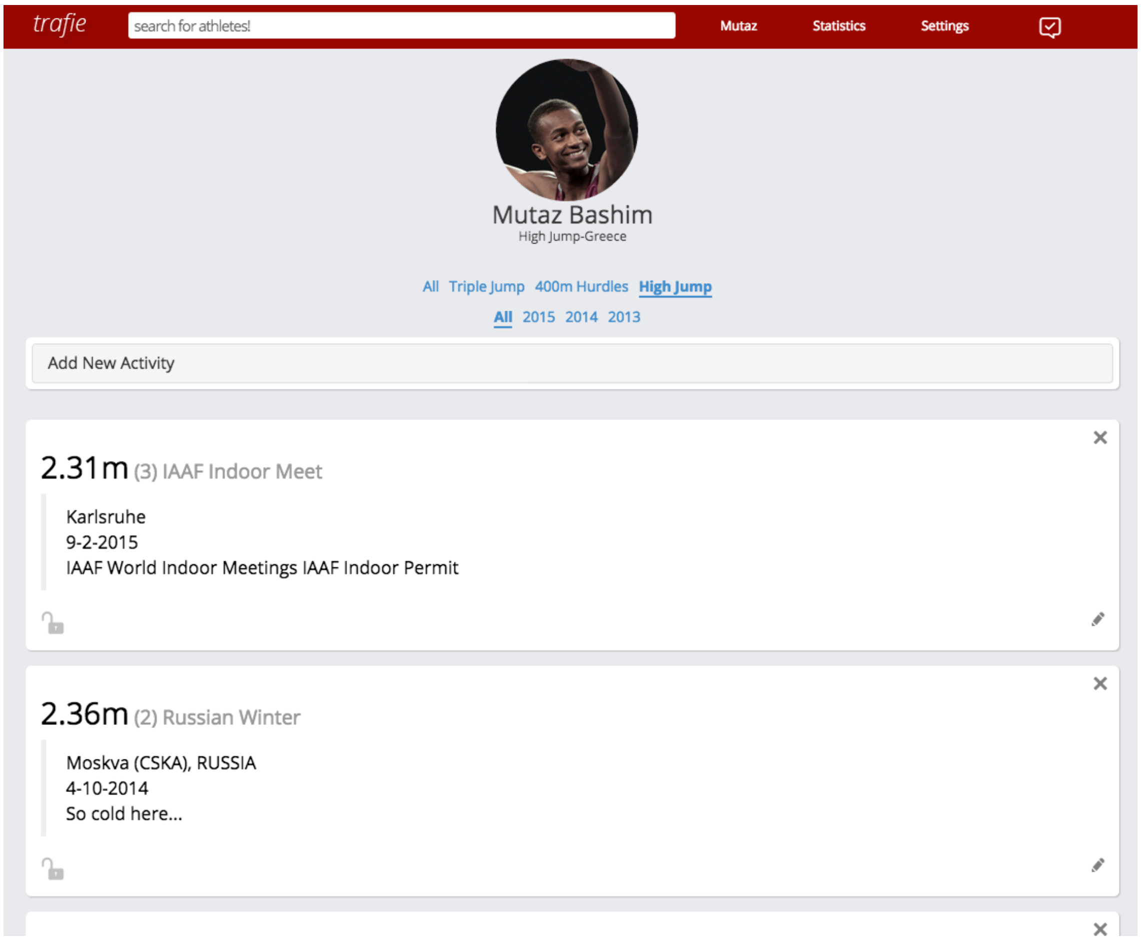
6.13.4 Sprint review

In this Sprint we added a very important functionality that allow users to share their performances and profile, taking care of privacy issues, and also search for other users. With this functionality we believe that we have a first version of a complete product, that has meaning and can be launched to end users.

6.14 Final User Interface

During the whole process the user interface followed the UI design that appeared in previous Sprints. After all iterations the final design, (for desktop, tablet and mobiles) of our application is as shown in the following images.

Desktop



2.31m (3) IAAF Indoor Meet
Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

2.36m (2) Russian Winter
Moskva (CSKA), RUSSIA
4-10-2014
So cold here...

image 14.1 : Profile page on desktop version.

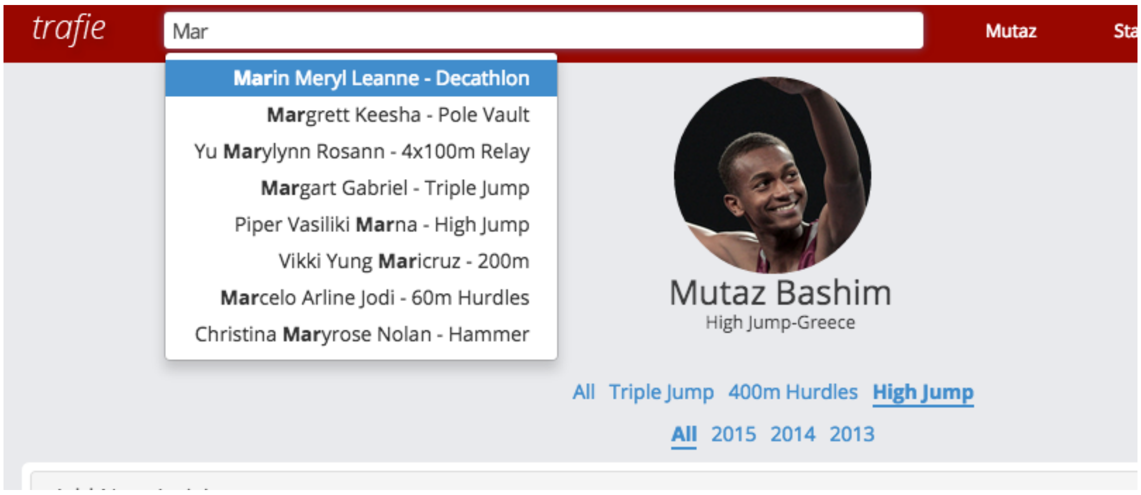
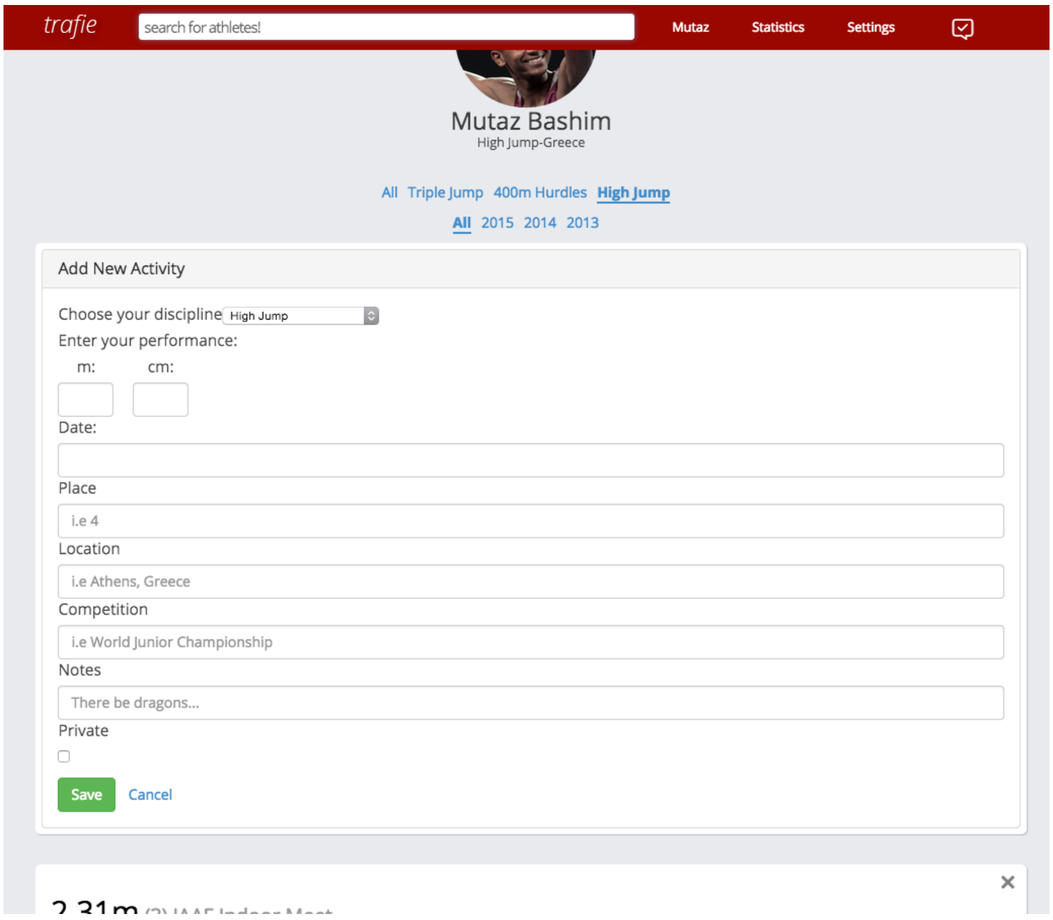

trafie
Mar
Mutaz
Marin Meryl Leanne - Decathlon
Margrett Keesha - Pole Vault
Yu Marylynn Rosann - 4x100m Relay
Margart Gabriel - Triple Jump
Piper Vasiliki Marna - High Jump
Vikki Yung Maricruz - 200m
Marcelo Arline Jodi - 60m Hurdles
Christina Maryrose Nolan - Hammer
Mutaz Bashim
High Jump-Greece
All Triple Jump 400m Hurdles High Jump
All 2015 2014 2013

image 14.2 : Search users on desktop version through search field, using autocomplete suggestion feature.



Add New Activity

Choose your discipline High Jump

Enter your performance:

m: cm:

Date:

Place

i.e 4

Location

i.e Athens, Greece

Competition

i.e World Junior Championship

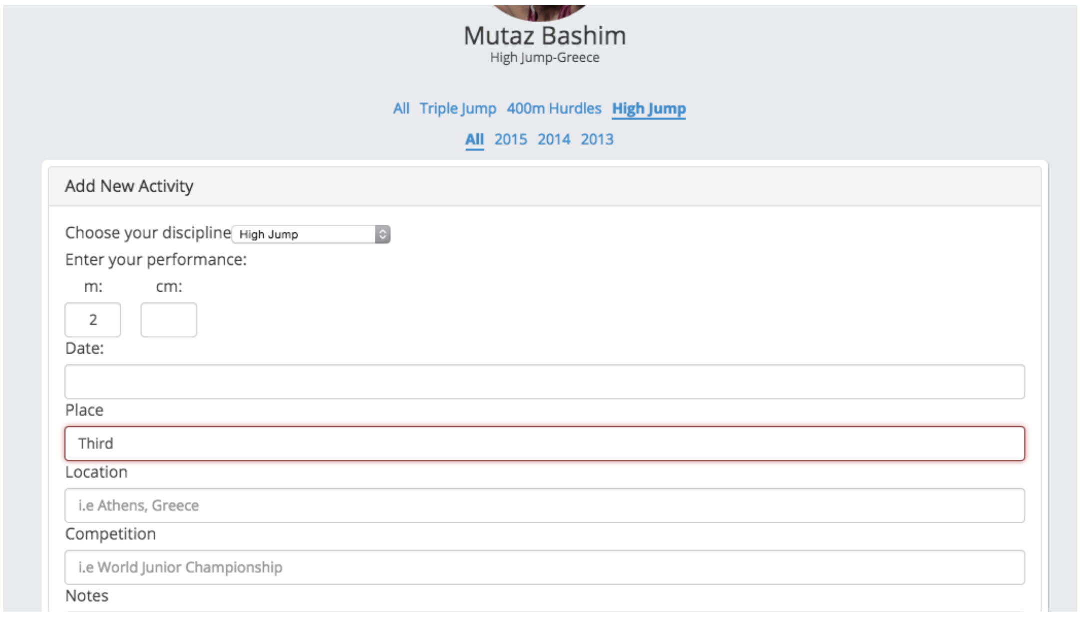
Notes

There be dragons...

Private

Save Cancel

image 14.3 : Add new activity on desktop version.

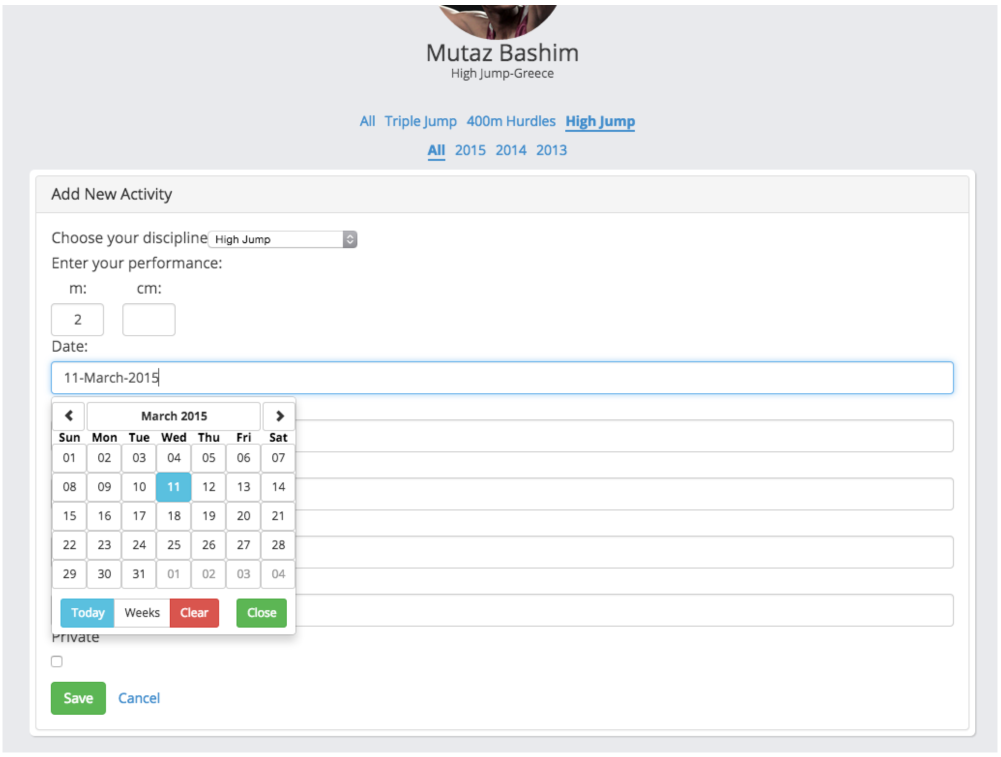

Mutaz Bashim
High Jump-Greece

All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

Add New Activity

Choose your discipline: High Jump
Enter your performance:
m: 2 cm:
Date:
Place: Third
Location: i.e Athens, Greece
Competition: i.e World Junior Championship
Notes

image 14.4 : Error shown instantly inside add activity form on desktop version.


Mutaz Bashim
High Jump-Greece

All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

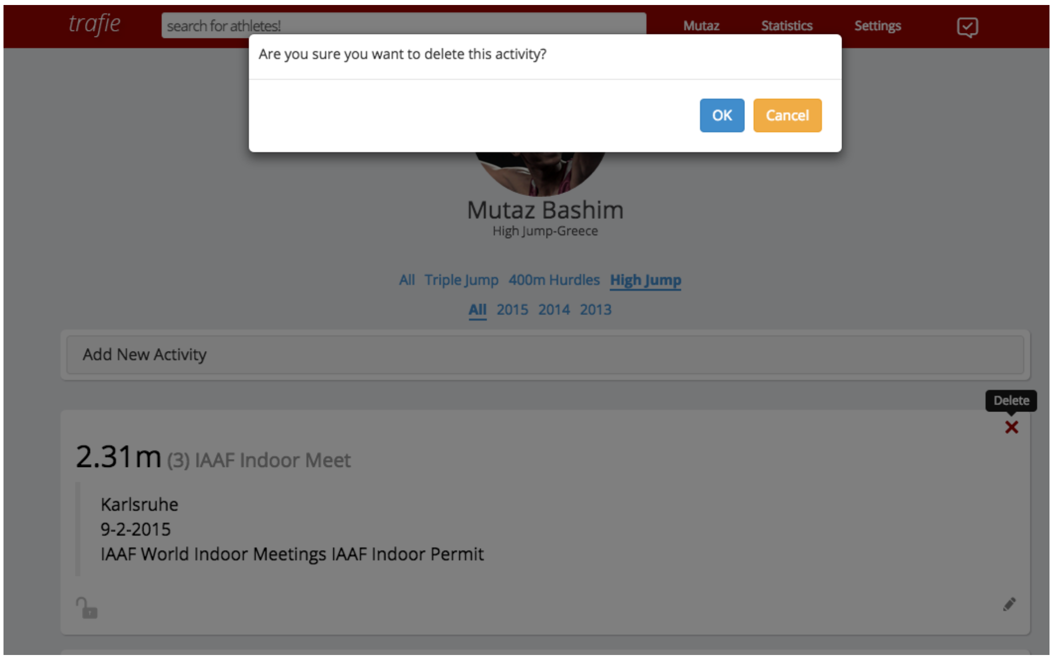
Add New Activity

Choose your discipline: High Jump
Enter your performance:
m: 2 cm:
Date: 11-March-2015

March 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
01	02	03	04	05	06	07
08	09	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	01	02	03	04

image 14.5 : Calendar picker for easy selection of a date in add activity form on desktop version.



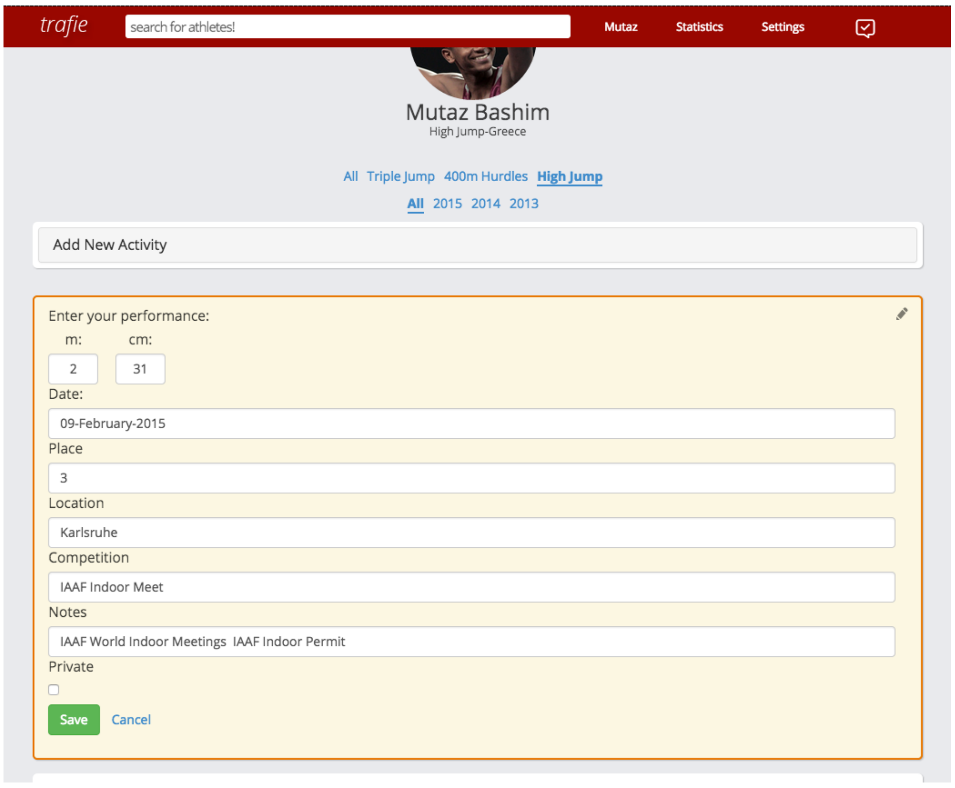
Are you sure you want to delete this activity?

Mutaz Bashim
High Jump-Greece

2.31m (3) IAAF Indoor Meet

Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

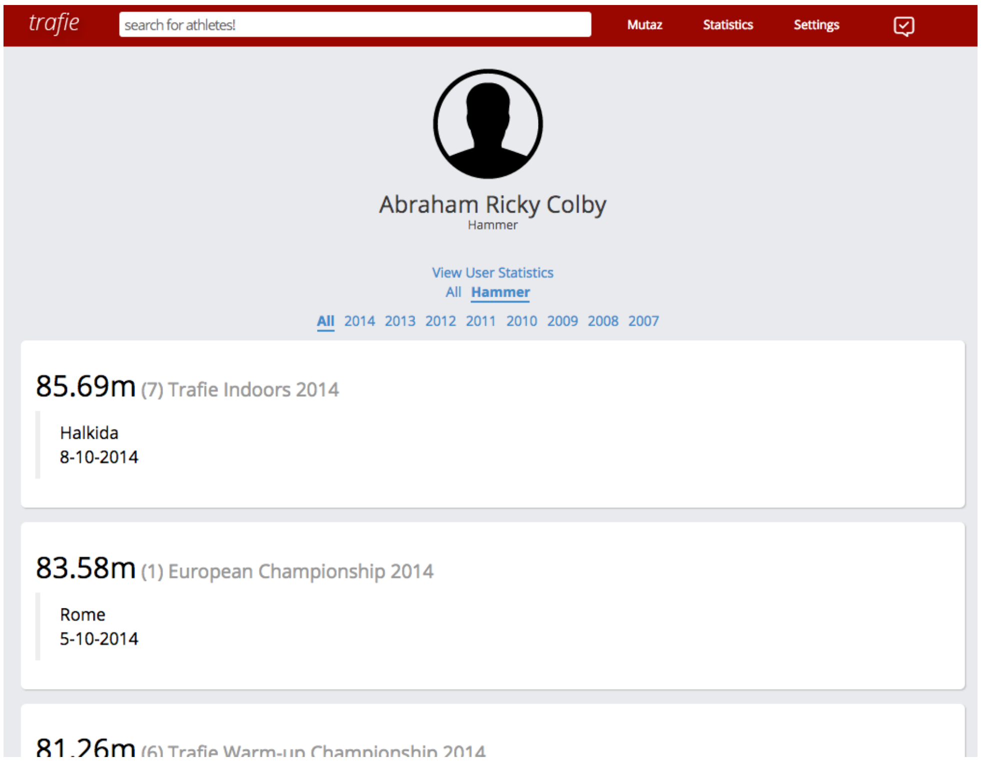
image 14.6 : Confirmation pop-up for deletion of an activity on desktop version



Mutaz Bashim
High Jump-Greece

Enter your performance:
m: 2 cm: 31
Date: 09-February-2015
Place: 3
Location: Karlsruhe
Competition: IAAF Indoor Meet
Notes: IAAF World Indoor Meetings IAAF Indoor Permit
Private: ☐

image 14.7 : Form for editing an existing activity on desktop version



Abraham Ricky Colby
Hammer

View User Statistics
All Hammer
All 2014 2013 2012 2011 2010 2009 2008 2007

85.69m (7) Trafie Indoors 2014
Halkida
8-10-2014

83.58m (1) European Championship 2014
Rome
5-10-2014

81.26m (6) Trafie Warm-up Championship 2014

image 14.8 : The profile of an other user in our system on desktop version. 'View User Statistics' option is visible in this case.

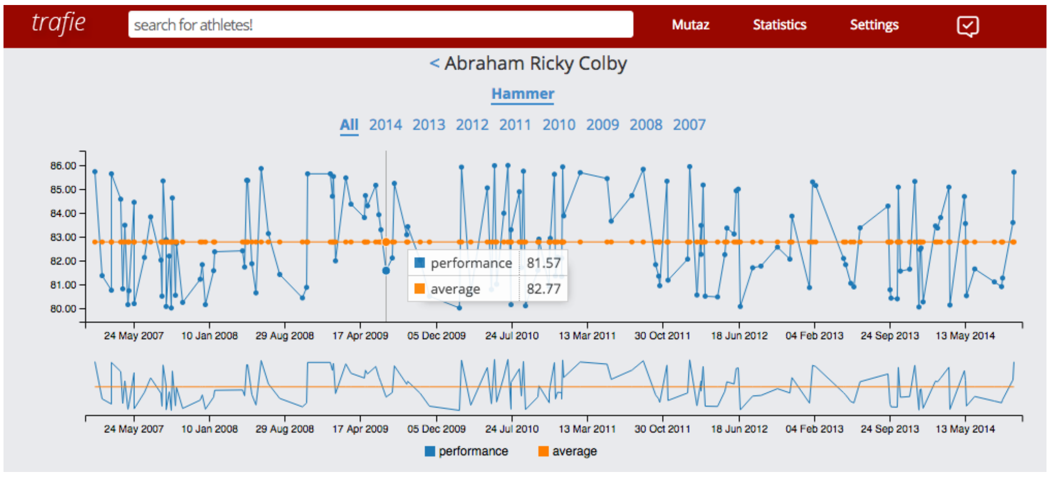


image 14.9 : Statistics page on desktop version, with 'all' years selected as an option for time filtering. Also a tooltip is visible while the mouse is hover one activity.

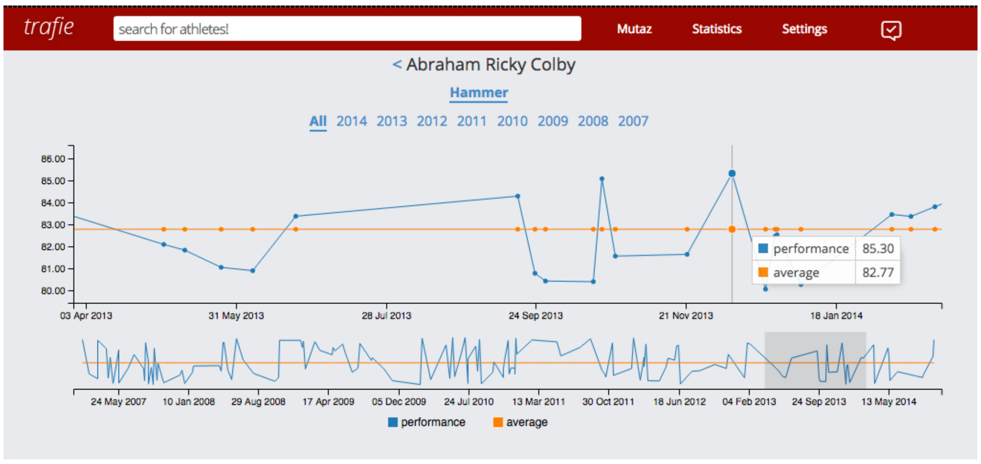
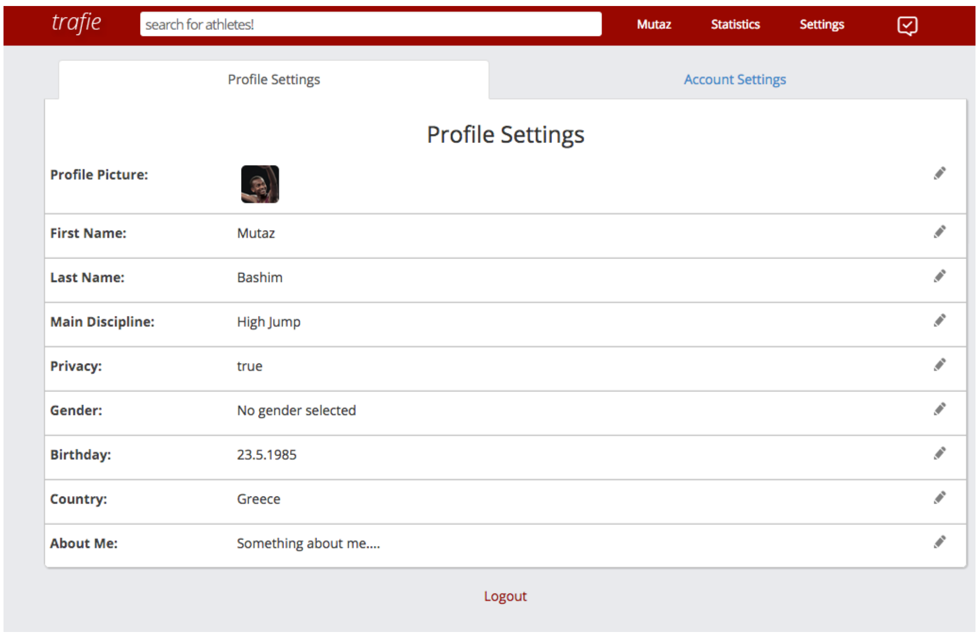


image 14.10 : Statistics page on desktop version, with 'all' years selected as an option for time filtering. User has select a specific time-window in order to zoom in and see activities in bigger detail.

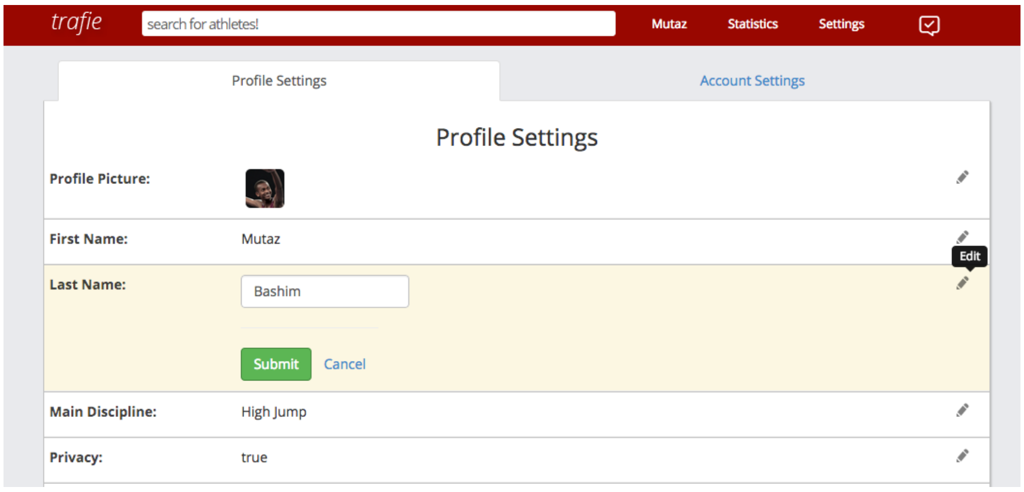


Profile Settings

Profile Picture:	
First Name:	Mutaz
Last Name:	Bashim
Main Discipline:	High Jump
Privacy:	true
Gender:	No gender selected
Birthday:	23.5.1985
Country:	Greece
About Me:	Something about me....

Logout

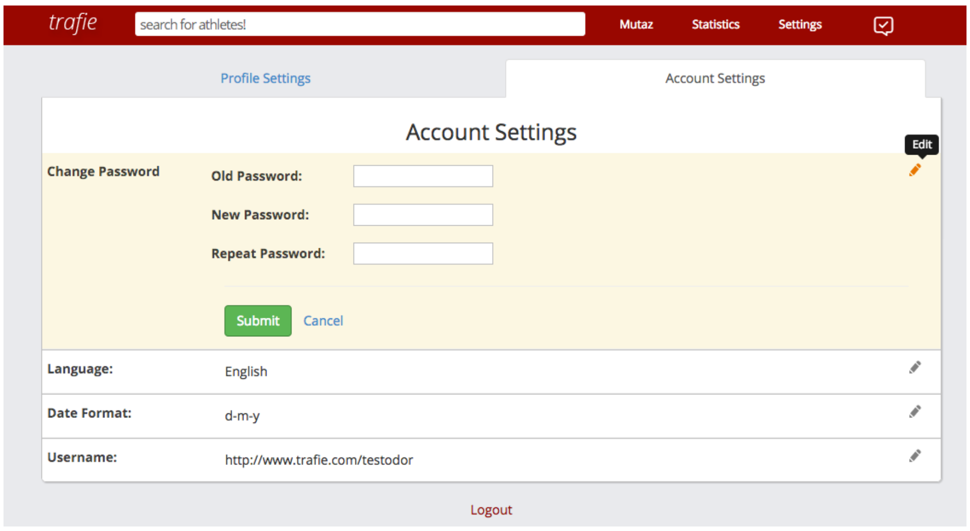
image 14.11 : Settings page (profile tab) for desktop version.



Profile Settings

Profile Picture:	
First Name:	Mutaz
Last Name:	Bashim
Main Discipline:	High Jump
Privacy:	true

image 14.12 : Settings page for desktop page (profile tab) with 'last name' option in editable state.



Account Settings

Change Password	Old Password: / New Password: / Repeat Password:
Language:	English
Date Format:	d-m-y
Username:	http://www.trafie.com/testodor

image 14.13 : Settings page for desktop page (account tab) with 'change password' option in editable state

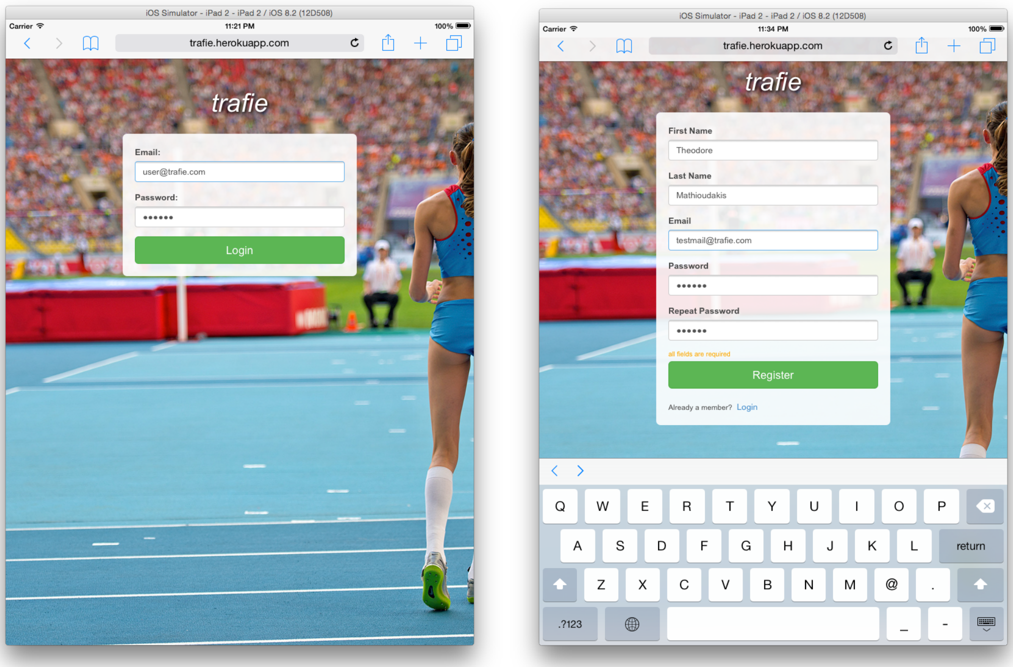


image 14.14 : Login and Registration pages on tablet version

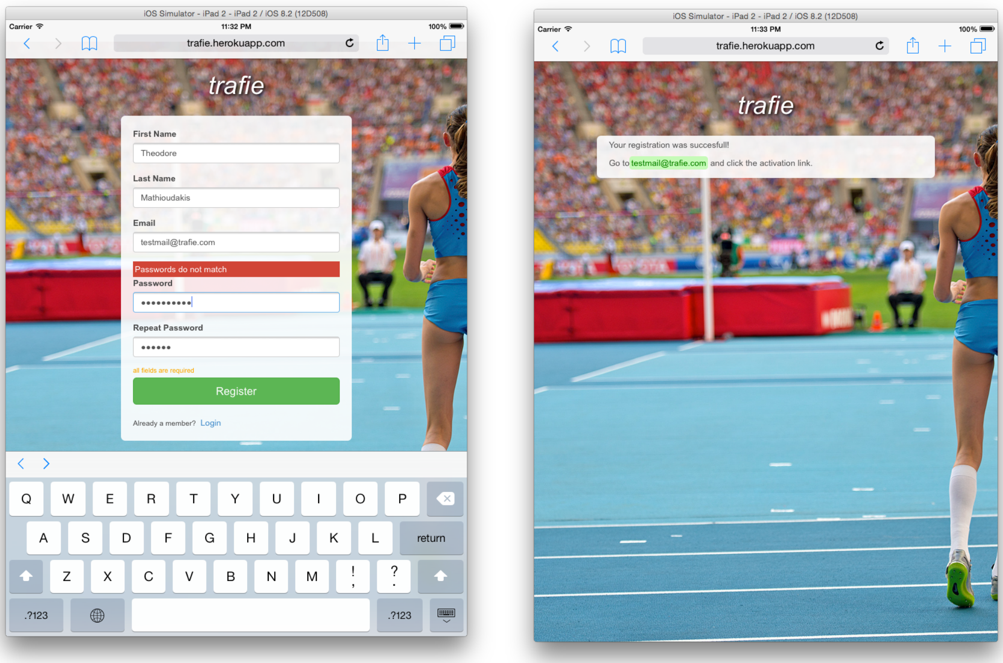
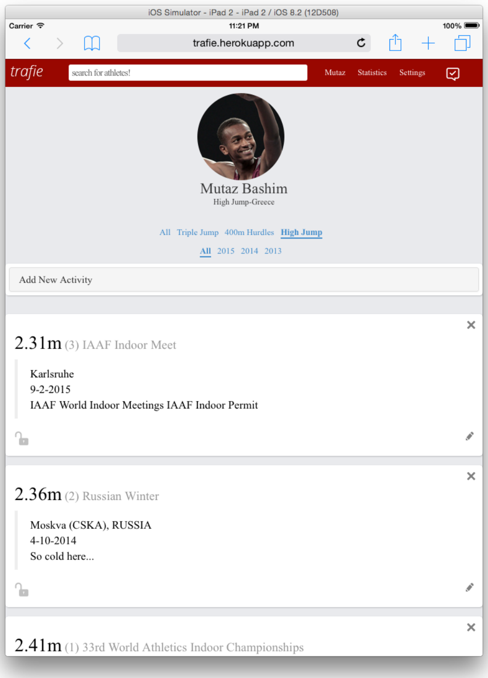


image 14.15 : Registration page on the left. With an error message visible. And a prompt message for user, after the successful registration, to check his email and confirm his registration.


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
Carrier 11:21 PM 100%
trafie.herokuapp.com

trafie | search for athletes! | Mutaz | Statistics | Settings

Mutaz Bashim
High Jump-Greece

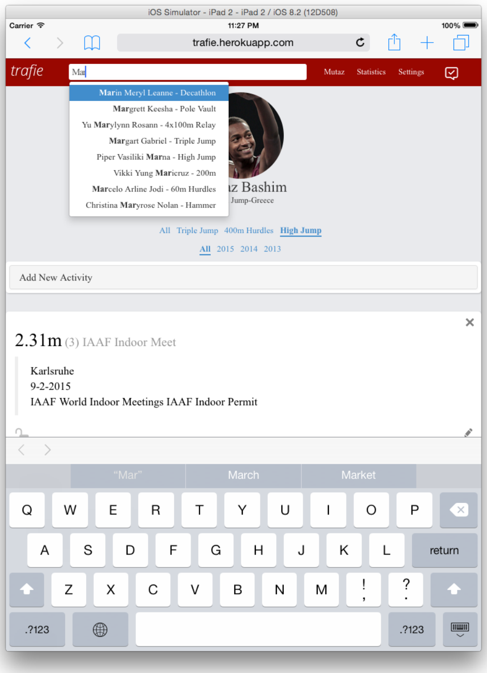
All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

Add New Activity

2.31m (3) IAAF Indoor Meet
Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

2.36m (2) Russian Winter
Moskva (CSKA), RUSSIA
4-10-2014
So cold here...

2.41m (1) 33rd World Athletics Indoor Championships


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
Carrier 11:27 PM 100%
trafie.herokuapp.com

trafie | Mar | Mutaz | Statistics | Settings

Marin Meryl Leanne - Decathlon
Margrett Keesha - Pole Vault
Yu **Mar**ylynn Rosann - 4x100m Relay
Margart Gabriel - Triple Jump
Piper Vasiliki **Mar**na - High Jump
Vikki Yung **Mar**icruz - 200m
Marcelo Arline Jodi - 60m Hurdles
Christina **Mar**yrose Nolan - Hammer

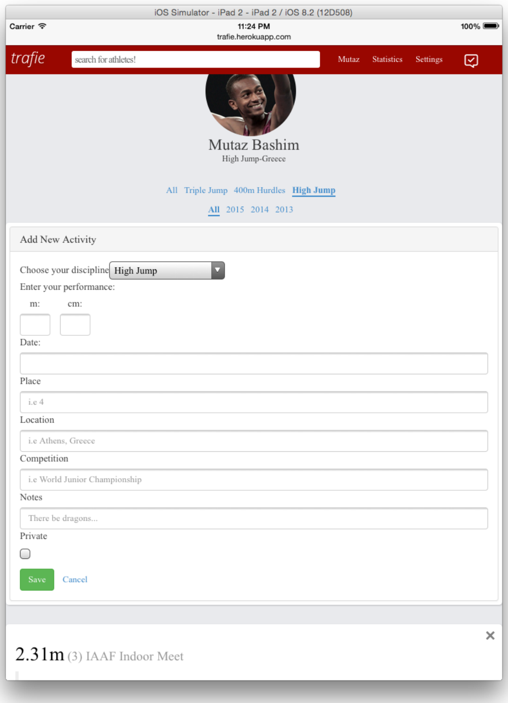
z Bashim
Jump-Greece

All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

Add New Activity

2.31m (3) IAAF Indoor Meet
Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

image 14.16 : Profile page (left) and search users through search field using autocomplete feature (right) on tablet version


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
Carrier 11:24 PM 100%
trafie.herokuapp.com

trafie | search for athletes! | Mutaz | Statistics | Settings

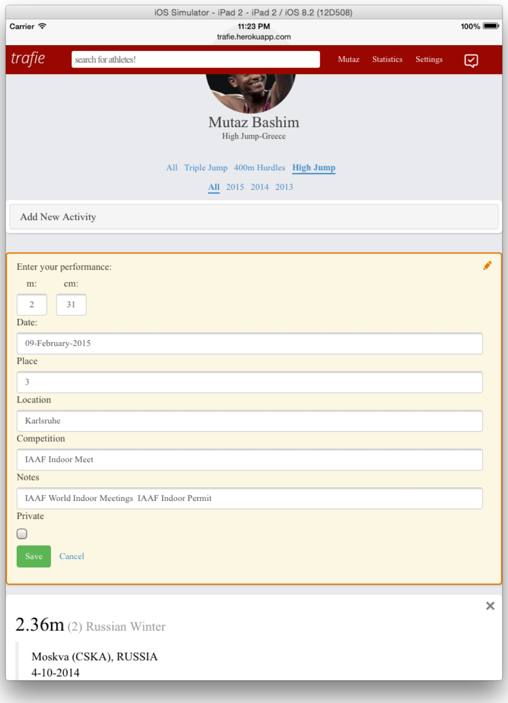
Mutaz Bashim
High Jump-Greece

All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

Add New Activity

Choose your discipline: High Jump
Enter your performance:
m: cm:
Date:
Place
i.e 4
Location
i.e Athens, Greece
Competition
i.e World Junior Championship
Notes
There be dragons...
Private
Save Cancel

2.31m (3) IAAF Indoor Meet


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
Carrier 11:23 PM 100%
trafie.herokuapp.com

trafie | search for athletes! | Mutaz | Statistics | Settings

Mutaz Bashim
High Jump-Greece

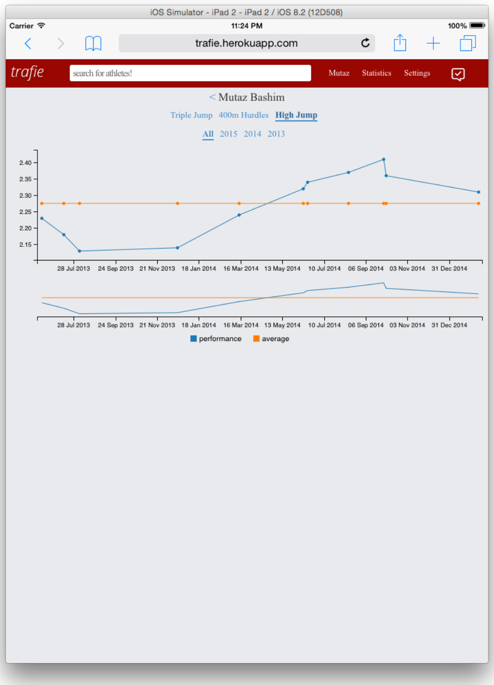
All Triple Jump 400m Hurdles **High Jump**
All 2015 2014 2013

Add New Activity

Enter your performance:
m: cm:
2 31
Date:
09-February-2015
Place
3
Location
Karlsruhe
Competition
IAAF Indoor Meet
Notes
IAAF World Indoor Meetings IAAF Indoor Permit
Private
Save Cancel

2.36m (2) Russian Winter
Moskva (CSKA), RUSSIA
4-10-2014

image 14.17 : Add new Activity Form (left) and Edit Existing Activity Form (right) on tablet version.


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
trafie.herokuapp.com
trafie
search for athletes!
Mutaz Statistics Settings
< Mutaz Bashim
Triple Jump 400m Hurdles High Jump
All 2015 2014 2013
performance
average

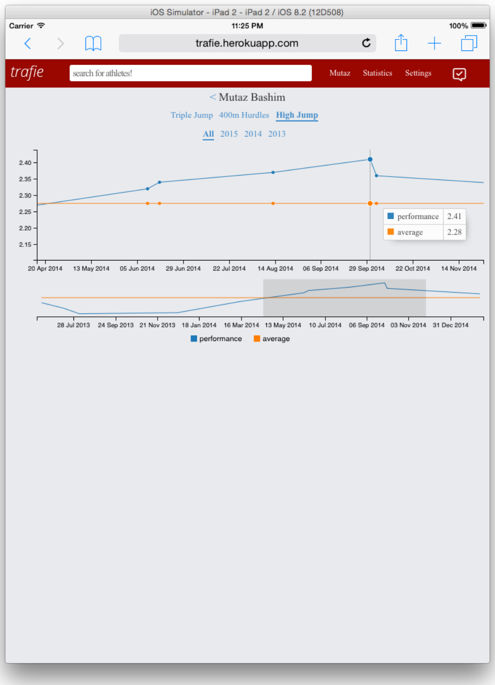
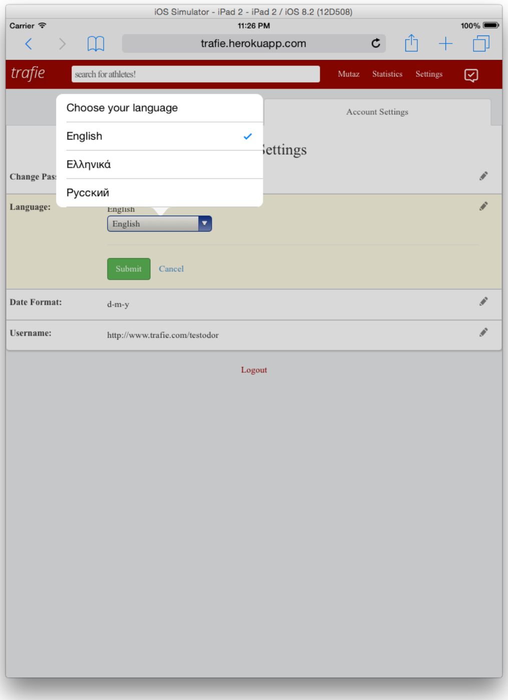


image 14.18 : Statistics page without filtering (left) and with filtering and time-window selected (right).


iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)
trafie.herokuapp.com
trafie
search for athletes!
Mutaz Statistics Settings
Account Settings

Choose your language

English ✓

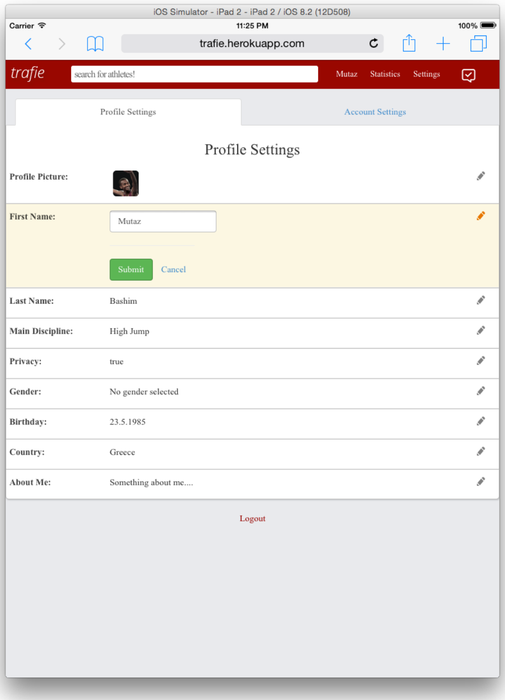
Ελληνικά

Русский

Change Pas	
Language:	English
Date Format:	d-m-y
Username:	http://www.trafie.com/testodor

Submit Cancel

Logout



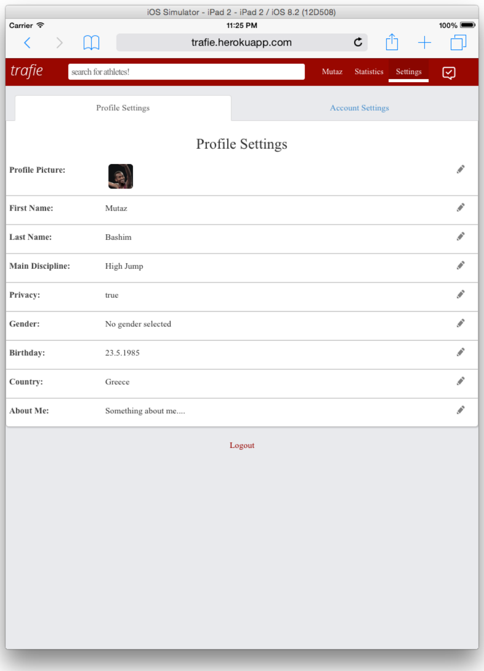
Profile Settings

Profile Picture:	
First Name:	Mutaz
Last Name:	Bashim
Main Discipline:	High Jump
Privacy:	true
Gender:	No gender selected
Birthday:	23.5.1985
Country:	Greece
About Me:	Something about me....

Submit Cancel

Logout

image 14.19 : Settings page with option in editable state on tablet version.



iOS Simulator - iPad 2 - iPad 2 / iOS 8.2 (12D508)

Carrier 11:25 PM 100%

trafie.herokuapp.com

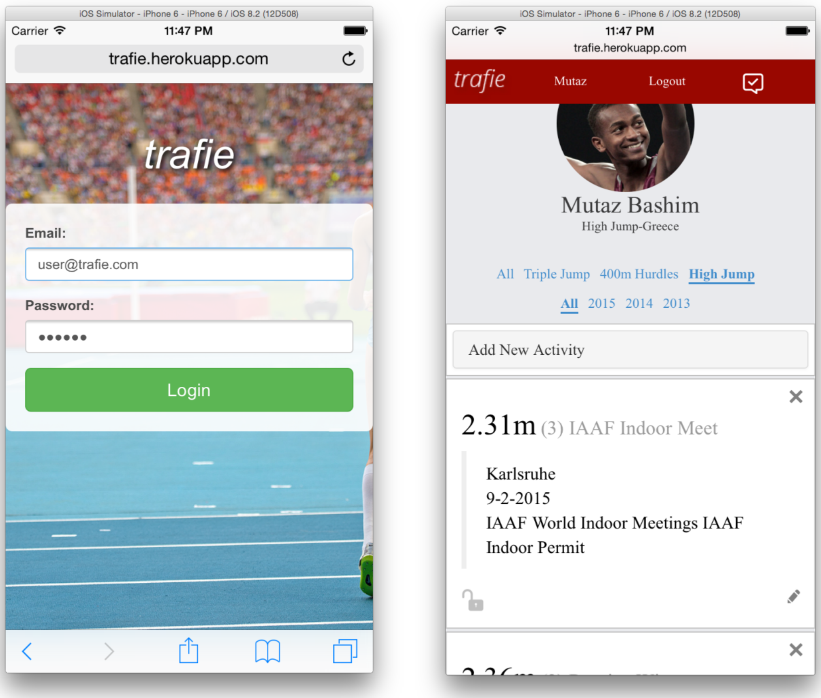
Profile Settings | Account Settings

Profile Settings

Profile Picture:	
First Name:	Mutaz
Last Name:	Bashim
Main Discipline:	High Jump
Privacy:	true
Gender:	No gender selected
Birthday:	23.5.1985
Country:	Greece
About Me:	Something about me....

Logout

image 14.20 : Settings page(profile tab) on tablet version.



iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.2 (12D508)

Carrier 11:47 PM

trafie.herokuapp.com

Email: user@trafie.com

Password: ••••••

iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.2 (12D508)

Carrier 11:47 PM

trafie.herokuapp.com

Mutaz Bashim

High Jump-Greece

All Triple Jump 400m Hurdles High Jump

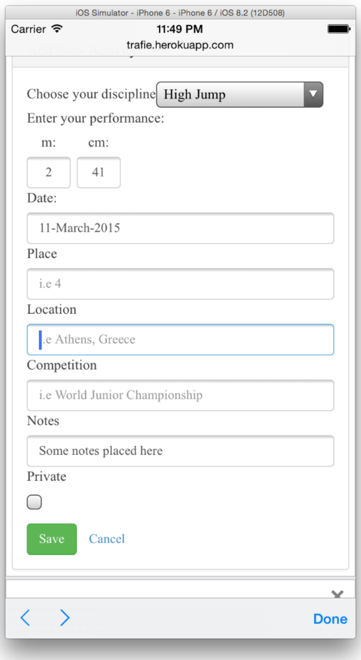
All 2015 2014 2013

Add New Activity

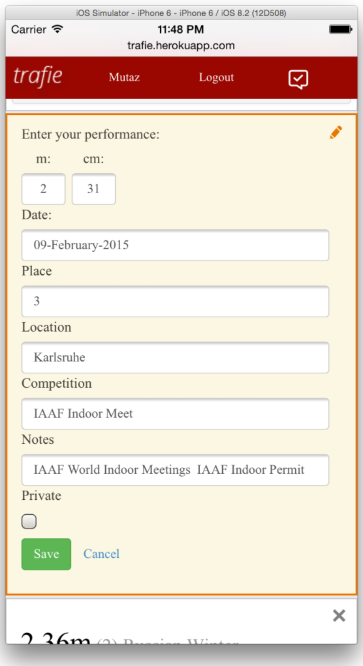
2.31m (3) IAAF Indoor Meet

Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

image 14.21 : Login (left) and Profile (right) pages, on mobile version.

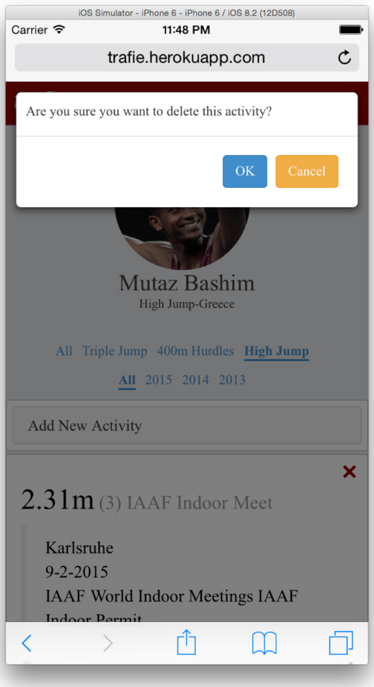

iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.2 (12D508)
Carrier 11:49 PM
trafie.herokuapp.com

Choose your discipline High Jump
Enter your performance:
m: cm:
2 41
Date:
11-March-2015
Place
i.e 4
Location
i.e Athens, Greece
Competition
i.e World Junior Championship
Notes
Some notes placed here
Private
Save Cancel


iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.2 (12D508)
Carrier 11:48 PM
trafie.herokuapp.com

Enter your performance:
m: cm:
2 31
Date:
09-February-2015
Place
3
Location
Karlsruhe
Competition
IAAF Indoor Meet
Notes
IAAF World Indoor Meetings IAAF Indoor Permit
Private
Save Cancel

image 14.22 : Add New (left) and Edit Existing (right) Activity on mobile version.


iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.2 (12D508)
Carrier 11:48 PM
trafie.herokuapp.com

Are you sure you want to delete this activity?

OK Cancel

Mutaz Bashim
High Jump-Greece

All Triple Jump 400m Hurdles High Jump
All 2015 2014 2013

Add New Activity

2.31m (3) IAAF Indoor Meet

Karlsruhe
9-2-2015
IAAF World Indoor Meetings IAAF Indoor Permit

image 14.23 : Confirmation pop-up for deletion of an activity on mobile version

CONCLUSIONS AND FUTURE WORK

In this thesis we have build an infrastructure with a clean REST API over a noSQL database system and a web application which consumes this API, offering to our end users a simple way to keep track and analyse their data through a clean and intuitive User Interface, using every device they have, with internet access.

After running the Scrum Agile Software Development methodology for building an end-to-end web application, and comparing with our experience in other software development mythologies like “Waterfall” and “RUP”, we saw in practice how efficient this methodology is and how flexible we can become, when big or small changes are needed in our implementation. We needed to make major architecture changes in many different levels of the software stack, in order to improve the scalability and maintainability of our software, and also easily change and prioritise features per case, even if we need to change our backlog in every Sprint.

We have many great ideas, that we want to implement in the future, starting by building native applications for iOS and Android devices taking advantage of their hardware for tracking human activity and automatically generate data for our application. Using the geolocation sensors these devices contains, will help us socialise our application and creating a social network of athlete through the world. Of course we can implement more mainstream features, connecting existing social networks with our application and bring existing athletes’ communities in our application.

We can enhance our REST API and build applications for smart-watches, that will consume it, and will take advantage of their unique property to following our end-users everywhere, tracking his health and other important body metrics. This will help them to understand and analyse in more detail their overall progress.

Finally providing an open SDK(software development kit) to everyone will allow other developers, who also love “track and field”, to build their own applications for different devices and offer their ideas and services to athletes of our network.

BIBLIOGRAPHY

1. “Distributed Scrum: Agile Project Management with Outsourced Development Teams” - J. Sutherland, A. Viktorov, J. Blount, & N. Puntikov, *IEEE HICSS 40th Hawaii International Conference on Software Systems*, 2007.

2. “The Future of Scrum: Parallel Pipelining of Sprints in Complex Projects” - J. Sutherland, in *Agile 2005*, Denver, CO: IEEE, 2005.

3. “The First Scrum” - Jeff Sutherland, Ph.D. PatientKeeper, Inc., 2004

4. “Scrum- A Pattern Language for Software Development” - M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland, vol. 4, N. Harrison, Ed. Boston: Addison-Wesley, 1999, pp. 637-651.

5. http://agileatlas.org

6. http://scrumguides.org/scrum-guide.html

7. http://scrumfoundation.com/library

8. https://www.scrumalliance.org

9. https://www.samrantech.com/methodologies-comparison-between-agile-and-rup-2.html

10. http://www.agilemodeling.com/essays/agileModelingRUP.htm

11. http://docs.mongodb.org/

12. https://nodejs.org/documentation/

13. https://docs.angularjs.org/guide