



University of Piraeus
Department of Digital Systems
Postgraduate Program "Information Systems & Services"

**Ηλεκτρονική Δήλωση
Ατυχήματος Και Ανίχνευση
Βλάβης Σε Υπηρεσιοστρεφή
Περιβάλλοντα**

**Service-oriented Environments for:
'Here We Are'**

**Vehicle Accident Statement & Car
Damage Detection**

ME1905

Nikitas Michail Kastis

Supervising Professor: Dimosthenis Kiriazis

PIRAEUS 2021

Acknowledgements

It has been two special years for me since I decided to change career path and join the current MSc program at University of Piraeus, and there are many people that I would like to mention and thank here.

First, I would like to thank my supervisor, Dr. Dimosthenis Kiriazis, for his unconditional support, for his patience and constant feedback he provided throughout my MSc studies.

I would also like to acknowledge my colleagues in the MSc course, especially Dimitris Kakomitas. Collaborating, discussing, and even arguing with Dimitris not only helped me get a deeper understanding of Computer Science, but also feeling confident for my career change.

Additionally, I would like to thank my two friends, Afroditi Tzifa and Ioannis Mourelatos for being so understanding and supportive when things got tough.

Finally, I would like to thank my family because they love me so much! I love them too! Especially, my twin brother, Dr. Kastis Eleftherios, who was always there, any time of the day, to support me!

Kastis Nikitas

Abstract

This paper aims to present the creation of an application using Microservices technology and Machine Learning. 'HereWeAre' is a web application for reporting vehicle accidents and consists of three microservices.

Each microservice is built in Python language using FastApi framework and relational Postgres database. The first service is an Authentication System while the second one concerns the vehicle accident statement. The third service uses image analysis models, created with TensorFlow, to detect vehicle damage in Python. User interface is build in React.

Next steps would include a React-Native user interface for mobile devices, more services and the development of better models as more data will be collected.

To run this application, you need: docker desktop, minikube, makefile and Linux system or a Linux subsystem in windows. To get access to the source code, please contact me at email: nitaskastis15@gmail.com.

Πίνακας περιεχομένων

Acknowledgements.....	2
Abstract.....	3
1.INTRODUCTION	6
1.1 APPLICATIONS LIKE "HERE WE ARE"	7
1.1.1 Web Forms.....	7
1.1.2 Mobile App Assisto: Report Your Car Accident	7
1.1.3 Car Damage Recognition by Altoros	7
1.1.4 Claim Genius.....	8
1.2 WHAT DOES "HERE WE ARE" APP OFFER MORE?	8
2. SERVICE-ORIENTED ARCHITECTURE.....	9
2.1 CONTAINERS	9
2.1.1 Container History	9
2.2 DOCKER.....	10
2.2.1 Docker History	10
2.2.2 Docker Desktop	10
2.2.3 Docker Vocabulary	11
2.3 MICROSERVICES.....	11
2.3.1 Microservices History and Future	12
2.3.2 Benefits and Challenges of Microservices.....	12
2.4 ORCHESTRATION OF MICROSERVICES.....	13
2.5 KUBERNETES	13
2.5.1 Kubernetes History	14
2.5.2 Kubernetes Vocabulary	14
2.5.2 What Does Kubernetes?	15
3. OTHER TECHNOLOGIES IN 'HERE WE ARE' APP	16
3.1 BACK-END APPLICATION.....	16
3.1.1 Python.....	16
3.1.1.1 Pros and Cons of Python.....	17
3.1.1.2 Python History.....	17
3.1.2 FastAPI Framework	19
3.1.2.1 Pros and Cons of FastAPI framework	19
3.1.2.2 FastAPI History	20
3.1.3 Tensorflow Library	20

3.1.3.1 TensorFlow History	20
3.1.4 SQLAlchemy.....	20
3.2 DATABASE.....	21
3.2.1 PostgreSQL	21
3.3 FRONT-END	21
3.3.1 React	21
3.3.1.1 React History	22
3.3.1.2 React Now	22
3.3.2 Redux.....	23
4. MACHINE LEARNING	24
4.1 SUPERVISED LEARNING	24
4.2 UNSUPERVISED LEARNING	25
4.3 OVERFITTING.....	26
4.4 CONVOLUTIONAL NEURAL NETWORKS	26
4.5 CREATING MODELS IN TENSORFLOW.KERAS	27
4.6 THE FLOW TO GIVE A PREDICTION	29
4.7 CREATED MODELS – IT DOES NOT ALWAYS WORK	31
4.7.1 First model – Sequential.....	31
4.7.2 Second model – InceptionV3.....	40
5. CREATING THE APPLICATION	47
5.1 CREATE APPLICATION FLOW	47
5.2 DEVELOPING EACH CONTAINER	49
5.3 AUTHENTICATION MICROSERVICES.....	51
5.4 ACCIDENT STATEMENT MICROSERVICE	55
5.5 VEHICLE DAMAGE DETECTION MICROSERVICE.....	60
5.6 FRONT END: REACT @ REDUX	62
5.6.1 How Redux works?.....	62
5.7 KUBERNETES	64
6. CONCLUSIONS.....	72
6.1 PROBLEMS CREATING THE APP	72
6.2 FUTURE WORKS ON 'HERE WE ARE'	72
7. REFERENCES	74
8. USER INTERFACE	79

1.INTRODUCTION

In the last 30 years, the rapid progress in information technology has led people and organizations to demand newer, faster, more reliable, and more secure applications. Especially, during the Covid-19 pandemic, as described in the Organization for Economic Co-operation and Development Outlook 2020 (OECD2020), usage of internet services increased by 60% in the first semester of 2020 in some countries. More bureaucratic activities become digitized, students attend school classes remotely, while co-workers learn to co-operate via online rooms, achieving cost reduction for the companies. According to OECD2020, the governments of these countries stressed the national need for digital transformation, after decades of a slow-paced approach. Irreversible changes prepare us for future opportunities where most services and human interactions may depend on digital technologies more than ever before. ^[1]

The year 2021 has also been exceptional. Apart from the massive vaccine research and production, Artificial Intelligence (AI) meets explosive growth, shifting the relevant community from advanced language translate models to NoCode platforms that achieve building without much coding knowledge.

Following the current trend, HereWeAre web application is created to digitize vehicle accident statements, providing both drivers and insurance companies access to drivers' statements, photos from the accident, a car damage detection AI system, etc.

Googling "online accident statement", the first results show instructions on how to complete the paper forms and/or give examples of completed forms over the last decades. Most insurance companies offer e-mail services for vehicle accident statements. Our objective is to create a new application that is easy to use, safe, fast, and also includes sketch and photos in the accident statement.

1.1 APPLICATIONS LIKE “HERE WE ARE”

1.1.1 Web Forms

There are insurance company sites like Asfalish.gr that provide the user with the ability to complete a form about a vehicle accident, to add/remove other drivers, and to send the form, without editing, saving, history, files or sketch options.

1.1.2 Mobile App Assisto: Report Your Car Accident

User can find information in the page www.assis.to where they can add any accident between one or two vehicles. This app has a very nice user interface (UI) with animations, since the user can keep their personal data, their vehicle accident statement history and add photos in a vehicle accident or even a sketch. Moreover, this app uses gps services.

It is available in 42 countries, and collaborates with many insurance companies, and other stakeholders in repairing vehicle damage field, like Carglass-Belgium. It has more than 50,000 subscribers, who have rewarded with a 3.3/5 grade (based on 558 users).

A major weakness of the app is that the user cannot make a statement for an accident with more than 2 vehicles.

1.1.3 Car Damage Recognition by Altoros

Altoros is an IT consultant company that provides services to more than 2000 organizations. Among the services included in their site, there is Car Damage Recognition. This system uses ML algorithms with an API that utilizes computer vision, as it is described in their webpage. The model follows these steps: ^[2]

- ✓ Identifying the car:
 - ✓ Car parts localization/segmentation
 - ✓ Check in photo quality
 - ✓ Photo stream processing
- ✓ Understanding the damage:
 - ✓ Car part damage level estimation
 - ✓ AI-generated estimated cost
 - ✓ Repair or Replace decision
 - ✓ Overall car damage estimation

Commercially, it is addressed to the insurance company, and not for public usage.

1.1.4 Claim Genius

Claim genius is a mobile app that uses AI technology over image analysis to predict the repair estimation, suggesting the repair or the replacement of the damaged part of the vehicle. Commercially, it is addressed to the carriers, and again not to the public.

1.2 WHAT DOES “HERE WE ARE” APP OFFER MORE?

This new web application combines the services of all the above apps, having a vehicle accident statement system and a vehicle damage detection system; it is free-source for users and aims to develop partnerships with all the insurance companies and other industries relative to vehicles.

User can declare a car accident for more than 2 cars, draw a sketch over the accident, add photos, add the detected damage to the accident statement, and keep track of their vehicle accident statements. If the other drivers' companies are partners with the app, this enables drivers to respond to the car accident, so all insurance companies can have all data in the application. Additionally, using filtering that is added in the accident section, they can find an accident by address, city, date or last name of the driver who made the statement.

2. SERVICE-ORIENTED ARCHITECTURE

The architectural style of service-oriented architecture encourages the use of services with loose coupling. In the field of software design, application components give services to other components through a network using a communication protocol.

Microservices are a modern interpretation of service-oriented architectures used in building distributed software systems.

2.1 CONTAINERS

A container is an executable unit of software, that includes the code along with all its dependencies, enabling the app to run fast and be reliable in every computable environment, from a desktop to the cloud. [3]

The advantages of using containers in apps are:

- ✓ Containerized apps can be deployed, scaled or patched faster
- ✓ DevOps know that their containerized apps will run smoothly, regardless the platform where these apps are deployed.
- ✓ Container technology needs fewer physical resources than VM environments as it does not have an operation system in it.
- ✓ Agile business is better supported with container technology. [4]

2.1.1 Container History

Container technology came to life in 1979, when chroot was in development. Chroot command isolated processes into their own separated filesystems, facilitating testing without affecting global system environment. [5]

In 2000, Chroot was followed by jail command including additional features that provided users the ability to configure software installations, assign IP addresses, and modify each jail, although included apps did not have much of functionality. In 2004, Solaris containers appeared, which used Solaris Zones to create full app environments.

In 2006, engineers in Google designed process containers that isolated and limited the sources that a process used. In 2008, these containers were merged to the Linux Kernel 2.6.24, leading to the Linux Containers (LXC), as they are known today. LXC provide virtualization at OS by permitting multiple segregated Linux environments to run on a shared Linux kernel. [6]

In 2013, Google open-sourced container stack with the LMCTFY project, that ended two years later, as Google offered its core concepts to the project libcontainer, which was added to Docker 0.9. [7]

2.2 DOCKER

Docker is a container technology. It's a platform system using OS-level virtualization for creating and managing containers. It enables user to isolate the apps they are working on from infrastructure and to eliminate time between coding and executing it in production.

2.2.1 Docker History

In 2008, 3 friends, Solomon Hykes, Sebastien Pahl and Kamel Founadi, created the company DotCloud. Their goal was to create technologies based on containers that everyone could use. 5 years later, DotCloud changed to Docker. ^[8]

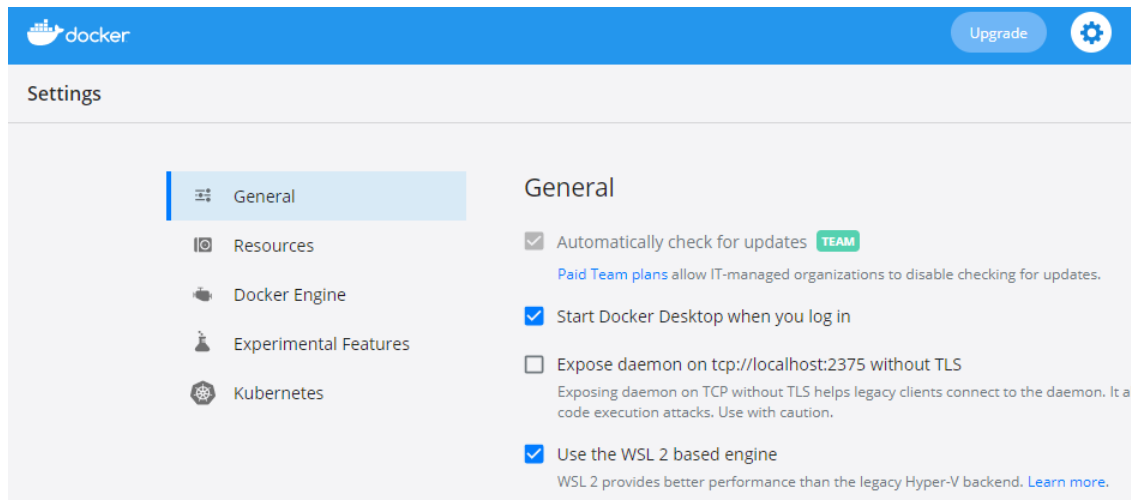
Docker was released in March 2013, using LXC as its default environment. Later, libcontainer replaced LXC. Except for the above-mentioned Google contribution, other main contributors are: Docker team, Huawei, IBM, Red Hat, Cisco and Microsoft. ^[9]

In November 2019, cloud-computing Mirantis acquired Docker enterprise, including Docker Engine, with the vision to accelerate Kubernetes-as-a-Service.

2.2.2 Docker Desktop

Docker desktop is an open-source, downstream product that contains Docker Engine. Instead of creating a full Linux Virtual Machine, Docker Desktop is as optimized and light-weighted, achieving a much better performance.

Docker desktop supports Windows Sub-system for Linux 2 based engine, which brings an important change in its architecture; it permits the user to leverage Windows and Linux containers in one machine, without emulator. In addition, its container tasks run faster and Docker desktop utilizes only the required recourses. ^[10]



1. Docker - Usage of WSL 2 based engine

2.2.3 Docker Vocabulary

Image	A package including all dependencies and info needed to have a container created and running. Once an image is created, it's immutable, until it's removed.
Container	Running instance of docker image.
Volume	Preferred tool for data persistence. A volume can last more than a container, keeping data every time a container stops or restarts.
Dockerfile	A file that includes building instructions for a docker image. It starts stating the base image in the top line. Then it guides the user about requirement installments and demonstrates how to copy the files to the working environment.
Build	An action that builds a container image following Dockerfile information. ^[11]
Compose	A tool used to define and to run multi-container apps (Docker-compose.yml).

2.3 MICROSERVICES

Developing an app in microservices architecture is based in the logic of creating small and separated services that can work together via network calls. As separated and autonomous, a microservice needs to have the ability to change without affecting the rest microservices of an app. ^[12]

2.3.1 Microservices History and Future

In 2005, Dr. Peter Rodgers utilized the term "Micro Web Services" on a cloud computing presentation, promoting the idea of well-organized software components-services that apply the architectural fundamentals of REST and Web services along with Unix-like pipeline scheduling. Emphasizing in the simplicity and flexibility of this service-oriented architecture, he originated a functional model that led to microservices. [13]

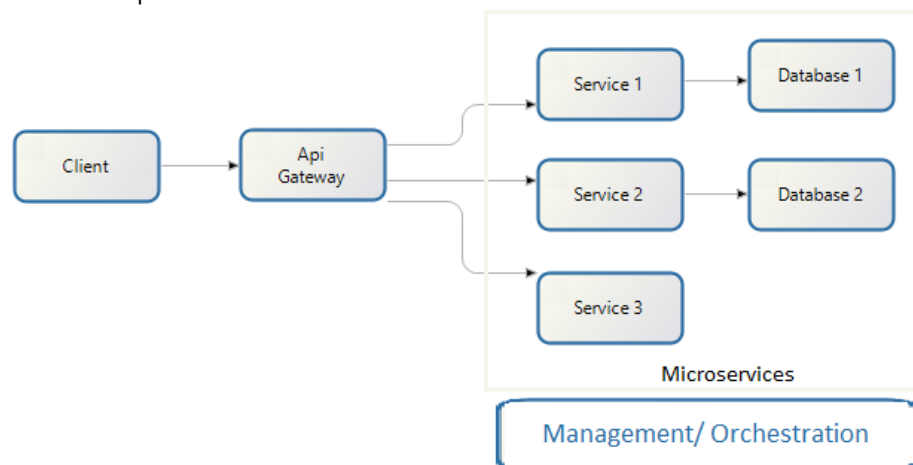
According to Juval Löwy in 2007, each class is a service in building systems, instructing Windows Communication Foundation to acquire the technology to support this particular use of services. [14]

In March 2012, James Lewis presents a case study "Micro Services – Java, the Unix Way" at 33rd Degree in Kraków. Former director for the Cloud Systems of Netflix, Adrian Cockcroft characterized this approach as "fine-grained SOA". [15]

In 2018, the value of Global Cloud Microservices Market was \$649.04 million, while projections set the value at \$3,054.23 Million, with Compound Annual Growth Rate at 21.37% during this period. [16]

2.3.2 Benefits and Challenges of Microservices

An example of the microservices architecture is described in image 2:



2. Microservices architecture

Client may send its calls to an Api Gateway, which will further send these calls to the appropriate services. Each service may use its own database. Orchestration will be described in the next chapter.

Advantages:

- ✓ Agility: The independence of each microservice makes updates much easier. Either bug fixing or new add-ons in a service does not need the redeployment of the entire app, only of the specific service

- ✓ Scalability: Using an orchestrator, each microservice can request more resources, without creating further issues to the app
- ✓ Preventing dependency problems: Adding new features in microservices is easier, because developers will not need to check or rewrite code in the app, due to the autonomy of each service
- ✓ Usage of different technologies: Each service can be written in the language that fits best and make use of the appropriate database, relational or no-relational. This also helps to data isolation
- ✓ Fall down isolation: When a microservice confronts a problem and becomes unavailable, the entire app goes on
- ✓ Makes developer happier: Creating smaller teams that work on smaller services than a whole app is a key for better communication and higher productivity.

Challenges:

- ✓ Handling the architecture may be complex
- ✓ Too many technologies are difficult to handle
- ✓ Bad architecture that depends on the communication between the services can cause latency
- ✓ Updating services that are vital to other services in the app should not break them. ^[17]

2.4 ORCHESTRATION OF MICROSERVICES

To understand "orchestration", let us think an orchestra of musicians. Each musician has a particular role, and the whole group needs timing, coordination, arrangements, etc. to generate a beautiful result. That's what microservices need too, and orchestration tools are the solution to achieve that.

Following the sharp rise of container adoption, container orchestration technology is expanding rapidly and massively. Orchestration services allow developer teams to control, schedule, track and operationalize containers at scale in a more efficient way.

Benefits of using orchestration to app architecture:

- ✓ Enables communication between containers
- ✓ Shows developer the right time to run new containers
- ✓ Ensures high availability across your infrastructure. ^[18]

2.5 KUBERNETES

Kubernetes or K8s, is described in "Kubernetes.io" as "an open-source system that is used to automate deployment, scaling, and management of containerized apps". ^[19] William Henry, cloud strategist in Red Hat, writes: "In

other words, you can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters. The power of the open-source cloud-native comes only in part from individual projects such as Kubernetes. It derives, perhaps even more, from the breadth of complementary projects that come together to create a true cloud-native platform" [20]

2.5.1 Kubernetes History

Kubernetes has its etymological root in the Greek word “κυβερνήτης”- “helmsman”. Brendan Burns, Craig McLuckie and Joe Beda founded Kubernetes and soon enough, other Google engineers joined. It was first announced in mid-2014 by Google and hit version 1.0 on July 21, 2015. The same day, Google donated the whole project to the Cloud Native Computing Foundation, run by the Linux Foundation. [21]

Over the last six years, Kubernetes is synonymous with the cloud-native development. Microsoft Azure and Amazon AWS initially offered different solutions to community but finally became supporters of K8s. The competition among cloud providers comes to who will run containerized apps with Kubernetes more efficiently. [22]

2.5.2 Kubernetes Vocabulary

Node	Nodes are ‘worker’ machines – physical or virtual – on clusters that include all that is needed to run app containers, plus container runtime and more services.
Cluster	Cluster includes a group of nodes that run containers.
Pods	The smallest object in the Kubernetes ecosystem that represents a bunch of one or more containers running together in a cluster.
Kubernetes API	As Kubernetes official site describes Kubernetes API as “the application that serves Kubernetes functionality through a RESTful interface and stores the state of the cluster”. It is the key of the system that follows developer orders and makes them reality.
Kubernetes Control Panel	As Kubernetes official site describes “Kubernetes Control Panel maintains a record of all of the Kubernetes Objects in the system and runs continuous control loops to manage those objects’ state.” Essentially, it checks if all behave properly. Its place is between Kubernetes and a cluster.
Master	In every cluster there is a master node and some “worker” nodes as well. Master node includes kube-apiserver, kube-controller-manager and kube-scheduler processes for state management of the cluster.

Kubectl	Command line interface for managing operations on each Kubernetes cluster. It follows a standard syntax for running commands: <code>kubectl [command] [TYPE] [NAME] [flags]</code>
Minikube	Minikube is a tool to run Kubernetes on a local machine that is used to simplify learning and developing for Kubernetes. ^[23]

2.5.2 What Does Kubernetes?

It is hard to maintain manually deployment of containers. There are problems like:

- ✓ a container crashes and goes down, then it needs to be replaced, otherwise the application container will be unreachable.
- ✓ a containerized microservice needs more or less instances in relation to traffic spikes or lower traffic hours.
- ✓ traffic is not distributed equally among multiple container images.

And here comes Kubernetes...

Using Kubernetes, developer can define in a Kubernetes configuration file the desired architecture, which containers to be deployed, the number of running instances, if it should be scaled up, if containers should be replaced, etc., and share that configuration file to any cloud provider and then create the resources and the deployment specified in the file.

Here is an example of configuration file:

```
apiVersion: v1
kind: Service
metadata:
  name: hereweare-deployment-service
  labels:
    app: hereweare
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: hereweare
```

That is the general idea of Kubernetes: a standardized way to describe the way to deploy containerized application on a multi machine set up. ^[24]

3. OTHER TECHNOLOGIES IN 'HERE WE ARE' APP

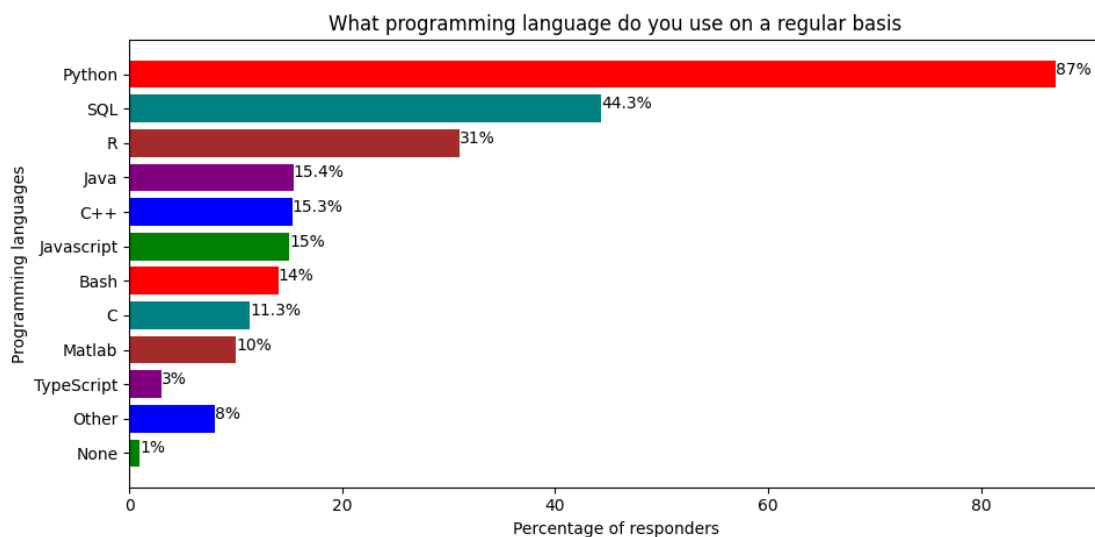
3.1 BACK-END APPLICATION

To create each microservice in the back-end we used Python language, in FastAPI framework, while TensorFlow library was used for the creation of the image-based prediction models.

3.1.1 Python

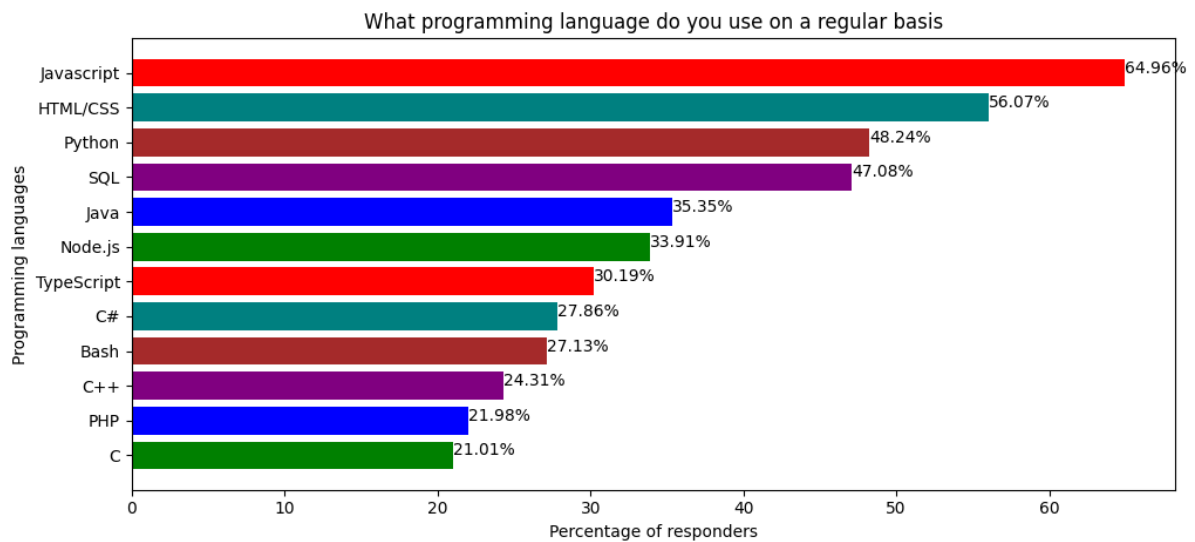
Python is an open-source programming language that is used in web development projects, data management, artificial intelligence, machine learning algorithms, video games, operating systems. It supports procedural, functional and object-oriented programming, among many programming paradigms. Google, Netflix, Facebook, Instagram, Amazon, Quora, slack, intel, Nasa, Dropbox, e-bay, Spotify, Uber are among the companies that use Python either for recommendations to the clients or for structural programs. [25]

Based on a Kaggle survey that took place in October 2019 over 19717 data scientists, Python is by far the most popular language. [25]



3. Kaggle survey 2019 in data scientists

In another survey, held by Stack Overflow, that took place among 83052 software developers worldwide from May 25, 2021 to June 15, 2021, Python came third after JavaScript and HTML/CSS. [26]



4. Kaggle survey 2021 in developers

3.1.1.1 Pros and Cons of Python

Pros:

- ✓ As an interpreted language, Python uses an interpreter that executes code line-by-line and not the entire code. This makes Python easier to debug.
- ✓ One apparent reason of popularity of Python is the number of available libraries. Machine learning, game development, web development for Python programmers does not start from scratch, since over 137,000 libraries provide already written functions for them.
- ✓ Permits asynchronous coding in web development.
- ✓ Easily readable with a simple syntax utilizing English words.
- ✓ Open-source language, updated frequently from a wide community with more than 10,000 developers. [27]

Cons:

- ✓ Slower than compiled languages, like C or C++.
- ✓ Python faces compatibility issues with mobile operating systems.
- ✓ Python consumes more memory, because it is flexible with data-types.
- ✓ Python has a quite underdeveloped database access layer
- ✓ Python programmers noted that Python gives errors that appear only at runtime.

3.1.1.2 Python History

Guido van Rossum conceived Python in the late '80s and started its implementation in the Netherlands in December 1989 as a next step to ABC programming language. [28]

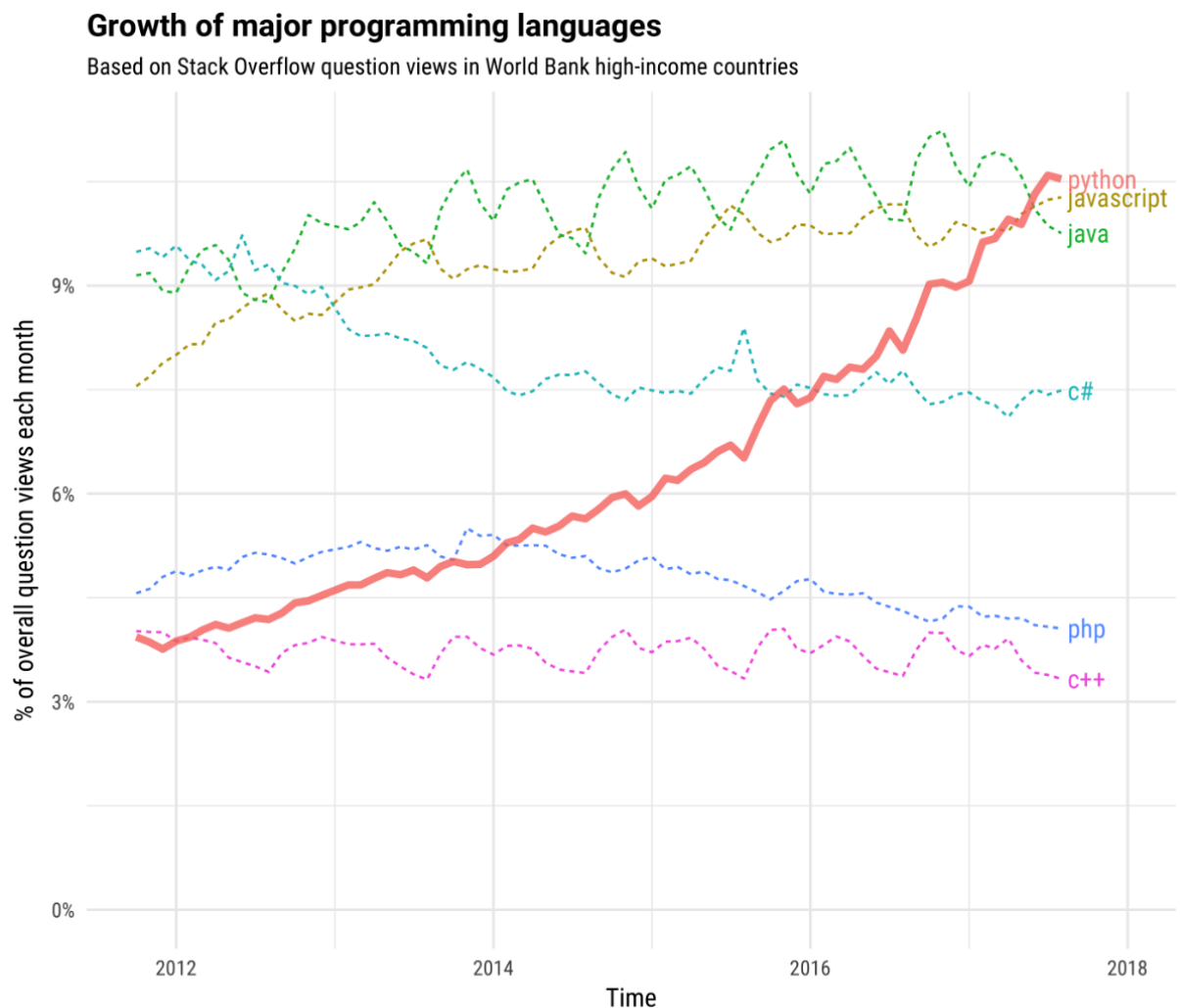
In February 1991, the labeled version 0.9.0 was published by Van Rossum, presenting, already, classes with inheritance, functions, core datatypes (dict, list, str, etc.) and exception handling.

In January 1994, Python 1.0 was released, and Van Rossum stated “Python acquired lambda, reduce(), filter() and map(), courtesy of a Lisp hacker who missed them and submitted working patches”.

In October 2000, Python 2.0 appeared, introducing list comprehensions and garbage collector while newer versions of Python 2 supported nested scoped, type unification, with statement, etc.

In December 2008, Python 3.0 was released and it was the first language to break backward compatibility. Version 3.0 had fundamental changes, such as:

- ✓ Print, from statement, became a built-in function.
 - ✓ Raw-input became input and returns string.
 - ✓ Support of function annotations.
 - ✓ Types str/Unicode were unified, bytes and bytearray were introduced.
- [29]



5. Image Source: [stackoverflow.blog 1](https://stackoverflow.blog/1)

Due to the growth of data analytics and machine learning during the 2010s, python went to become one of major programming languages reaching levels of java and JavaScript.

On July 12, 2018, Python world was shocked hearing that Van Rossum decided to remove himself totally from decision making of Python future. [30]

3.1.2 FastAPI Framework

FastAPI is a high-performance web framework for the purpose of building APIs using Python 3.6+ and standard Python type hints. FastAPI is used mostly in authentication projects and forms for web apps, deploying AI models, company management. Companies Uber and Netflix are among the FastAPI users. [31] [32]

Having a fully functional API with Uvicorn server already installed is simple as:

- ✓ Copying in main.py file the code below:

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
async def root():
    return {"message": "Hello World"}
```

- ✓ running `unicorn main:app -reload`.

3.1.2.1 Pros and Cons of FastAPI framework

Pros:

- ✓ Fast like Node.js and GO, due to Pydantic and Starlette. Pydantic offers settings management and data validation, providing user friendly errors. Starlette is a framework/toolkit – very lightweight, offering asyncio services.
- ✓ Development speed is extreme.
- ✓ Lower possibility of user errors or bugs and less time debugging.
- ✓ Not complicated, easy to learn, straightforward.
- ✓ Short, minimizing code duplication.
- ✓ Based on the open standards of APIs, OpenAPI and JSON Schema.

Cons:

- ✓ There are times that developers need to write custom validator, because using Pydantic validation is not always intuitive.
- ✓ As a new framework, fastAPI has a smaller community, comparing the other frameworks. [33]

3.1.2.2 FastAPI History

FastAPI was initially released on December 8, 2018. It was created by Sebastián Ramírez, because he was not satisfied with the existing web frameworks like Flask. Two and a half years later, Stack Overflow Developer survey named FastAPI as the third most loved web framework. ^[34]

3.1.3 TensorFlow Library

TensorFlow is an open-source library that provides workflows to train and build models. TensorFlow can be used to generate large-scale neural networks. TensorFlow includes tools for prediction, discovering, classification, perception, etc. Some cases that developers use TensorFlow are voice recognition, text base applications (like fraud detection in Finance), image recognition, video detection, etc. ^[35]

3.1.3.1 TensorFlow History

Google Brain team started working on a closed-source machine learning system, which later became TensorFlow, for internal use in Google. Its core was written in a combination of highly optimized C++, Python and CUDA, while the first well-supported language was Python. ^[36]

It was open sourced by Google in September 2015, while version 1.0.0 reached on February 11, 2017. In December of the same year Kubeflow was introduced, a project that permits TensorFlow to be deployed in Kubernetes. In March 2018, TensorFlow 1.0 was announced by Google from JavaScript. In August 2018, keras was integrated within the TensorFlow v1.10.0 package for the first time.^[37]

The release of TensorFlow 2.0 took place in September 2019. Nowadays, TensorFlow provides stable Python and C APIs, while without API backwards compatibility guarantee, is available for: C++, Java, Go, JavaScript, Swift. There are third-party packages for C#, Haskell, MATLAB, R, Julia, Scala, Crystal, OCaml and Rust. ^[38]

3.1.4 SQLAlchemy

SQLAlchemy library is used to facilitate communication between Python programming and databases. This library is mostly used as an Object Relational Mapper (ORM) tool that matches classes written in Python to tables on relational databases and transforms functional calls to SQL statements. ^[39]

3.2 DATABASE

Relational databases were selected, due to application will have to handle well-structured data, with clear relationships among them, that will demand complex queries. Relational databases follow ACID properties from transaction operations. ACID stands for:

- ✓ Atomicity: ensures the validation of all the data. If transaction fails, no updates take place at all.
- ✓ Consistency: guarantees that a processed data transaction does not damage the structural integrity of the database.
- ✓ Isolation: all transactions are isolated from the other data transactions.
- ✓ Durability: completed transactions are permanent. [40]

3.2.1 PostgreSQL

PostgreSQL is an open-source relational database with many features: user-defined types, table inheritance, sophisticated locking mechanism, foreign key referential integrity, views, rules, subquery, nested transactions, multi-version concurrency control, asynchronous replication, native microsoft windows server version, tablespaces, and point-in-time recovery. It has been developing for over two decades and is based on a proven good architecture which has created a strong perception of its users around reliability, data integrity and proper operation. [41]

3.3 FRONT-END

To create front-end, React library was selected. Redux library is used for application state management.

3.3.1 React

React is a free, open-source JavaScript library that is used for building user interfaces. React splits UI into independent, reusable pieces using components. Components act like JavaScript functions, as they accept inputs (props in React language) and return React elements. Components can be function or class components and usually are written using JavaScript Syntax Extension(JSX). Let us check an example of a function component:

```
const welcome = (props) => {  
  return(  
    <div>Welcome, {props.name}!  
    </div>  
  ) [42]
```

3.3.1.1 React History

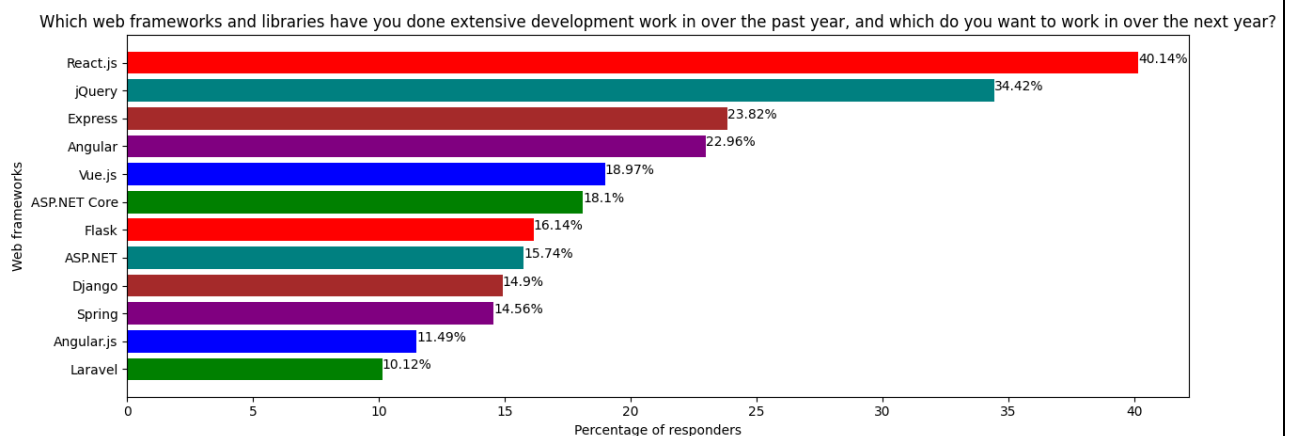
Jordan Walke, Facebook software engineer, created React influenced by XHP, an HTML component library for PHP. It was first introduced on Facebook News Feed in 2011. Instagram -acquired already by Facebook- followed in 2012. React became open-sourced in May 2013.

In 2015, the first version of React Native is released by Facebook and becomes available for iOS and Android. React Fiber was announced in April 2017 and it was replacing Stack, the React rendering algorithm till then.

React hooks were introduced as part of React 16.8 in February 2019, supporting a new easier way of handling component logic and behavior. [43]

3.3.1.2 React Now

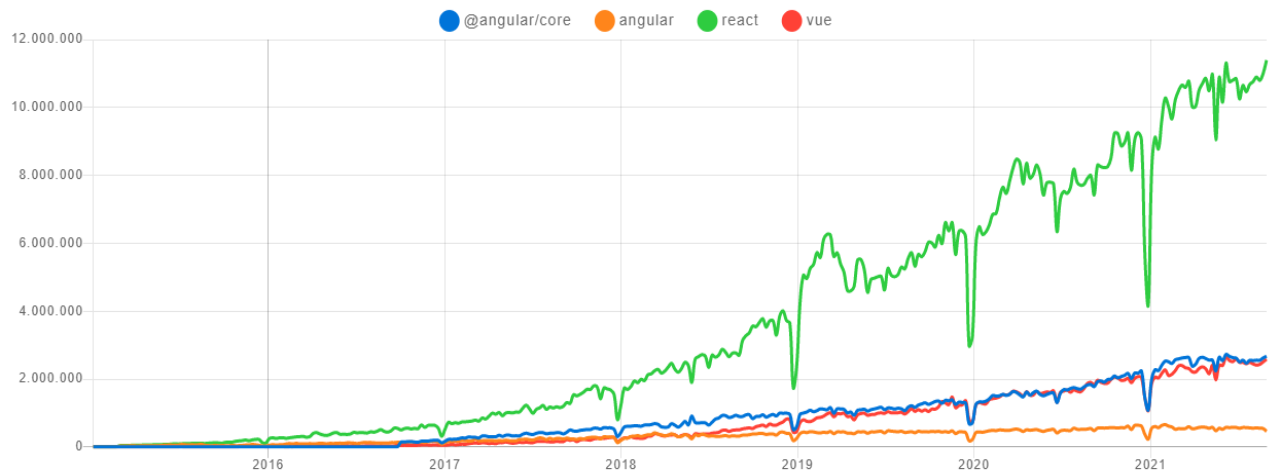
In a Stack Overflow 2021 Developer survey, 67,593 developers were asked "Which web frameworks and libraries have you done extensive development work in over the past year, and which do you want to work in over the next year?". React came first, surpassing jQuery. [44]



6. StackOverflow Survey 2021 in developers

In image 7, the download stats for React, Vue, Angular and Angular Core are presented based on the official data provided by npm.inc, the company that supports node package manager, the npm client and the npm registry.

Downloads in past All time ▾



7: Based on <https://www.npmtrends.com/@angular/core-vs-angular-vs-react-vs-vue>

3.3.2 Redux

Redux is a predictable state container for JS Apps and helps developers write apps that behave consistently and centralize application's state. React-Redux library is a library that offers Redux designed to work with the React component model, optimizing performance by re-rendering each component only when there is data change in it. [45]

4. MACHINE LEARNING

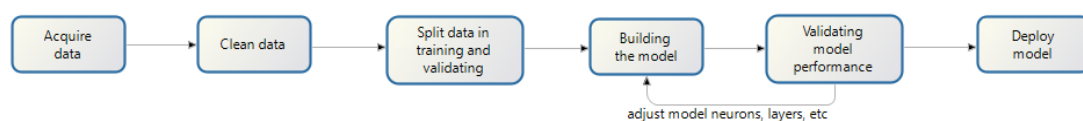
Before entering the TensorFlow world, some machine learning fundamentals should be described. Machine learning is a data analysis method that automates analytical model building. It is based on algorithms that iteratively learn from given data. [46] Today, it is used for a wide range of topics, like fraud detection, customer segmentation, email spam filtering, pricing models, image classification, etc. For the image classification problem, the (almost) only possible machine learning approach is neural networks. Neural networks are described as the method of creating models after biological neuron systems in a mathematical way.

There are two different data science approaches of machine learning: Supervised learning, like classification and regression, and unsupervised learning.

Labeled data in supervised learning make the type of the output expected. In unsupervised learning the type of result cannot be predicted. For example, a model based on labeled data with images of dogs, cats and camels answers if an image has a dog, a cat or a camel. In unsupervised learning, the model may create groups of small or big animals.

4.1 SUPERVISED LEARNING

Supervised learning includes the algorithms that use labeled historical data to get trained and based on that they predict the new data label. A neural network gets data and the correct labels, and during the training and validating process it compares its outputs with the correct labels to find errors, and then modifies the created model accordingly. The whole process looks like the one described in image 8



8. Process steps in supervised learning

As the image depicts, after acquiring and cleaning the data, data should be split in two or three categories: training data, validating data and test data. For the purposes of 'HereWeAre' images were split in two categories training and validating.

In order to evaluate a model, the model answers can be either correct or incorrect. Concentrating all the answers of the model and comparing with the real answers to find a relation layer between them, metrics are needed.

In classification problems, widely used metrics are accuracy, recall, precision, F1-score. To understand the metrics, we need a confusion matrix:

		What's predicted?	
	total	Positive	negative
What's the truth	positive	True positive	False negative
	negative	False positive	True negative

Accuracy is the quotient of the number of correct predictions (true positive and true negative) over the number of total predictions. It's preferred in balanced classes of data.

Recall is the fraction of true positive predictions among the sum of true positives and false negatives. It demonstrates the ability of the model to find true positive results.

Precision is the ratio between true positive predictions and total positive predictions (true positive plus false positive). It depicts the ability of the model of showing true positives that are actually true positives.

F1 score, given by the following formula, is a measure to check extreme differences between precision and recall:

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

In regression problems, common evaluation metrics are mean absolute error, mean squared error, root mean square error, etc. [47]

Creating supervised learning models needs skills and expertise for labeling inputs and outputs. Furthermore, training them consumes time.

4.2 UNSUPERVISED LEARNING

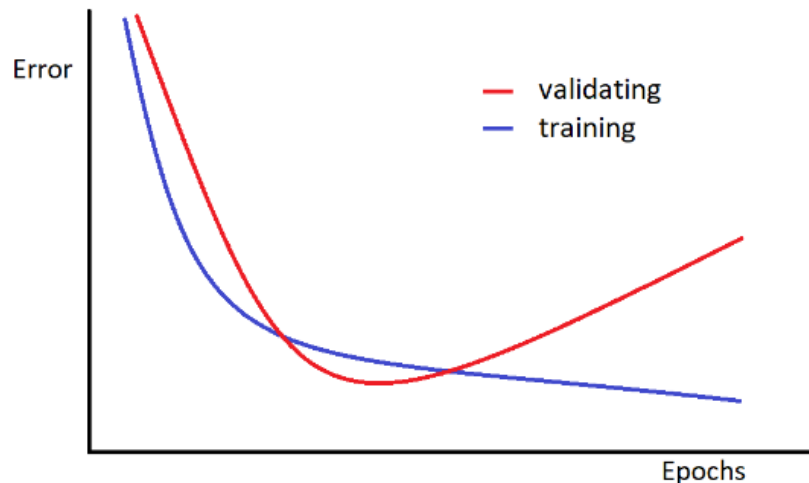
Unsupervised learning, in contrast to supervised, does not make use of historical labels, but only raw data. There are several methods that fall under the umbrella of the unsupervised learning, like:

- ✓ Clustering: setting in groups/clusters unlabeled data based on similar characteristics)
- ✓ Anomaly detection (finding the outliers of a dataset)
- ✓ Dimensional reduction (reducing features of a dataset, in order to compress or better understand trends).

Creating unsupervised learning models without human intervention may result in inaccurate outcomes. [48]

4.3 OVERFITTING

Overfitting happens when a model in training time fits too much on the training data that includes the data noise and results in high errors on validating sets.



9. Overfitting ends up in high errors in validating sets

4.4 CONVOLUTIONAL NEURAL NETWORKS

Generally, neural networks consist of node layers: an input layer, one or more hidden middle layers and an output layer. Each node has an associated weight and bias. When the output of each node surpasses the bias, gets activated and sends data to the next layer of the neural network.

Convolutional neural networks (CNN) are a tool that is used for computer and classification vision tasks. The term “convolutional” stands for the mathematical operation used in the network, called convolution – a specialized kind of linear operation, performed in the hidden layers. ^[49]

The input in a CNN is a tensor with a shape: $N \text{ inputs} * \text{input height} * \text{input width} * \text{input channels (colors)}$. Exiting from a convolutional layer, it becomes abstracted to a feature/activation map. Each convolutional neuron processes only in its sensory space sending data to the next layer. Convolution works in reducing the number of weights a neuron will receive to a significantly smaller number of learnable parameters, and hence reducing the computational resources needed.

A CNN may also include pooling layers alongside the before-mentioned convolutional layers. Pooling layers' responsibility is to combine the outputs of same layer neuron clusters into one neuron in the next layer. Most common pooling layers are average and max pooling. ^[50]

Layers have full connectivity when each node in one layer connects with all the nodes of the next layer. This may cause overfitting, so trimming connectivity methods are usually used.

The vectors of weights and bias of each node are called filters or kernels and represent a particular feature that the model tries to detect during training and to place in the input.

4.5 CREATING MODELS IN TENSORFLOW.KERAS

Explaining TensorFlow.Keras implementations used:

```
import tensorflow.keras.backend as K
```

Keras offers high-level building blocks for creating deep learning models. Backend engine of Keras is a specially designed, well-optimized tensor control library that handles low-level operations such as convolutions, tensor products, and so on. Instead of choosing a single tensor library and having Keras implementation bound to that library, more backend engines can be plugged into Keras. [51]

```
from tensorflow.keras import regularizers
```

Regularizers are used for applying penalties on layer's kernel, bias or output during optimization. These penalties are summed into the loss function, the method that predicts the error of the neural network. Applying L1 and L2 is used to reduce overfitting and large weights are a sign for that.

L1 regularization penalty is computed as: $loss = l1 * reduce_sum(abs(x))$, while L2 regularization penalty is given by: $loss = l2 * reduce_sum(square(x))$. Default values for l1 and l2 are 0.01 [52]

```
from tensorflow.keras.models import Model
```

Keras models define the structure of the layers. There are two available types:

Sequential Model	Sequential model simplifies the layer-by-layer creation of models in a sequential order. It should not be used when one of the layers has more than 1 input or 1 output tensor. Sequential example: <pre><i>from keras.models import Sequential</i> <i>from keras.layers import Dense</i> <i>model=Sequential()</i> <i>model.add(Dense(64,input_shape=8,))</i> <i>model.add(Dense(32))</i></pre>
Functional API	Functional API is used for creating models with multiple inputs and outputs. Functional API example: <pre><i>from keras.layers import Input, Dense</i> <i>from keras.models import Model</i> <i>input=Input(shape=(32,))</i> <i>layer=Dense(32)(input)</i></pre>

```
model=Model(inputs=input,outputs=layer)
//To create model with multiple inputs and outputs:
model=Model(inputs=[input1,input2],outputs=[layer1,layer2,layer3])
```

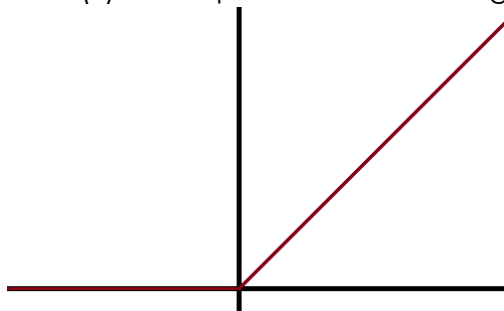
Creating the model, these functions should be filled:

- ✓ compile, when optimizers, loss function, metrics, etc., are selected. Optimizers are the algorithms or methods used to adjust the weights and the learning rate of the model, aiming to reduce the losses. Low learning rate demands more epochs in order to train the model, but it is preferred cause high learning rate may not achieve to select best weights. Loss depends on the type of the problem. If the problem is multi-class classification, categorical_crossentropy should be selected, if the problem is binary classification, then, binary_crossentropy should be selected, etc. Metric shows the metric references that will be shown apart from loss.
- ✓ Fit, when training_data, validation_data, epochs, batches, callbacks, etc., are selected. Epochs refer to the times the model passes (forward and backward) the entire dataset. Batches are the number of images the model is fed to be trained in each iteration.
Epochs = Batch Size * Iteration

```
from tensorflow.keras.layers import Dense, Dropout, Activation,
BatchNormalization, Conv2D, GlobalAveragePooling2D, MaxPool2D, Flatten
```

In layers we describe how the model is going to be created:

- ✓ Dense refers to a layer -each neuron is connected to each neuron- that is added to the model. The first argument (units) is actually the neurons of the layer.
- ✓ Dropout is used to turn off a percentage of neurons to each layer during training. It is a way to avoid overfitting during training.
- ✓ Activation refers to the activation function that describes how the output of one node emerged from the set of inputs of that node. This function could be sigmoid, binary, etc. In models created, the activation function used is a rectified linear unit, where if $x \leq 0$, $f(x) = 0$, else $f(x) = x$. Its plot is shown in image 10.



10. Plot of Relu activation function (in red)

- ✓ Conv2D refers to the numbers of filter values used for each image to train the model and to the input of the images (width, height, number of colors)
- ✓ Pooling choices takes as input the last convolutional layer parameters and reduces them by either selecting the max value or the average value of each window selected in the arguments.
- ✓ Flatten is the function that serializes a multidimensional tensor.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

Imagedatagenerator is a tool to expand the length of the dataset by modifying -zooming, rotating, blurring, etc. the images of the dataset.

from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger

Callbacks in Keras are functions that can be activated when the model has the best metric or the less loss estimated at one epoch.

from tensorflow.keras.applications.inception_v3 import InceptionV3

Inception v3 is a CNN used in image analysis and object detection. It started as a module for Googlenet. Inception-v3 is 159 layers deep and has size 92MB. ^[53] The third edition of Google's Inception CNN was originally introduced during the ImageNet Recognition Challenge. ImageNet can be thought of as a database of classified visual objects. It was trained using a dataset of 1,001 classes from more than 1 million classified images. ^[54]

Inception values for image dimensions are 299x299, if include_top argument is "True". Different image dimensions can be selected, if included_top is stated "False".

4.6 THE FLOW TO GIVE A PREDICTION

Preparing 'HereWeAre', one of the first ideas was the creation of a service that a user would provide image of the damaged car as an input and get information about that damage as an input.

To create models in TensorFlow, two image datasets were used. First dataset created by Ting Neo. ^[55] Structure of data followed the rule that "each class has its data in its own folder". So, in the validation set, "damaged cars" was a different folder from "car without damage". Second dataset is found in Peltarion site. ^[56] There all data were separated to training and validation and classes were described in csv files.

To clear datasets: first dataset had some images removed and images from the second dataset were added. The selected model should find in a where the vehicle has damages to satisfactory percentages.

Two models were selected among the others:

First model follows 4 steps/sub-models, all steps are sequential models.



9. Steps of predicting car damage in model 1

In the first two steps, it is checked that the image depicts a damaged car, while the rest examine if the damage is smaller or bigger and where the vehicle is damaged.

First check includes 4600 images that are separated equally in two classes: "car" and "notacar". 80% of the images are used for training and 20% are used for validation.

Second check includes 2300 images, which are also separated equally in two classes: "damaged" and "whole". Again, 80% of the images are used for training and 20% are used for validation.

Third check includes 1334 images and includes 6 classes: bumper (13.9% of images), door (22.9% of images), glass (9% of images), front light (9.2% images), back light (8.6% images) and bigger damage (36.3% of images). Due to the low number of images training – validation split is 85%-15%.

Fourth check includes 1510 images and includes 3 classes: front (42.78% of images), rear (29.39% of images), side (27.83% of images). Due to the low number of images training – validation split is 85%-15%.

Second model follows 2 steps/sub-models, both based in inception_v3 application.

First check includes 3450 images, that are separated equally in three classes: "notacar", "damaged" and "whole". 80% of the images are used for training and 20% are used for validation.

Second check includes 1905 images and includes 7 classes: front (24.34% of images), rear (12.17% of images), side (12.43% of images), bumper (11.11% of images), door (18.25% of images), glass (7.4% of images) and light (14.29% images). 85% of the images are used for training and 15% are used for validation.

A model that has size bigger than 200MB will be rejected.

4.7 CREATED MODELS

Creating each model, average dimensions of images should be found.

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
data_dir = "C:\\Users\\Nikitas\\Downloads\\carOrnot"
test_path = data_dir + "\\validation\\"
train_path = data_dir + "\\training\\"
dim1 = []
dim2 = []
for image_filename in os.listdir(train_path + 'notAcar\\'):
    img = imread(train_path + 'notAcar\\' + image_filename)
    if len(img.shape) == 3:
        d1, d2, colours = img.shape
    else:
        d1, d2 = img.shape
    dim1.append(d1)
    dim2.append(d2)
print(np.mean(dim1))
print(np.mean(dim2))
```

Now, average dimensions are available.

4.7.1 First model – Sequential

Sequential. model 1, created to answer if the image contains a car or not:

```
image_shape = (211, 262, 3)
image_gen = ImageDataGenerator(rotation_range=10, width_shift_range=0.1,
                               height_shift_range=0.1, rescale=1/255,
                               shear_range=0.1, zoom_range=0.1,
                               horizontal_flip=True, fill_mode='nearest')
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=image_shape,
                 activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
.....more layers added.....
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.2))
.....more layers added.....
model.add(Dense(1, activation='sigmoid'))
sdg = optimizers.SGD(lr=0.0005, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='binary_crossentropy', optimizer=sdg, metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
batch_size = 16
train_image_gen = image_gen.flow_from_directory(train_path,
                                                target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
```

```

class_mode='binary')
test_image_gen = image_gen.flow_from_directory(test_path,
target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
class_mode='binary', shuffle=False)
results = model.fit_generator(train_image_gen, epochs=100,
validation_data=test_image_gen, callbacks=[early_stop])
model.save('carOrnot11.h5')

```

- ✓ Average size of images in this model is 211*262 and the model is feed with colored images. Hence, the image shape is (211,262,3).
- ✓ ImageDataGenerator is used to apply random transformations in each image while the model is training. It shifts each given image horizontally and vertically in a percentage of 10% of total width and height. It shears, zooms-in and zooms-out by 10%, and may flip horizontally the image. To cover the generated empty space during transformation, nearest pixels will be copied and used.
- ✓ Sequential model is used to build a model as a simple stack of layers.
- ✓ Added conv2D layers use 64 filters with kernel size 3*3 and activation function 'relu'. Filter number is usually chosen to be a power of two. Due to the small dataset and the binary problem (Car or Not), the selected value of filters is 64. To determine the optimal number of layers, several attempts with different numbers of layers were explored and the one with the best results was selected. Kernel size is selected 3*3, which is a common choice, used to move kernel by 3 rows or columns of pixels in the image map. Relu activation is preferred because it does not activate all the neurons at the same time. Negative input values do not activate neuron.
- ✓ MaxPool2D layer 2*2 downsamples inputs' spatial dimension by getting the max value for each 2*2 input window.
- ✓ Flattening is used to transform the data into a 1-dimensional array for inputting it to the next layer.
- ✓ Dense layers start with 64 neurons in the first layer, the next layer has 16 neurons and the last layer is described below. To determine the optimal number of layers, several attempts with different numbers of layers were explored and the one with the best results was selected. Activation function is 'relu' again and dropout 0.2 meaning that a random 20% of the input units will be set to 0. This is done to avoid overfitting.
- ✓ In the last dense layer, both sigmoid activation and 1 neuron are selected to generate one value, like a regression task. If the value is higher than 0.5 then the model predicts: "It's a car"
- ✓ Stochastic Gradient Descent(SGD) Optimizer is the iterative method to optimize the model. The selected learning rate is 0.0005, in order to train slowly the model, making tiny updates to the weights of the neuron network. Momentum is usually set to 0.9. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. [57]

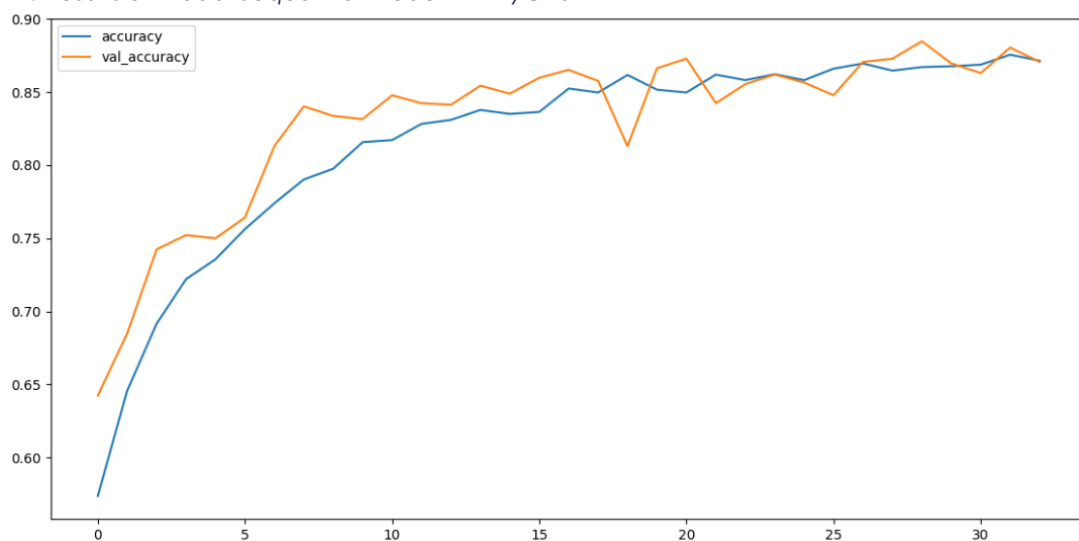
- ✓ Cross entropy is used during the model training to compare the predicted probabilities of the model with the real classes in order to decrease loss. Accuracy metric is added to be shown during epochs.
- ✓ Early stops are used when the model does not show smaller losses in validation data for 5 consecutive epochs, restoring the best weights it got during training.
- ✓ Models usually are fed with batch sizes of 32 images, but this model will be fed with 16 images at each iteration, due to the small datasets, aiming for better performance.
- ✓ Flow from directory identifies classes automatically. We have two different folders, one for 'cars' and one for not 'cars', so the class mode must be stated as 'binary'.
- ✓ Shuffle in validation set is set False, to allow generator map filenames to the batches that are yielded by the data generator.
- ✓ Fit generator gets train image generator and test image generator to train the model for 100 epochs. This procedure will automatically stop if val_loss does not get smaller in 5 consecutive epochs.

```

125 print(classification_report(test_image_gen.classes, predictions))
126
Run: caromot x
warnings.warn('Model.predict_generator is deprecated and '
precision recall f1-score support
0 0.89 0.87 0.88 460
1 0.87 0.89 0.88 460
accuracy 0.88 920
macro avg 0.88 0.88 0.88 920
weighted avg 0.88 0.88 0.88 920
Process finished with exit code 0

```

11. Results of model Sequential Model 1 in PyCharm



12. Plot of accuracy in training data & accuracy in validation data for each epoch in Model 1

Accuracy, precision, recall, f1-score = 88%, while 32 epochs took place. Epoch 28 generated the model with the higher accuracy for validation data. Epochs stopped when val_loss value was lower.

Sequential. model 2, created to answer if the image contains a vehicle damaged:

```
image_shape = (221, 319, 3)
image_gen = ImageDataGenerator(rotation_range=10, width_shift_range=0.1,
                               height_shift_range=0.1, rescale=1/255,
                               shear_range=0.1, zoom_range=0.1,
                               horizontal_flip=True, fill_mode='nearest')

model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=image_shape,
                 activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
.....more layers added.....
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.2))
.....more layers added.....
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
early_stop = EarlyStopping(monitor='val_loss', patience=10,
                           restore_best_weights=True)
batch_size = 16
train_image_gen = image_gen.flow_from_directory(train_path,
                                                target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
                                                class_mode='binary')
test_image_gen = image_gen.flow_from_directory(test_path,
                                              target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
                                              class_mode='binary', shuffle=False)
results = model.fit_generator(train_image_gen, epochs=250,
                             validation_data=test_image_gen, callbacks=[early_stop])
model.save('wholeordamaged23a.h5')
```

- ✓ Average size of images in this model is 221*319 and the model is feed with colored images. Hence, the image shape is (221,319,3).
- ✓ Next steps are the same with the previous model as it is a binary classification problem. In this model, the selected optimizer for this model is Adam (Adaptive Moment Estimation). Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.^[58] Best properties of the AdaGrad and RMSProp algorithms are combined in Adam, generating an optimization algorithm that can handle sparse gradients on noisy problems. Default learning rate is 0.001.
- ✓ Early stops are used when the model does not show smaller losses in validation data for 10 consecutive epochs, restoring the best weights it got during training.

- ✓ Flow from directory identifies classes automatically. We have two different folders, one for 'whole' and one for 'damaged', so the class mode must be stated as 'binary'.
- ✓ Fit generator gets train image generator and test image generator to train the model for 250 epochs. This procedure will automatically stop if val_loss does not get smaller in 10 consecutive epochs.

```

122
123 model.save('wholeordamaged23a.h5')
124
125 losses = nd.DataEpoch(model.history.history)

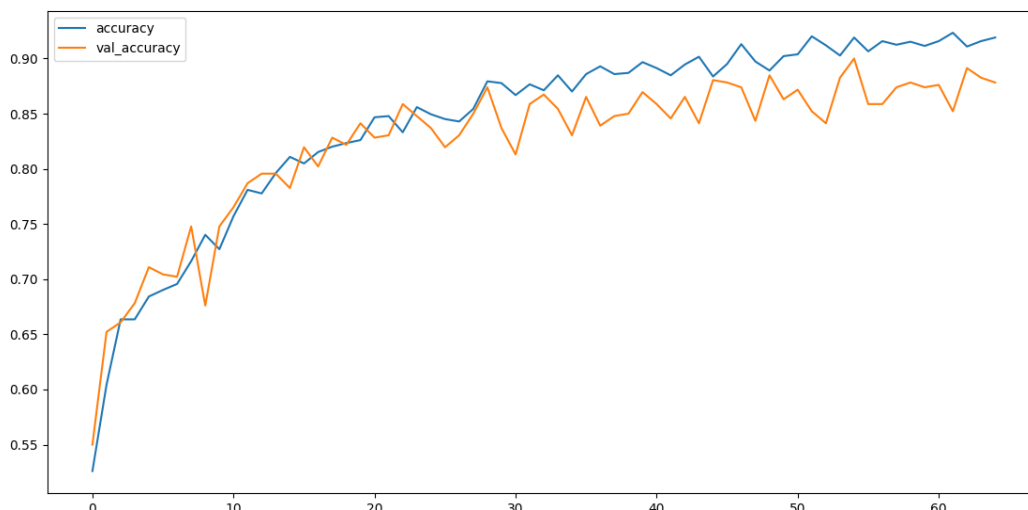
```

```

Run: newPredict x
115/115 [=====] - 210s 2s/step - loss: 0.2052 - accuracy: 0.9140 - val_loss: 0.2897 - val_accuracy: 0.8783
Epoch 60/250
115/115 [=====] - 216s 2s/step - loss: 0.1994 - accuracy: 0.9240 - val_loss: 0.3021 - val_accuracy: 0.8739
Epoch 61/250
115/115 [=====] - 216s 2s/step - loss: 0.1985 - accuracy: 0.9184 - val_loss: 0.2835 - val_accuracy: 0.8761
Epoch 62/250
115/115 [=====] - 218s 2s/step - loss: 0.1825 - accuracy: 0.9295 - val_loss: 0.3469 - val_accuracy: 0.8522
Epoch 63/250
115/115 [=====] - 251s 2s/step - loss: 0.1990 - accuracy: 0.9177 - val_loss: 0.2875 - val_accuracy: 0.8913
Epoch 64/250
115/115 [=====] - 261s 2s/step - loss: 0.1994 - accuracy: 0.9183 - val_loss: 0.3390 - val_accuracy: 0.8826
Epoch 65/250
115/115 [=====] - 239s 2s/step - loss: 0.2094 - accuracy: 0.9168 - val_loss: 0.3098 - val_accuracy: 0.8783
C:\Python\Python3810_64\lib\site-packages\tensorflow\python\keras\engine\training.py:1905: UserWarning: 'Model.predict_generator' is
warnings.warn('Model.predict_generator' is deprecated and '
precision recall f1-score support
0 0.87 0.92 0.90 230
1 0.92 0.87 0.89 230
accuracy 0.89 460
macro avg 0.89 0.89 0.89 460
weighted avg 0.89 0.89 0.89 460

```

13. Results of Sequential Model 2 in PyCharm



14. Plot of accuracy in training data & accuracy in validation data for each epoch in Model 2

Accuracy, precision, recall, f1-score = 89%, while 65 epochs took place. Epoch 55 generated the model with the higher accuracy for validation data. Epochs stopped when val_loss value was lower.

Sequential. model 3, created to answer is the damage small and if yes where is it? Unlike previous problems, model 3 meets a multi-class classification. Additionally, the data structure is different. Data are not organized in different folders based on their class, but they are placed in one folder, while a csv file describes for every image its class. Data are separated 85% train – 15% validation (validation is stated as test in the algorithm). In 80%-20% separation case generated models with low metrics.

```

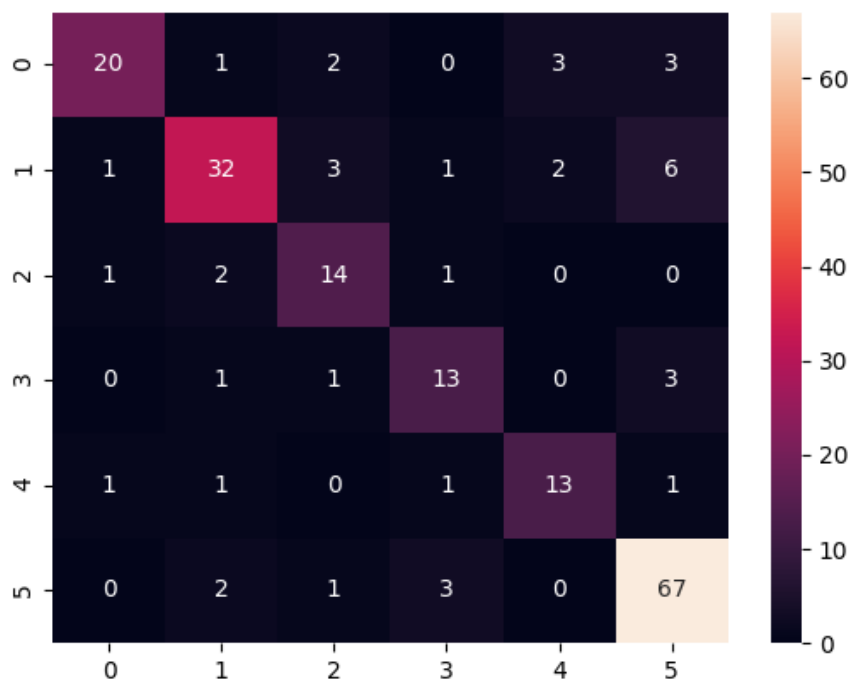
traindf = pd.read_csv('NewTrain.csv')
testdf = pd.read_csv('NewTest.csv')
traindf.id = traindf.id.astype(str)
testdf.id = testdf.id.astype(str)
traindf.id += ".jpeg"
testdf.id += ".jpeg"
datagen = ImageDataGenerator(rotation_range=10, width_shift_range=0.1,
                             height_shift_range=0.1, rescale=1/255,
                             shear_range=0.1, zoom_range=0.1,
                             horizontal_flip=True, fill_mode='nearest')
train_generator = datagen.flow_from_dataframe(
    dataframe=traindf,
    directory="C:\\Users\\Nikitas\\Downloads\\preprocessed\\image\\train",
    x_col="id", y_col="label", subset="training", batch_size=8, shuffle=True,
    class_mode="categorical", target_size=(224, 224))
test_datagen=ImageDataGenerator(rescale=1./255.)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=testdf,
    directory="C:\\Users\\Nikitas\\Downloads\\preprocessed\\image\\test",
    x_col="id", y_col="label", batch_size=8, shuffle=False,
    class_mode="categorical", target_size=(224, 224))
image_shape = (224, 224, 3)
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=image_shape,
    activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
.....more layers added.....
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.2))
.....more layers added.....
model.add(Dense(6, activation='softmax'))
sdg = optimizers.SGD(lr=0.0001, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sdg, metrics=['accuracy'])
model.summary()
early_stop = EarlyStopping(monitor='val_loss', patience=15,
    restore_best_weights=True)
train_image_gen.class_indices
results = model.fit_generator(train_generator, validation_data=test_generator,
    epochs=150, callbacks=[early_stop] )
model.save('newcheck14.h5')

```

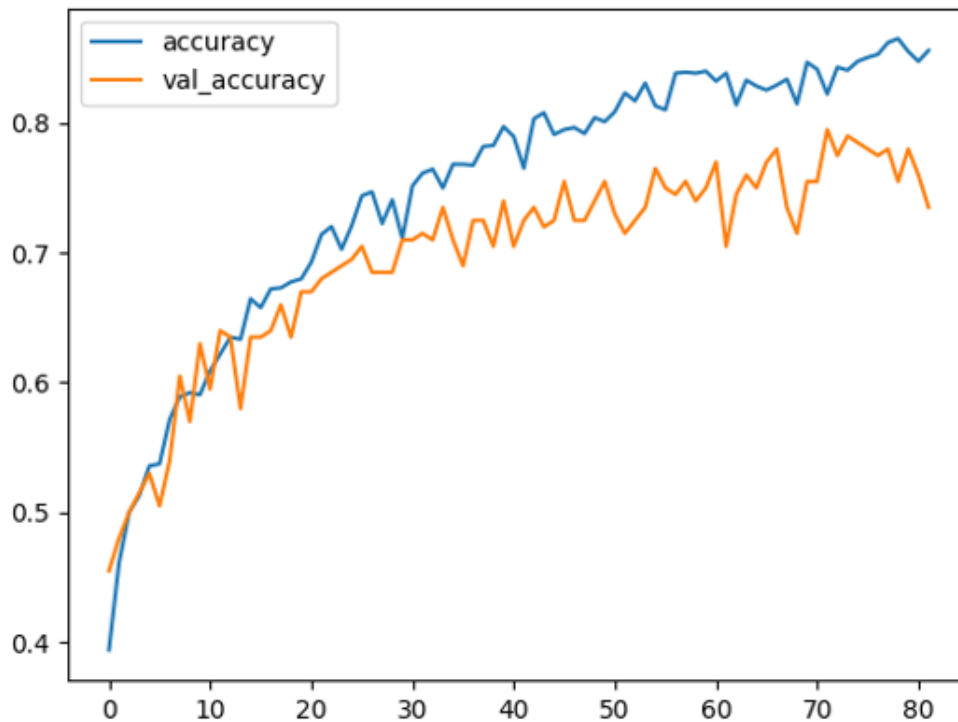
- ✓ Using pandas dataframe, train and test generators are created through the function `flow_from_dataframe`. Parameters are: the columns of the dataframes that state the filename and the class of the image, the directory of the images, the batch size, the `class_mode`, etc. Batch size selected is 8, which is smaller than the two aforementioned models, since this dataset responds better in this size. Class mode is categorical.
- ✓ Average size of images in this model is 224*224 and the model is feed with colored images. Hence, the image shape is (221,224,3).
- ✓ Last dense layer has 6 neurons, the number of the different classes of images, while its activation function is softmax. Softmax normalizes the output of the network to a probability distribution over predicted output classes.
- ✓ Fit generator gets train image generator and test image generator to train the model for 150 epochs. This procedure will automatically stop if `val_loss` does not get smaller in 10 consecutive epochs.

	precision	recall	f1-score	support
bumper	0.87	0.69	0.77	29
door	0.82	0.71	0.76	45
glass_shatter	0.67	0.78	0.72	18
head_lamp	0.68	0.72	0.70	18
tail_lamp	0.72	0.76	0.74	17
unknown	0.84	0.92	0.88	73
accuracy			0.80	200
macro avg	0.77	0.76	0.76	200
weighted avg	0.80	0.80	0.79	200

15. Classification Report for Sequential Model 3 in PyCharm



16. Confusion Matrix for Sequential model 3 in PyCharm



17. Plot of accuracy in training data & accuracy in validation data for each epoch in model3

Accuracy is 80% and while it's a multiclassification problem, total precision, recall and f1-score are quite high. The confusion matrix in image 16 depicts the predictions of the model for each class. For example, 73 images of the class 'unknown'(bigger damage) had 67 right predictions, while 2 images were predicted as "door", 1 image was predicted as "glass_shutter" and 2 images were predicted as "head_lamp". Epoch 72 had the higher accuracy for validation data. Epochs stopped when val_loss value was lower.

Sequential. model 4, created to answer where is the big damage?:

```

image_shape = (193, 263, 3)
image_gen = ImageDataGenerator(rotation_range=10, width_shift_range=0.1,
                               height_shift_range=0.1, rescale=1/255,
                               shear_range=0.1, zoom_range=0.1,
                               horizontal_flip=True, fill_mode='nearest')

model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=image_shape,
                activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2)))
.....more layers added.....
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.2))
.....more layers added.....
sdg = optimizers.SGD(lr=0.0001, decay=1e-6, momentum=0.9, nesterov=True)
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=sdg, metrics=['accuracy'])

```

```

model.summary()
early_stop = EarlyStopping(monitor='val_loss', patience=30,
restore_best_weights=True)
batch_size = 8
train_image_gen = image_gen.flow_from_directory(train_path,
target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
class_mode='categorical')
test_image_gen = image_gen.flow_from_directory(test_path,
target_size=image_shape[:2], color_mode='rgb', batch_size=batch_size,
class_mode='categorical', shuffle=False)
train_image_gen.class_indices
results = model.fit_generator(train_image_gen, epochs=150,
validation_data=test_image_gen,
callbacks=[early_stop])
model.save('carpart30c.h5')

```

- ✓ Average size of images in this model is 193*263 and the model is feed with colored images. Hence, the image shape is (193,263,3).
- ✓ Next steps are the same with the previous models: selected optimizer is SGD with learning rate 0.0001, decay acts like l2, to keep weights in lower values. Batch size is 8 as it brings better results for this data set than bigger sizes.
- ✓ Early stops are used when the model does not show smaller losses in validation data for 30 consecutive epochs, restoring the best weights it got during training.
- ✓ Model gets trained for 250 epochs. This procedure will automatically stop if val_loss does not get smaller in 30 consecutive epochs.

```

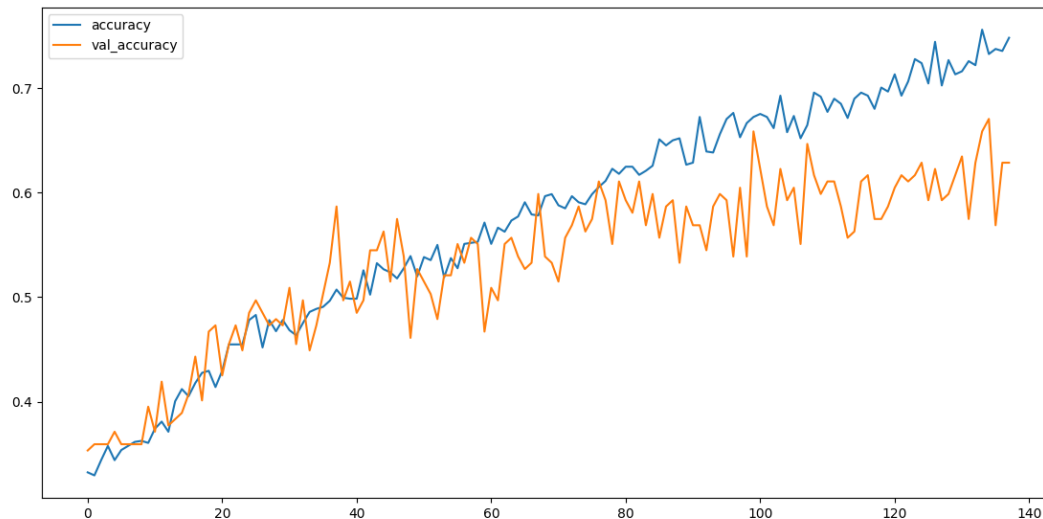
129/129 [=====] - 211s 2s/step - loss: 0.6374 - accuracy: 0.7418 - val_lo
Epoch 138/150
129/129 [=====] - 206s 2s/step - loss: 0.6568 - accuracy: 0.7438 - val_lo
C:\Python\Python3810_64\lib\site-packages\tensorflow\python\keras\engine\training.py:1905: UserWar
warnings.warn('Model.predict_generator' is deprecated and '
[0 0 1 0 0 0 0 1 0 0 2 0 1 2 0 0 2 1 0 0 0 0 0 0 2 1 0 0 0 0 2 0 0 0 2 0 0
0 1 0 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 0 0 1 1 1 1 0 1 0 1 1 1 1
1 0 0 2 0 1 0 1 1 0 1 2 1 1 0 2 0 1 0 1 1 0 0 0 1 1 1 1 1 1 0 1 0 2 1 1 0
0 2 2 2 1 2 2 1 1 0 2 0 2 2 2 2 0 2 0 1 1 2 2 0 2 2 2 2 0 2 2 0 2 2 0 2 1
2 0 2 0 2 2 2 2 1 0 2 2 2 0 2 2 2 0]
          precision    recall  f1-score   support

0         0.58        0.75        0.66         60
1         0.67        0.55        0.60         51
2         0.73        0.62        0.67         56

 accuracy
macro avg   0.66        0.64        0.64        167
weighted avg 0.66        0.65        0.65        167

```

18. Results of model "CarPart30c" in Pycharm



19. Plot of accuracy in training data & accuracy in validation data for each epoch in model4

Accuracy: 65%, precision:66%, recall:64% and f1-score:64%. Not high metrics.

Test Sequential Model

To check if the model predicts correctly, 70 images were used to test it. 4 of the are images without a car, 1 image has a car without damage and all other images have car damaged in several ways.



20. Test images

Unfortunately, the model did not find a car in 7 images and did not find damages in 5. Generally, accuracy was 57%. Quite low. Model failed when the image included people and when the image is a bit blur.

4.7.2 Second model – InceptionV3

Inception module is selected to generate the second model, since it combines:

- ✓ high-performance on convolutional neural networks,

- ✓ ability to extract features from input data at varying scales through varying filter sizes
- ✓ minimal increase in utilization of computing resources. [59]

InceptionV3. model1, created to answer if the image contains a vehicle with damages (0:damaged, 1:not_damaged, 2:not a car):

```

K.clear_session()
n_classes = 3
img_width, img_height = 224,224
data_dir = "C:\\Users\\Nikitas\\Downloads\\insurance_project-
master\\insurance_project-master\\new\\data1a"
validation_data_dir = data_dir + "\\validation\\"
train_data_dir = data_dir + "\\training\\"
nb_train_samples = 2760
nb_validation_samples = 690
batch_size =64
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1. / 255)
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')
validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
inception = InceptionV3(weights='imagenet', include_top=False,
input_shape=(224,224,3))
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(64)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(n_classes, kernel_regularizer=regularizers.l2(0.005),
activation='softmax')(x)
model = Model(inputs=inception.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer =
ModelCheckpoint(filepath="C:/Users/Nikitas/Documents/is_car_damaged2.hdf5",
verbose=1, save_best_only=True)
csv_logger = CSVLogger("C:/Users/Nikitas/Documents/history_is_car_damaged2.log")

```

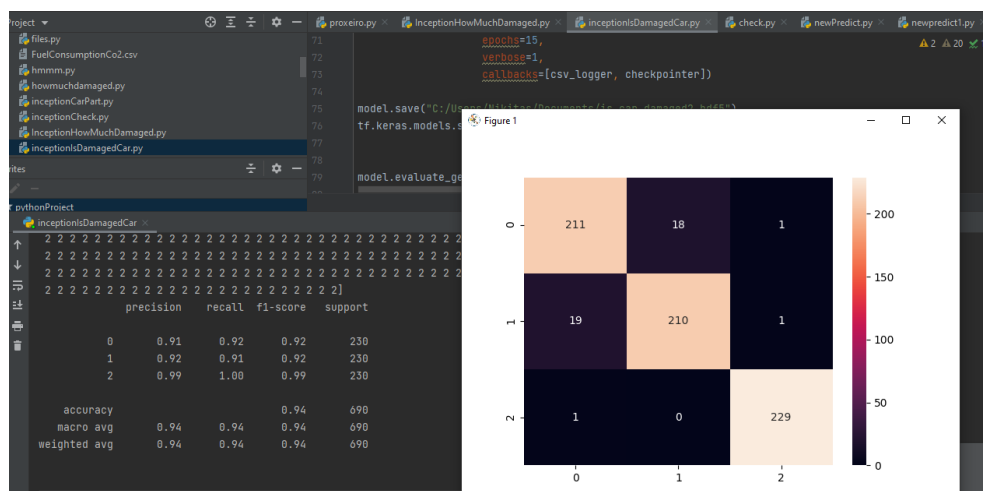
```

history = model.fit_generator(train_generator,
                             steps_per_epoch = nb_train_samples // batch_size,
                             validation_data=validation_generator,
                             validation_steps=nb_validation_samples // batch_size,
                             epochs=15,
                             verbose=1,
                             callbacks=[csv_logger, checkpointer])
model.save("C:/Users/Nikitas/Documents/is_car_damaged2.hdf5")
tf.keras.models.save_model(model,"C:/Users/Nikitas/Documents/is_car_damaged2")

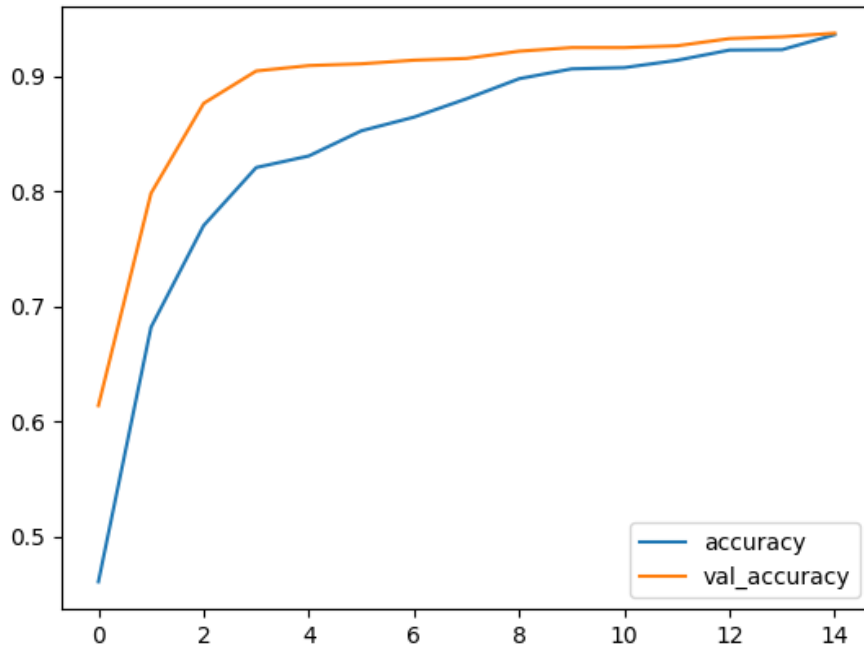
```

Preparation of data generators is exactly the same with the aforementioned models. 2760 images are used to train the model and 690 images are used to validate the model. Let's explore how inception V3 model 1 is created:

- ✓ The pre-trained weights of ImageNet are selected in inception V3 function argument "weights"
- ✓ In the same function, include_top refers to the fully-connected layer at the top of the network, if it is included or not. If it is True, image shape should be (299,299,3). Since image shape for the dataset is (224,224,3), we stated it as false.
- ✓ In the output of this function, GlobalAveragePooling2D with default arguments is applied. That is: the last channels' 4d tensor with shape (batch_size, rows, cols, channels) is transformed to 2D tensors with shape (batch_size, channels)
- ✓ Dense 64 is selected, since bigger number brought quite fast results with high accuracy and low loss in training data, but not so good metrics in validation data. Last layer of dense has 3 units, as the classes of the classification problem.
- ✓ Model is ready to get trained and compiled. Learning rate is declared at 0.0001.
- ✓ ModelCheckpoint checks the model with the best weights, that is only if a next epoch generates a model with better weights the previous one will be overwritten. Criterion is the lower val_loss.
- ✓ CVS logger saves the metrics of all epochs, in order to create plots.
- ✓ Model gets trained for 15 epochs.



21.17. Results of model "InceptionIsDamagedCar" in Pycharm



22. Plot of accuracy in training data & accuracy in validation data for each epoch in InceptionV3 model

Accuracy: 94%, precision:94%, recall:94% and f1-score:94%. These are high metrics from a model that run only 15 epochs.

InceptionV3. model 2, Created to answer where vehicle is damaged (0: front, 1: rear, 2: side, 3: bumper, 4: door, 5: glass, 6: light):

```
K.clear_session()
n_classes = 7
img_width, img_height = 224,224
data_dir = "C:\\Users\\Nikitas\\Downloads\\preprocessed\\new"
validation_data_dir = data_dir + "\\validation\\"
train_data_dir = data_dir + "\\train\\"
nb_train_samples = 1618
nb_validation_samples = 288
batch_size = 16
train_steps_per_epoch = np.ceil((nb_train_samples/batch_size)-1)
validation_steps_per_epoch = np.ceil((nb_validation_samples/batch_size)-1)
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1. / 255)
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')
validation_generator = test_datagen.flow_from_directory(
```

```

validation_data_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical',
shuffle=False)
inception = InceptionV3(weights='imagenet', include_top=False,
input_shape=(224,224,3))
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(128)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(n_classes, kernel_regularizer=regularizers.l2(0.005),
activation='softmax')(x)
model = Model(inputs=inception.input, outputs=predictions)
model.compile(optimizer=SGD(learning_rate=0.0001, momentum=0.9),
loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer =
ModelCheckpoint(filepath="C:/Users/Nikitas/Documents/predict_all.hdf5", verbose=1,
save_best_only=True)
csv_logger = CSVLogger("C:/Users/Nikitas/Documents/predict_all.log")
history = model.fit_generator(train_generator,
steps_per_epoch = train_steps_per_epoch,
validation_data=validation_generator,
validation_steps= validation_steps_per_epoch,
epochs=80,
verbose=1,
callbacks=[csv_logger, checkpointer])
model.save("C:/Users/Nikitas/Documents/predict_all.hdf5")
tf.keras.models.save_model(model,"C:/Users/Nikitas/Documents/predict_all")

```

Dataset was created by mixing the two datasets, cleaning and classifying images into folders for the next classes: front, rear, side, bumper, door, glass, light. 1618 images are used to train the model and 288 images are used to validate the model. Let's explore how inception V3 model 1 is created:

- ✓ The pre-trained weights of ImageNet are selected in inception V3 function argument "weights"
- ✓ In the same function, include_top is stated false since image shape for the dataset is (224,224,3).
- ✓ In the output of this function, GlobalAveragePooling2D with default arguments is applied.
- ✓ Dense 128 is selected, since this is the number that brought highest metrics in validation data. Several attempts were made. Last layer of dense has 7 units, as the classes of the classification problem.

- ✓ Model is ready to get trained and compiled. Learning rate is declared at 0.0001.
- ✓ ModelCheckpoint checks the model with the best weights, that is only if a next epoch generates a model with better weights the previous one will be overwritten.
- ✓ CVS logger saves the metrics of all epochs, in order to create plots.
- ✓ Model gets trained for 80 epochs.

	precision	recall	f1-score	support
0	0.79	0.74	0.76	70
1	0.67	0.47	0.55	34
2	0.68	0.43	0.53	35
3	0.67	0.94	0.78	33
4	0.82	0.94	0.87	52
5	0.77	0.87	0.82	23
6	0.80	0.85	0.82	41
accuracy			0.76	288
macro avg	0.74	0.75	0.73	288
weighted avg	0.75	0.76	0.75	288

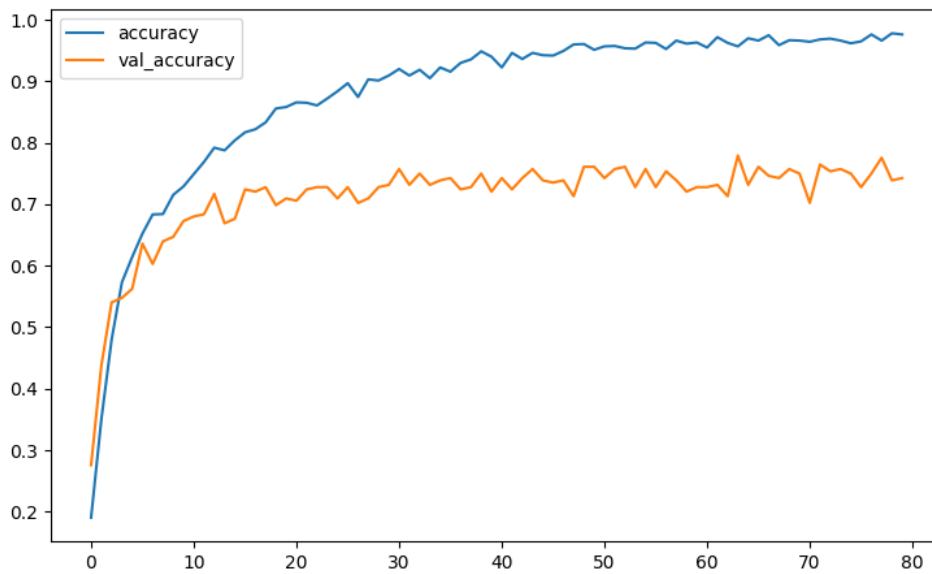
23. Classification Report for InceptionV3 predict_all

Classification report in image 23 shows that the model has 76% accuracy but recall is relatively low if the car is damaged in the rear or the side.



24. Confusion matrix of model InceptionV3 model 2 predict_all

Confusion matrix in image 24 shows that when the damage occurs on the rear-end of the car, the model may give wrong answers, such as: on front or on the bumper. Additionally, when the damage is on the side, the model may give a result of: damage on the front, side or the door.



25..Plot of accuracy in training data & accuracy in validation data for each epoch in InceptionV3 model 2

To check if the model gives correct predictions, the same 70 images were used to test it. The model responded rather well, giving 80% correct answers, while the wrong answers were mainly the following: predicted damaged door instead of damage on the side part. To be clear, when damage is referred as on the side part means that the damage is bigger than a door damaged.

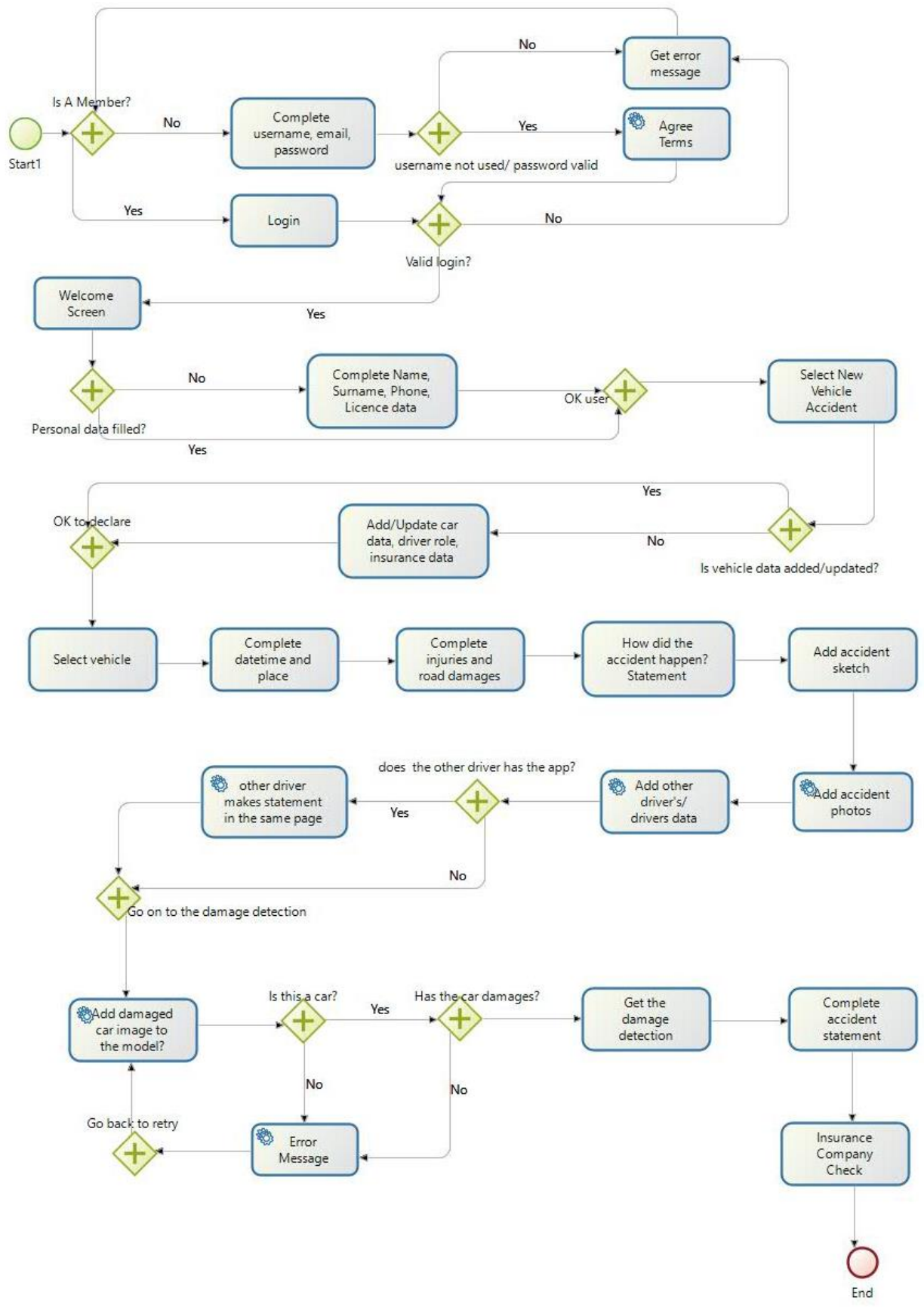
So, until more images of damaged cars become available, we choose the second model, since this one provides more accurate results.

5. CREATING THE APPLICATION

5.1 CREATE APPLICATION FLOW

At start, the application flow was created:

- ✓ User visits web application. First step is "login". If the user is not a member, he should register first. On register, user has to choose an email and a username that are not already used and has to agree to terms and conditions. Otherwise, app returns to register screen.
- ✓ When the user logs in, he is directed to home screen. If he has not filled his personal data, he is directed to fill the profile form.
- ✓ If he wants to declare a vehicle accident, he should already have added the vehicle to his vehicle list. Otherwise, he has to go the vehicle screen in order to fill forms about the vehicle and its insurance.
- ✓ In the accident page, he selects the vehicle he was driving in the accident, add place and time data and complete the fields about injuries due to the accident and road problems.
- ✓ Next step is to add the reasons of the accident and describe how accident happened and whose fault it was.
- ✓ Next, he adds accident photos and draws the accident.
- ✓ Form also includes fields about the other driver that should be filled. If there were more than two vehicles in the accident, user can add all of them. Each other driver can respond in the same page in the same way.
- ✓ Last step before completing the accident statement is to add an image in the damage detection page, to let know the insurance company the damage degree.
- ✓ State accident statement as completed.
- ✓ Insurance company employees have access to accident data, and when they want they can state accident as case closed.



26. Flow of accident statement

5.2 DEVELOPING EACH CONTAINER

Developing a vehicle accident statement app as a system of microservices starts by separating each microservice, related to its business objective, and declaring the ways of communication between them if that is necessary. So, in this app there are 3 microservices in back end: authentication system, accident statement system and detect the vehicle damage detection system. Each microservice has open apis, so restrictions are set in almost every call to match the restrictions of the app, and some of them are described below.

In development stage, to start the app, first step is to have Desktop Docker running. Each technology used in the app should be written in the requirements.txt file. Starting the code, server.py is being written. It includes the function that returns an app using the FastAPI framework.

Then, Dockerfile is created, as it is described in the code below: pulling the docker image of python 3.8, setting the environment, copying requirements to the app, installing the necessary dependencies, copying all files to the app and start.

```
FROM python:3.8
RUN cat /etc/issue
RUN pip3 install --upgrade pip
COPY ./requirements.txt ./app/requirements.txt
# install requirements
RUN pip3 install -r ./app/requirements.txt
RUN rm -r /root/.cache
# copy files and start
COPY . /app
WORKDIR /app
CMD ['bash']
```

Next step is the creation of the docker-compose.yaml file:

```
version: '3.7'
services:
  server:
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    command: uvicorn app.api.server:app --reload --workers 2 --host 0.0.0.0 --port 8000
    env_file:
      - ./env
    image: ${APP_IMG}:${APP_TAG}
    ports:
      - 8000:8000
```

```

depends_on:
  - db

db:
  image: postgres:12.1-alpine
  volumes:
    - postgres_data:/var/lib/postgresql/data/
  env_file:
    - ./env
  ports:
    - 5432:5432

```

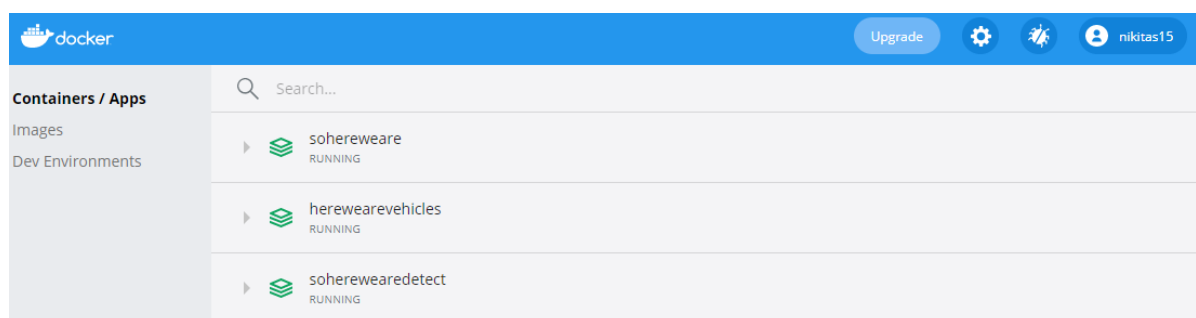
Here, two volumes of the container are configured. Different volumes permit the database to keep running when the server volume is down. As env is configured, the connection between FastAPI and PostgreSQL is established whenever the app starts. Instead of docker commands, makefile is used to simplify the commands:

- ✓ "docker build -f Dockerfile -t \${APP_IMG}:\${APP_TAG} ." is replaced by "make build-img"
- ✓ "docker-compose up -d" is replaced by "make compose-up".

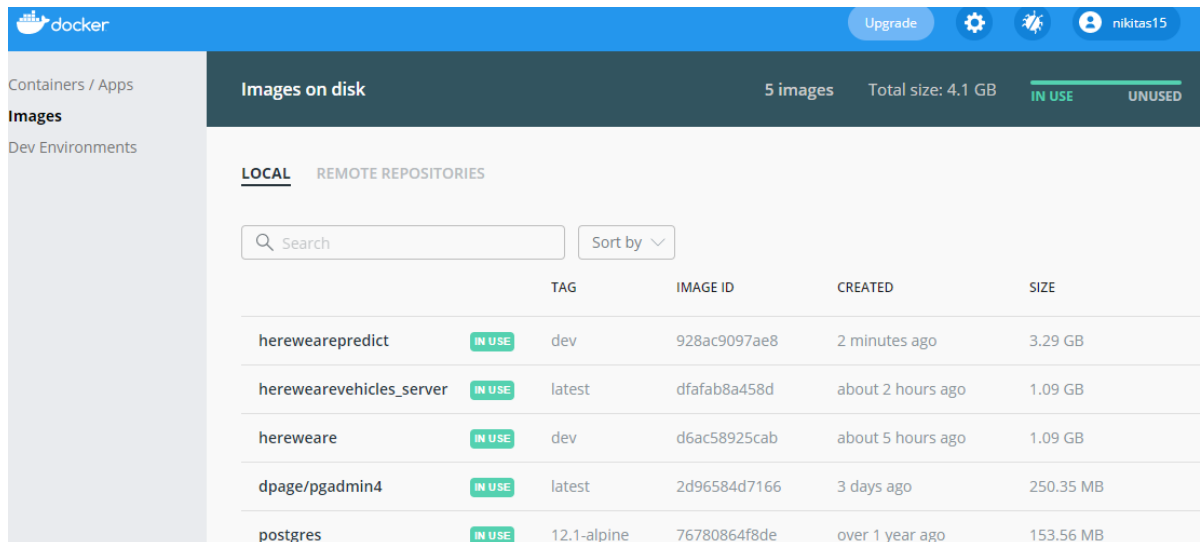
So, container with server and db is up and running. Following, migration will create the database architecture. As before, using the makefile:

- ✓ 'docker-compose exec server alembic revision -m "\$(m)" ' is replaced by make make-migration
- ✓ 'docker-compose exec server alembic upgrade head' is replaced by 'make migrate'.

For the creation of each microservice, the same steps are followed, changing the ports. In development, when an app is in still a local system, a network between the docker containers is needed. When all containers are up, docker pages "Containers/Apps" and "Images" should contain them with an indication that they are "IN USE".

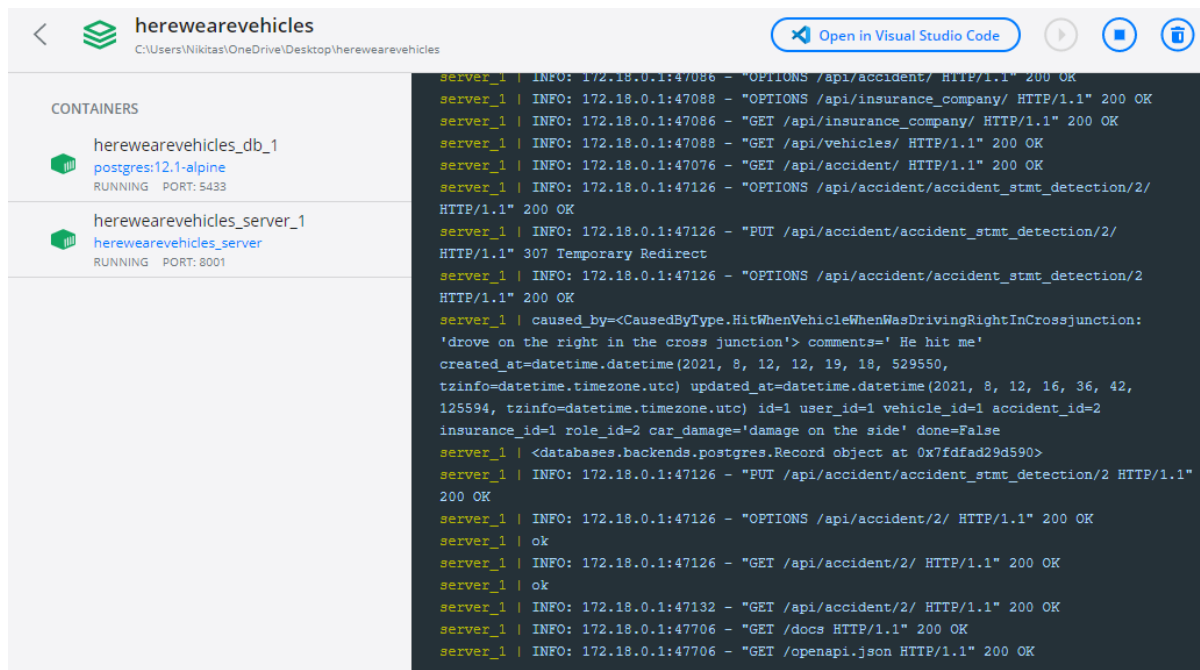


27. Containers of 'HereWeAre'



28. Images of 'HereWeAre'

In each container, server shows the outcome of each request.

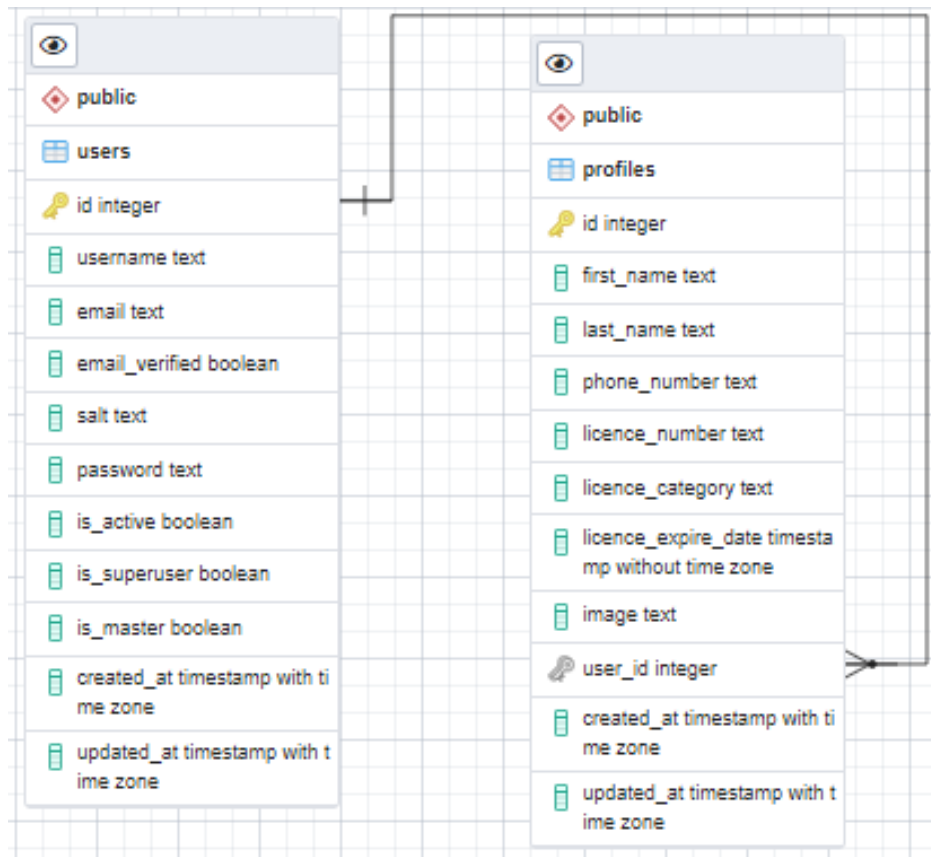


29. Logs of herewearevehicles container

5.3 AUTHENTICATION MICROSERVICES

The authentication microservice verifies that the user has the rights to use the app, checks the layer of the authorization of their actions in the app, and holds their personal data.

Starting from the database of the authentication system, it includes two tables: users, profiles. The database schema is shown below:



30. Entity Relationship Diagram for authentication database

User table is used for the authentication and the authorization of each user. A user has to provide a unique username, a unique email and a password to become a member. App checks the uniqueness of email and username, then generates text, known as salt, to hash user password and saves them in the database, returning a token to user by JWT library based on username. This token is used every time they make a request.

Token creation using JWT:

```
def create_access_token_for_user(
    self, *,
    user: UserBase,
    secret_key: str = str(SECRET_KEY),
    audience: str = JWT_AUDIENCE,
    expires_in: int = ACCESS_TOKEN_EXPIRE_MINUTES,
) -> str:
    if not user or not isinstance(user, UserBase):
        return None
    jwt_meta = JWTMeta(
        aud=audience,
        iat=datetime.timestamp(datetime.utcnow()),
        exp=datetime.timestamp(datetime.utcnow() + timedelta(minutes=expires_in)),
    )
    jwt_creds = JWTCreds(sub=user.email, username=user.username)
    token_payload = JWTPayload(
        **jwt_meta.dict(),
```

```

        **jwt_creds.dict(),
    )
    access_token = jwt.encode(token_payload.dict(), secret_key,
algorithm=JWT_ALGORITHM)
    return access_token

```

Get username from token:

```

def get_username_from_token(self, *, token: str, secret_key: str) -> Optional[str]:
    try:
        decoded_token = jwt.decode(token, str(secret_key),
audience=JWT_AUDIENCE, algorithms=[JWT_ALGORITHM])
        payload = JWTPayload(**decoded_token)
    except (jwt.PyJWTError, ValidationError):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Could not validate token credentials.",
            headers={"WWW-Authenticate": "Bearer"},
        )
    return payload.username

```

User APIs

users		^
POST	/api/users/ Users: Register-New-User	v
POST	/api/users/login/token/ Users: Login-Email-And-Password	v
GET	/api/users/me/ Users: Get-Current-User	v 🔒
profiles		^
GET	/api/profiles/{user_id}/ Profiles: Get-Profile-By-Id	v
PUT	/api/profiles/me/ Profiles: Update-Own-Profile	v 🔒
admin		^
GET	/api/admin/ Users: Get-All-Users	v 🔒
PUT	/api/admin/activate_not/{id} Users: Block-Activate-User	v 🔒
PUT	/api/admin/superuser/{id} Users: Superuser	v 🔒

31. Open APIs for authentication microservice

The columns `is_active`, `is_superuser`, `is_master` depict the user's type of authorization. In the authentication system there are three levels of authorization. Users of the web app may have one of the three roles: `simple_user`, insurance company, admin.

`Simple_user` is the client that has a vehicle accident and wants to make a statement. They should have `is_active` column equal to true. Insurance company is the company that collects the accidents statements from the client. They should have `is_superuser` column equal to true.

Admin is the controller of the flow of the app that intervenes when someone of the previous roles causes problems. They should have is_master column equal to true.

Admin is able to make an account inactive:

```
@router.put("/activate_not/{id}", response_model=UserPublic, name="users:block-activate-user")
async def block_activate_user(id:int,
    current_user: UserInDB = Depends(get_current_active_user),
    users_repo: UsersRepository = Depends(get_repository(UsersRepository))),
)-> UserPublic:
    if current_user.is_master:
        return await users_repo.block_unblock_user(id= id)
    else:
        raise HTTPException(status_code=HTTP_401_UNAUTHORIZED, detail="No access")
```

Admin can set a user as superuser:

```
@router.put("/superuser/{id}", response_model=UserPublic, name="users:superuser")
async def superuser(id:int,
    current_user: UserInDB = Depends(get_current_active_user),
    users_repo: UsersRepository = Depends(get_repository(UsersRepository))),
)-> UserPublic:
    if current_user.is_master:
        return await users_repo.superuser(id= id)
    else:
        raise HTTPException(status_code=HTTP_401_UNAUTHORIZED, detail="No access")
```

At profiles table, there are personal data provided by each user. Column user_id, as it is depicted in the schema, is a foreign key that refers to the users table, at id column. UserPublic model as stated in user models includes the profile data of the user, so the response in get-current-user is like the below:

```
{
  "email": "nikitaskastis@gmail.com",
  "username": "Nikitas",
  "email_verified": false,
  "is_active": true,
  "is_superuser": false,
  "is_master": false,
  "created_at": "2021-08-09T14:22:58.559302+00:00",
  "updated_at": "2021-08-09T14:22:58.559302+00:00",
  "id": 2,
  "access_token": null,
  "profile": {
    "first_name": "NIKITAS",
    "last_name": "KASTIS",
    "phone_number": "697-999-7777",
    "licence_number": "ME1905",
    "licence_category": "C",
    "licence_expire_date": "2040-12-31",
    "image": null,
  }
}
```

```

    "created_at": "2021-08-09T14:22:58.566879+00:00",
    "updated_at": "2021-08-09T14:42:29.041204+00:00",
    "id": 2,
    "user_id": 2,
  }
}

```

5.4 ACCIDENT STATEMENT MICROSERVICE

Accident statement microservice holds the data of the accident statements. When a user has a valid insurance that is one of the co-operative insurance companies, they can declare an accident with their accident statement. They can include images and a sketch of the accident.

At most requests, the app checks the current user to give the appropriate response. As described above, the current user's check takes place in the previous microservice. So, this microservice is dependent on the previous one. If there is no current user, response is: "Not authenticated".

```

oauth2_scheme = OAuth2PasswordBearer(tokenUrl=f"{API_PREFIX}/users/login/token/")
async def get_user_from_token(
    *,
    token: str = Depends(oauth2_scheme),
    user_repo: UsersRepository = Depends(get_repository(UsersRepository)),
) -> Optional[UserInDB]:
    try:
        username = auth_service.get_username_from_token(token=token,
secret_key=str(SECRET_KEY))
        user = await user_repo.get_user_by_username(username=username)
    except Exception as e:
        raise e
    return user

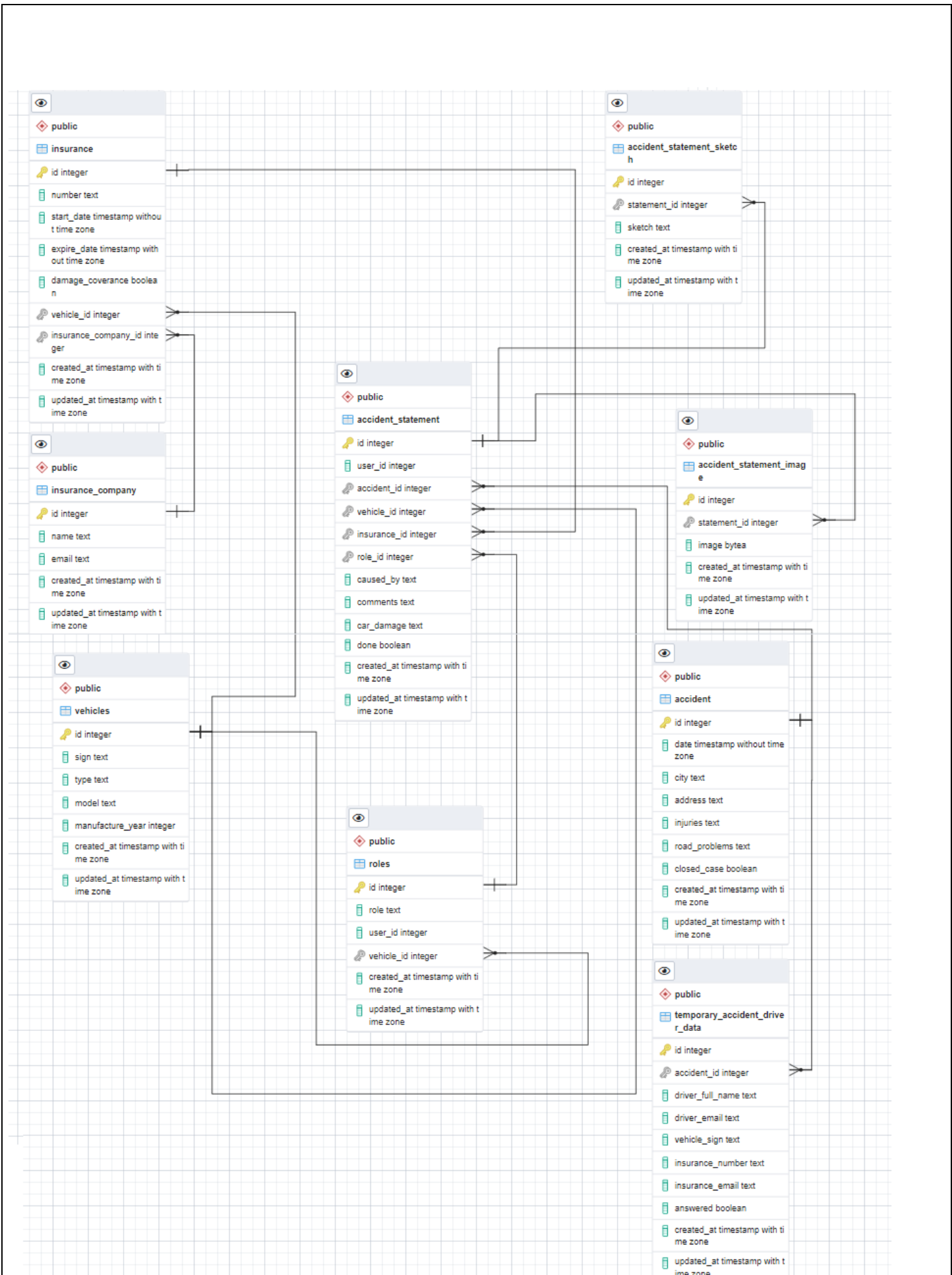
def get_current_active_user(current_user: UserInDB = Depends(get_user_from_token))
-> Optional[UserInDB]:
    if not current_user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="No authenticated user.",
            headers={"WWW-Authenticate": "Bearer"},
        )
    if not current_user.is_active:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Not an active user.",
            headers={"WWW-Authenticate": "Bearer"},
        )
    return current_user.

```

Entity Relationship Diagram is remarkably more complicated in Accident Statement services. Database includes 9 tables: vehicles, insurance, insurance_company, roles, accident, accident_statement, accident_statement_sketch, accident_statement_image and temporary_accident_driver_data.

The insurance company table includes all the insurance companies that the user can get a vehicle insurance. Insurance companies can only be added by admin.

```
@router.post("/", response_model=InsuranceCompanyPublic, name="insurance-
company:create-insurance-company", status_code=HTTP_201_CREATED)
async def create_new_insurance_company(
    current_user: UserPublic = Depends(get_current_active_user),
    new_insurance_company: InsuranceCompanyCreate = Body(..., embed=True),
    insurance_company_repo: InsuranceCompanyRepository =
    Depends(get_repository(InsuranceCompanyRepository)),
) -> InsuranceCompanyPublic:
    if current_user.is_master:
        created_insurance_company = await
insurance_company_repo.create_insurance_company(new_insurance_company=new
w_insurance_company)
    else:
        raise HTTPException(status_code=HTTP_401_UNAUTHORIZED, detail="No access")
    return created_insurance_company
```

32. Entity Relationship Diagram for vehicle database

In the vehicles table, there are information about the vehicles of each user, while insurance and roles are tables that populate the vehicle with info about the signed vehicle insurance and if they are the owner or a user of the vehicle. Insurance and role are different tables, because they keep track of every change. So each vehicle has many insurances and each user may have many roles in the vehicle over the years.

Accident table has info about the place and the time each accident happened and if there were injuries or road problems. It is populated with data from the accident statement table and data of the rest drivers, who are declared as participants in the accident. In the accident statement there is a column car_damage that saves the prediction from the other microservice, if it is used. Accident_statement_images table includes photos from the accident that users have provided and accident_statement_sketch has a list of coordinates of the sketch of the user saved as string.

As the open api page shows, back-end calls, which suggest the user add or update data, are based on the routes of the vehicles and the accidents.

Vehicle APIs

The screenshot shows the OpenAPI specification for 'herewearevehicles' (version 1.0.0, OAS3). It lists several endpoints:

- GET** /api/vehicles/all Vehicles: Get-All-Vehicles
- GET** /api/vehicles/ Vehicles: Get-All-Vehicles-By-User-Id
- POST** /api/vehicles/ Vehicles: Create-Vehicle
- GET** /api/vehicles/no/{id}/ Vehicles: Get-Vehicle-By-Id
- PUT** /api/vehicles/{vehicle_id}/insurance Insurance: Update-Insurance-By-Vehicle-Id
- POST** /api/vehicles/{vehicle_id}/insurance Insurance: Create-Insurance-By-Vehicle-Id
- PUT** /api/vehicles/{vehicle_id}/role Role: Update-Role-By-Vehicle-Id

33. Open APIs for vehicles in vehicle microservice

At the api call where the user requests to get all their vehicles, each vehicle is populated by the last insurance added. Additionally, insurance cannot be updated, if the last added insurance has not expired. If the user represents an insurance company (is_superuser), they can get all the vehicles that have last insurance in the same insurance company.

The screenshot shows an API error response with a status code of 400 (Undocumented) and the message "Error: Bad Request". The response body is:

```
{
  "detail": "Wait current insurance expire"
}
```

34. Insurance can not be updated if the last insurance has not expired

The vehicle sign column in the vehicle table should have unique entries. As someone can drive or owe the vehicle, there are different roles, as they are stated in the role table. If a different user add the same vehicle, the app updates the information that there is a new driver for the vehicle, but no new vehicle is added to the database.

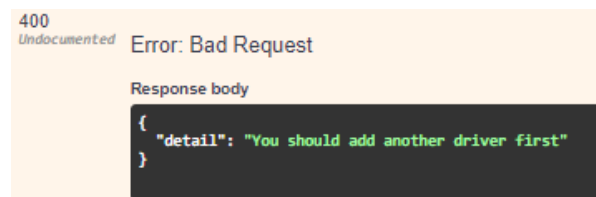
Accidents APIS

accident		^
POST	/api/accident/{vehicle_id}/	Accident.Create-Accident
GET	/api/accident/{id}/	Accident.Get-Accident-By-Id
POST	/api/accident/temporary/{accident_id}/	Accident.Add-Drivers-Accident-Id
GET	/api/accident/	Accident.Get-Accidents-By-User-Id
PUT	/api/accident/accident_stmt/{accident_id}	Accident.Edit-Accident-Stmt
POST	/api/accident/accident_stmt/{accident_id}	Accident.Add-Accident-Stmt
PUT	/api/accident/accident_stmt_detection/{accident_id}	Accident.Add-Damaged-Part-Detected
POST	/api/accident/accident_sketch/add/{accident_id}	Accident.Add-Sketch
POST	/api/accident/removetemporary/{id}	Accident.Remove-Driver
POST	/api/accident/newImage	Accident.Add-Accident-Image
GET	/api/accident/image/{id}	Accident.Get-Accident-Image
GET	/api/accident/imageList/{accident_id}	Accident.Get-Accident-Image-List
PUT	/api/accident/complete_statement/{accident_id}	Accident.Statement-Complete-Statement

35. Open APIs for accidents in vehicle microservice

Access to see and edit accident data is given only to the drivers that are added to the accident, while their respective insurance companies have only access to see the data.

Accident_statement has a column with the boolean variable "done". User can set an accident statement as "done=True", as long as they have filled all the necessary fields, they have added sketch, photos and the other driver of the accident.



36. Accident statement cannot be completed when...

When this is saved as True, neither changes can be made on the accident statement and the accident_statement_sketch nor more accident_statement photos can be added.

Only the driver that declares the accident can add the rest drivers that are involved, while they cannot add themselves independently. Moreover, the driver that declares the accident is the only one that can remove a driver from the accident as long as has not updated their statement as "done" = True or that driver has not made a statement.

```
404
Undocumented Error: Not Found

Response body
{
  "detail": "You can not add drivers"
}
```

37. When statement is stated as done, no changes can be made

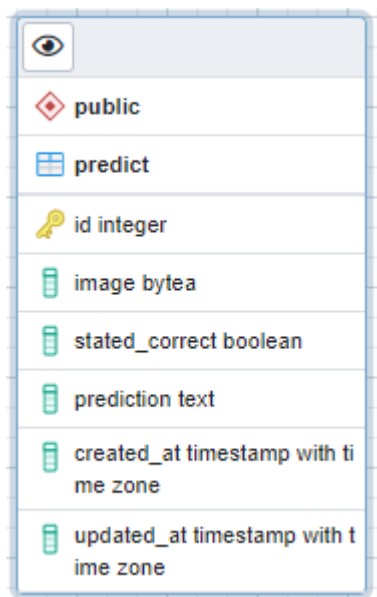
A driver cannot make an accident statement if they have not updated a valid insurance for the vehicle at the time of the accident.

If a driver has made an accident statement, they cannot add another statement, but only update the current one.

Adding image in a FastAPI call cannot be achieved through a json body request. As a multipart/ form-data call, it will get an image as a file and save it in the database as binary string.

5.5 VEHICLE DAMAGE DETECTION MICROSERVICE

Database volume is included, as damaged vehicle photos are not too many, and the correct predictions will be saved to improve the future reliability of the detection model. It consists of only one table "prediction" that has the following columns: id, image, prediction, stated_correct and the timestamps.



Schema	Table	Column	Type	Constraints
public	prediction	id	integer	Primary Key
public	prediction	image	bytea	
public	prediction	stated_correct	boolean	
public	prediction	prediction	text	
public	prediction	created_at	timestamp with time zone	
public	prediction	updated_at	timestamp with time zone	

38. One table for damage detection database

As it is depicted in the api page of the microservices, no authentication is needed. Everyone can add an image to check damage. Predictions that are not correct are deleted from the db. The boolean variable stated_correct is used to protect the correct predictions from the prediction. When it has "True" value, the prediction cannot be deleted.

predict	
POST	/api/predict/predict New Predict
POST	/api/predict/remove/{id} Predict.Remove-Prediction
GET	/api/predict/{id}/ Predict.Get-Prediction-By-Id
PUT	/api/predict/stated_correct/{id} Predict.State-Prediction-Correct-By-Id

39. Open APIs for damage detection microservice

In vehicle damage detection, user adds an image to get an answer about the damage that the app detected. This answer comes from the before-mentioned selected model as it was described in the previous chapter. Due to the small number of damaged vehicles, the app saves the photos to generate more efficient models in the future.

```
@router.post("/predict")
async def new_predict(
    file: UploadFile = File(...),
    prediction_repo: PredictRepository = Depends(get_repository(PredictRepository)),
):
    extension = file.filename.split(".")[-1] in ("jpg", "jpeg", "png", "JPG", "JPEG", "PNG")
    if not extension:
        raise HTTPException(status_code=HTTP_400_BAD_REQUEST, detail="Image must be
        jpg or png format!")
    image_wanted = read_imagefile(await file.read())
    image_check = image_wanted.resize(car_image_shape[0:2])
    check_array = image.img_to_array(image_check)/255
    check_array = np.expand_dims(check_array, axis=0)
    result1 = is_damaged_model.predict(check_array)
    print(result1)
    if result1.argmax() != 0:
        return "Sorry, I cannot see any vehicle damage. Please, try again"
    else:
        part = car_part_model.predict(check_array)
        if part.argmax() == 0:
            a = 'damage on the front part'
        elif part.argmax() == 1:
            a = 'damage on the rear'
        elif part.argmax() == 2:
            a = 'damage on the side'
        elif part.argmax() == 3:
            a = 'damage on the bumper'
        elif part.argmax() == 4:
            a = 'damage on the door'
        elif part.argmax() == 5:
            a = 'damage on the glass'
        else:
            a = 'damage on the lamp'
    contentsImage = await file.read()
    new_predict = Predict_Create(image = contentsImage, prediction= a)
```

```
prediction = await prediction_repo.predict(predict = new_predict)
return prediction
```

First model check that there is a car, and that the car has some damage. In case the first model does not detect a car, the user gets the message "Sorry, I cannot see any vehicle damage. Please, try again" and the app returns.

Otherwise, if damage gets detected, the image goes to the second model that predicts if the damage is on the door, bumper, glass, head light, taillight or, if it's a bigger damage in the car on the front, rear or side part.

5.6 FRONT END: REACT @ REDUX

Creating an app that is full of forms and data transfer, a predictable state container like Redux is needed, that centralizes the application state and enables a consistent response to users' requests.

5.6.1 How Redux works?

Initial state makes important variables reachable in every React component. All actions are created to make changes to the state, most of them making requests - api calls to the backend. The response of the request has two results, either success or failure, and depending to the result, each Redux reducer updates initialstate.

Initial state of app:

```
export default {
  auth: {
    isLoading: false,
    error: false,
    user: {},
    allUsers: {},
  },
  vehicles: {
    isLoading: false,
    error: null,
    data: {},
    currentVehicle: null,
    accidentData: {},
    currentAccident: null,
    insuranceCompanyData: {},
  }
}
```

Example of Reducer that updates the state according to the success or not of the request. So, when the app gets a request to fetch user data from token isLoading gets value 'true' and the loading sign appears on the user interface.

If the request is successful, user gets updated. Otherwise, the error message gets updated.

```
export default function authReducer(state = initialState.auth, action = {}) {
  switch (action.type) {
    case FETCHING_USER_FROM_TOKEN:
      return {
        ...state,
        isLoading: true
      }
    case FETCHING_USER_FROM_TOKEN_SUCCESS:
      return {
        ...state,
        isLoading: false,
        user: action.data
      }
    case FETCHING_USER_FROM_TOKEN_FAILURE:
      return {
        ...state,
        isLoading: false,
        error: action.error,
        user: {}
      }
  }
}
```

Due to many api calls, additional functions were created to make coding for these requests faster and simpler. Following function was used to make a Post api call:

```
const apiPost = ({ url, params, types: {REQUEST, SUCCESS, FAIL},
  onSuccess = (response) => ({ type: response.type, status: response.status })
}) => {
  return async (dispatch) => {
    const token = await localStorage.getItem('access_token');
    const fullUrl = baseUrl.concat(url);
    const defaultConfig = {
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
        Authorization: token ? `Bearer ${token} : `,
      },
    };
    try {
      const response = await axios.post(fullUrl, params, defaultConfig, );
      dispatch({
        type: SUCCESS,
        status: response.status,
        payload: response.data,
      });
      return onSuccess({ type: SUCCESS, ...response });
    } catch (error) {
      const errorResponse = ({ response }) => ({ response }))( error, );
      dispatch({ type: FAILURE,
        status: errorResponse?.response?.status,
      });
    }
  }
}
```

```

        payload: errorResponse?.response?.data?.error, });
    });
};
export default apiPost;

```

Action using the function above is:

```

Actions.registerNewUser = ({ username, email, password }) => {
const params = new URLSearchParams();
  params.append('email', email);
  params.append('username', username);
  params.append('password', password);
return (dispatch) =>
  dispatch(
    apiPost({
      url: `/users/`,
      params,
      types: {
        REQUEST: REQUEST_USER_SIGN_UP,
        SUCCESS: REQUEST_USER_SIGN_UP_SUCCESS,
        FAILURE: REQUEST_USER_SIGN_UP_FAILURE
      },
      onSuccess: (res) => {
        const access_token = res?.data?.access_token?.access_token
        localStorage.setItem("access_token", access_token)
        return dispatch(Actions.fetchUserFromToken(access_token))
      },
    })
  )
}

```

5.7 KUBERNETES

Having installed minikube, we create a Deployment yaml file.

For example, in the following deployment object for 'hereweare' (here we are 1st container):

- ✓ kind of resource in this file is 'deployment'
- ✓ name declared for the deployment is 'hereweare-deployment', in the cluster of 'here-we-are,
- ✓ deployed in 2 replicas/instances of the pod of the given container,
- ✓ its set up template has metadata label app: 'hereweare' and is matched in the selector
- ✓ and the specifications for the containers that will be run inside the Pods in this deployment

```

---
apiVersion: apps/v1
kind: Deployment

```



```

metadata:
  name: hereweare-deployment
  namespace: here-we-are
  labels:
    app: hereweare
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hereweare
  template:
    metadata:
      labels:
        app: hereweare
    spec:
      containers:
        - name: hereweare
          image: hereweare:dev
          imagePullPolicy: Never
          command: ["uvicorn"]
          args:
            [
              "app.api.server:app",
              "--host",
              "0.0.0.0",
              "--port",
              "8000",
              "--workers",
              "2",
            ]
---

```

Next step is Service yaml file that specifies the name of the port, port number, target port and the type of service. In the same example, port name is web, port selected is 8000 and selected port 8000. Type selected is LoadBalancer, aiming to create an outside available IP address and automatically distribute incoming requests across all pods no matter of which a pod runs.

```

apiVersion: v1
kind: Service
metadata:
  name: hereweare-app
  namespace: here-we-are
  labels:
    app: hereweare
spec:
  type: LoadBalancer
  ports:
    - name: web
      port: 8000
      targetPort: 8000
  selector:

```

app: hereweare

To have greater independence of the pods for the database containers, persistent volumes will be created.

Persistent Volume object:

- ✓ Like before kind of resource is declared for the file: Persistent Volume.
- ✓ Name added in metadata for the persistent volume is *hereweare-postgres-pv-volume* and gets labels for type: 'local' and app: 'hereweare-postgres'
- ✓ Storage capacity for the persistent volume is 1 Gigabyte.
- ✓ A Persistent Volume Claim requesting the same storageClass can be bound to this volume.
- ✓ Host path is used for development and testing. Persistent Volume uses a file/directory on the Node to emulate network-attached storage.
- ✓ PersistentVolumeReclaimPolicy Retain means that the PV even after PVC is deleted.
- ✓ ReadWriteMany means that the volume can be mounted as read-write by many Nodes

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: hereweare-postgres-pv-volume
  labels:
    type: local
    app: hereweare-postgres
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data/auth/"
  persistentVolumeReclaimPolicy: Retain
---
```

Persistent Volume Claim object:

- ✓ Kind of resource is declared for the file: Persistent Volume Claim. It requests a Persistent Volume Storage.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: hereweare-postgres-pv-claim
  labels:
    app: hereweare-postgres
spec: # Access mode and resource limits
```

```
storageClassName: manual
accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
---
```

Hereweare-postgres-secret, referred below, is another object that includes the environmental variables from a secret file whose name is "postgres-secret"

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hereweare-postgres
  labels:
    app: hereweare-postgres
spec:
  selector:
    matchLabels:
      app: hereweare-postgres
      tier: postgres
  replicas: 1
  template:
    metadata:
      labels:
        app: hereweare-postgres
        tier: postgres
    spec:
      containers:
        - image: postgres:12.1-alpine
          name: postgres
          imagePullPolicy: "IfNotPresent"
          envFrom:
            - secretRef:
                name: hereweare-postgres-secret
          ports:
            - containerPort: 5432
              name: postgres
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: hereweare-postgres-persistent-storage
      volumes:
        - name: hereweare-postgres-persistent-storage
          persistentVolumeClaim:
            claimName: hereweare-postgres-pv-claim
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: hereweare-postgres
```

```
labels:
  app: hereweare-postgres
spec:
  type: LoadBalancer
ports:
  - port: 5432
    targetPort: 5432
    protocol: TCP
selector:
  app: hereweare-postgres
  tier: postgres
```

Makefile is used again:

```
create-local-cluster:
  minikube start
  kubectl create ns here-we-are
  kubectl apply -n here-we-are -f .\hereweare\k8s\hereweare-postgres.yaml
  kubectl apply -n here-we-are -f .\herewearevehicles\k8s\herewearevehicles-
  postgres.yaml
  kubectl apply -n here-we-are -f .\herewearepredict\k8s\detect-
  postgres.yaml
```

```
images:
  cd hereweare & make build-img
  cd herewearevehicles & make build-img
  cd herewearepredict & make build-img
  cd herewearefrontend & make build-img
```

```
deploy: images
  kubectl apply -n here-we-are -f .\hereweare\k8s\hereweare-api.yaml
  kubectl apply -n here-we-are -f .\herewearevehicles\k8s\herewearevehicles-
  api.yaml
  kubectl apply -n here-we-are -f .\herewearepredict\k8s\detect-api.yaml
  kubectl apply -n here-we-are -f .\herewearefrontend\k8s\front.yaml
```

```
migrate:
  kubectl -n here-we-are exec deploy/hereweare-deployment -- alembic
  upgrade head
  kubectl -n here-we-are exec deploy/herewearevehicles-deployment --
  alembic upgrade head
  kubectl -n here-we-are exec deploy/detect-deployment -- alembic upgrade
  head
```

```
delete-local-cluster:
  minikube stop
  minikube delete
```

- ✓ Make create-local-cluster starts minikube, creates the cluster "here-we-are" and the three persistent volumes for the databases.

```

Windows PowerShell
Creating docker container (CPUs=2, Memory=3000MB) ...
Preparing Kubernetes v1.20.2 on Docker 20.10.5 ...
  Generating certificates and keys ...
  Booting up control plane ...
  Configuring RBAC rules ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
kubectl create ns here-we-are
namespace/here-we-are created
kubectl apply -n here-we-are -f .\hereweare\k8s\hereweare-postgres.yaml
secret/hereweare-postgres-secret created
persistentvolume/hereweare-postgres-pv-volume created
persistentvolumeclaim/hereweare-postgres-pv-claim created
deployment.apps/hereweare-postgres created
service/hereweare-postgres created
kubectl apply -n here-we-are -f .\herewearevehicles\k8s\herewearevehicles-postgres.yaml
secret/herewearevehicles-postgres-secret created

```

40. Starting minikube and creating persistent volumes

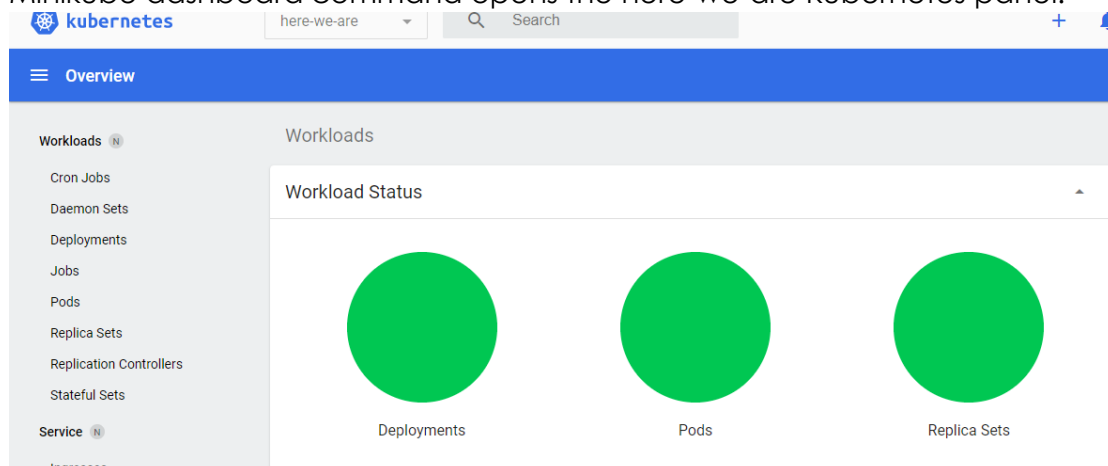
- ✓ Then, `minikube -p minikube docker-env | Invoke-Expression`, to point docker-machine to minikube's docker environment.
- ✓ Make deploy builds the images and recursively creates those Kubernetes objects.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
detect-postgres	LoadBalancer	10.97.241.77	<pending>	5434:31522/TCP	32m
hereweare-app	LoadBalancer	10.108.231.147	<pending>	8000:31896/TCP	22s
hereweare-postgres	LoadBalancer	10.107.119.143	<pending>	5432:32561/TCP	32m
herewearefront-app	LoadBalancer	10.104.23.84	<pending>	3000:31250/TCP	5m18s
herewearepredict-app	LoadBalancer	10.97.200.84	<pending>	8002:30959/TCP	5m19s
herewearevehicles-app	LoadBalancer	10.110.14.229	<pending>	8001:31326/TCP	5m20s
herewearevehicles-postgres	LoadBalancer	10.107.188.143	<pending>	5433:31279/TCP	32m

41. Created services in Minikube

- ✓ Make migrate makes the migrations to the databases.

Minikube dashboard command opens the here-we-are Kubernetes panel.



42. Minikube dashboard

Here we can see deployments, pods, logs for every pod, replica-sets, etc.

Deployments

Name	Labels	Pods	Created ↑	Images
✓ hereweare-deployment	app: hereweare	2 / 2	3 minutes ago	hereweare:dev
✓ front-deployment	app: herewearefront	1 / 1	8 minutes ago	soherewearefrontend:lates
✓ detect-deployment	app: herewearepredict	2 / 2	8 minutes ago	herewearepredict:dev
✓ herewearevehicles-deployment	app: herewearevehicles	2 / 2	8 minutes ago	herewearevehicles:dev
✓ detect-postgres	app: detect-postgres	1 / 1	35 minutes ago	postgres:12.1-alpine
✓ herewearevehicles-postgres	app: herewearevehicles-postgres	1 / 1	35 minutes ago	postgres:12.1-alpine
✓ hereweare-postgres	app: hereweare-postgres	1 / 1	35 minutes ago	postgres:12.1-alpine

43. Minikube deployments for here-we-are namespace

Here we see all deployments working with all pods, while front deployment had to restart 2 times.

✓ hereweare-deployment-c89b7b775-dm48d	app: hereweare pod-template-hash: c89b7b775	minikube	Running	0	-	-	5 minutes ago	⋮
✓ hereweare-deployment-c89b7b775-sn87w	app: hereweare pod-template-hash: c89b7b775	minikube	Running	0	-	-	5 minutes ago	⋮
✓ front-deployment-5c67f867b4-kkqlm	app: herewearefront pod-template-hash: 5c67f867b4	minikube	Running	2	-	-	10 minutes ago	⋮
✓ detect-deployment-55cff98fdd-5wkcc	app: herewearepredict pod-template-hash: 55cff98fdd	minikube	Running	0	-	-	10 minutes ago	⋮
✓ detect-deployment-55cff98fdd-vhl6n	app: herewearepredict pod-template-hash: 55cff98fdd	minikube	Running	0	-	-	10 minutes ago	⋮

44. Front deployment had 2 restarts

With minikube tunnel command, we can see the app in the browser, and check logs for each pod.

Pods > hereweare-deployment-c89b7b775-dm48d > Logs

Logs from hereweare in hereweare-deployment-c89b7b775-dm48d

```
INFO: 172.17.0.1:23688 - GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:35139 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:19565 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:33034 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:9999 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:17889 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:23710 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:30454 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:43775 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:20571 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:50842 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:46027 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:18988 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:8564 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:46313 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:15105 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:58795 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:5712 - "GET /api/users/me/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:37748 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:40311 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:50298 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:55336 - "GET /api/profiles/1/ HTTP/1.1" 200 OK
```

Logs from Sep 21, 2021 to Sep 21, 2021 UTC

45. Logs for hereweare deployment

Pods > detect-deployment-55cff98fdd-5wkcc > Logs

Logs from herewearepredict in detect-deployment-55cff98fdd-5wkcc

```
INFO: Started server process [9]
INFO: Waiting for application startup.
INFO: Connected to database postgres://postgres:*****@detect-postgres:5434/postgres
INFO: Application startup complete.
2021-09-20 23:12:48.985774: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2021-09-20 23:12:49.012109: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2400000000 Hz
[[0.95553976 0.03959123 0.004869 ]]
INFO: 172.17.0.1:12711 - "POST /api/predict/predict HTTP/1.1" 200 OK
INFO: 172.17.0.1:12711 - "OPTIONS /api/predict/stated_correct/1 HTTP/1.1" 200 OK
INFO: 172.17.0.1:12711 - "PUT /api/predict/stated_correct/1 HTTP/1.1" 200 OK
```

46. Logs for detect deployment

Pods > herewearevehicles-deployment-5b5b69c75-lr9j4 > Logs

Logs from herewearevehicles in herewearevehicles-deployment-5b5b69c75-lr9j4

```
INFO: Waiting for application startup.
INFO: Connected to database postgres://postgres:*****@herewearevehicles-postgres:5433/postgres
INFO: Application startup complete.
INFO: Connected to database postgres://postgres:*****@herewearevehicles-postgres:5433/postgres
INFO: Application startup complete.
INFO: 172.17.0.1:3100 - "OPTIONS /api/vehicles/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:58081 - "OPTIONS /api/insurance_company/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:3100 - "GET /api/insurance_company/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:58081 - "GET /api/vehicles/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:9875 - "GET /api/insurance_company/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:9069 - "GET /api/accident/ HTTP/1.1" 404 Not Found
INFO: 172.17.0.1:32614 - "GET /api/vehicles/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:39544 - "POST /api/vehicles/ HTTP/1.1" 201 Created
INFO: 172.17.0.1:39544 - "OPTIONS /api/vehicles/no/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:39544 - "GET /api/vehicles/no/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:62784 - "POST /api/vehicles/1/role/ HTTP/1.1" 307 Temporary Redirect
INFO: 172.17.0.1:62784 - "POST /api/vehicles/1/role HTTP/1.1" 200 OK
INFO: 172.17.0.1:62784 - "GET /api/vehicles/no/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:62784 - "GET /api/vehicles/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:4614 - "OPTIONS /api/accident/1/ HTTP/1.1" 200 OK
INFO: 172.17.0.1:4614 - "POST /api/accident/1/ HTTP/1.1" 201 Created
```

47. Logs for herewearevehicles deployment

6. CONCLUSIONS

6.1 PROBLEMS CREATING THE APP

Creating the app, many problems appeared:

- ✓ The third role of app users "master" is created mainly to take charge of activating - deactivating a user's account or not. Insurance companies should not have the right to do that, since a user may have vehicles insurance in different insurance companies.
- ✓ Generating predicting models with layers with big amounts of filters and neurons end up in .h5 files with 1GB sizes. This made container run slow and use more sources. To avoid this, lighter models were selected with the same results.
- ✓ Front-end had the nav option for the master to visit page "View all accidents". But when there were many accidents declared, the response was too slow and quite expensive. So, the page "View accidents" was deprecated, and master could request to view accidents through the user page, in the tab view user accident statements.
- ✓ Few images of damaged cars were available in creative commons, and no dataset that included prices for repairing damages. Insurance companies could not provide any data. So, detect damage system was limited to locate the accident in the car.

6.2 FUTURE WORKS ON 'HERE WE ARE'

In 'HereWeAre', when more data of the third microservice will be gathered, better models can be created. One step further, insurance companies can provide data of damage cost, so the model can predict monetary values of repairing a car damage.

Furthermore, 'Here We Are' is an application that can have more microservices added, such as the following:

- ✓ A "statistics" microservice that can be used by insurance companies to adjust their pricing policy. This microservice may provide information about the cities that has more vehicle accidents, the most common reasons of an accident. This tool is also useful for the respective authorities to check the road regulations and maybe adjust them in roads that have high number of accidents.
- ✓ A microservice that lets users know where roads have traffic due to vehicle accidents.

Additionally, 'HereWeAre' can add a chat service, using Socket.io, if drivers and insurance companies want to communicate via the website instantly.

Sockets can be used to update live the information to all users without the need of refreshing the page.

Finally, although 'Here We Are' is a mobile friendly web application, one of the next steps include the development of a mobile front app in React Native. This will permit push notifications, maps, etc.

7. REFERENCES

- [1] OECD (2020), Digital Transformation in the Age of COVID-19: Building Resilience and Bridging Divides, Digital Economy Outlook 2020 Supplement, OECD, Paris, www.oecd.org/digital/digital-economy-outlook-covid.pdf
- [2] Altoros (2019), Using machine learning to automate car damage assessment and document workflows, Intel® AI Builders Showcase, Grand Ballroom West, Vladimir Starostenkov, Siarhei Sukhadolski
- [3] IBM Cloud Education (2021), What are containers? Accessed 13 August 2021, www.ibm.com/cloud/learn/containers
- [4] NetApp, What are containers?, Accessed 14 August 2021, www.netapp.com/devops-solutions/what-are-containers/
- [5] Section (2019), Bora Basyildiz, A Brief History of Container Technology, Accessed 13 August 2021, www.section.io/engineering-education/history-of-container-technology/
- [6] A Cloud Guru (2018), Eli Marquez, The History of Container Technology, Accessed 14 August 2021, Pluralsight, acloudguru.com/blog/engineering/history-of-container-technology
- [7] Docker (2014), Docker 0.9: introducing execution drivers and libcontainer, Accessed 14 August 2021, Docker, www.docker.com/blog/docker-0-9-introducing-execution-drivers-and-libcontainer/
- [8] Docker Blog (2018), Solomon Hykes, Au Revoir, Accessed 16 August 2021, Docker, www.docker.com/blog/author/solomon-hykes/
- [9] GitHub-Gist(2016), Arnaud Porterie, "Docker – Updated project statistics, Accessed 16 August 2021, Github, gist.github.com/icecrime/18d72202f4569a0cab1ee60f7583425f
- [10] Docker Docs (2020), Docker Desktop WSL 2 backend, Accessed 16 August 2021, Docker, docs.docker.com/docker-for-windows/wsl
- [11] Microsoft | Docs (2021), Docker terminology, Accessed 16 August 2021, Microsoft, docs.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-terminology
- [12] Newman S. (2015), Building Microservices, 1st edition, USA, O'Reilly Media, Inc.
- [13] Dataversity (2021), Keith D Foote, A Brief History of Microservices, Accessed 17 August 2021, Dataversity, www.dataversity.net/a-brief-history-of-microservices/

- [14] Löwy J. (2007) Programming WCF Services, 1st edition, USA, O'Reilly Media, Inc.
- [15] James Lewis, MartinFowler(2014), Microservices, a definition of this new architectural term, Accessed 17 August 2021, martinfowler.com/articles/microservices.html
- [16] Verified Market Research (2021), Cloud Microservices Market 2020 Trends, Market Share, Industry Size, Opportunities, Analysis and Forecast by 2026, Accessed 17 August 2021, Market Reports, www.instanttechnews.com/technology-news/2021/02/23/cloud-microservices-market-2020-trends-market-share-industry-size-opportunities-analysis-and-forecast-by-2026/
- [17] Microsoft | Docs (2019), Microsoft Architecture Style, Accessed 17 August 2021, Microsoft, docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
- [18] Kong (2021), Microservice Container Orchestration Tools, Accessed 25 August 2021, Kong Inc. – KongHQ, konghq.com/learning-center/microservices/microservices-orchestration/
- [19] The Kubernetes Authors, Kubernetes, Accessed 25 August 2021, The Linux Foundation, kubernetes.io/
- [20] Gordon Haff, William Henry (2017), From Pots and Vats to Programs and Apps, Red Hat, s3.amazonaws.com/grhpublic/packaging_book_FINAL_ebook.pdf
- [21] Frederic Lardinois (2015), As Kubernetes Hits 1.0, Google Donates Technology to Newly Formed Cloud Native Computing Foundation, Accessed 28 August 2021, TechCrunch, techcrunch.com/2015/07/21/as-kubernetes-hits-1-0-google-donates-technology-to-newly-formed-cloud-native-computing-foundation-with-ibm-intel-twitter-and-others/
- [22] Jacek Chmiel, (2020), Kubernetes – how hot can it get? Accessed 28 August 2021, Avenga, www.avenga.com/magazine/kubernetes/
- [23] Laurianne MacLaughlin (2020), Kubernetes Glossary for Executives, Accessed 28 August 2021, The Enterprisers Project, enterpriseproject.com/sites/default/files/kubernetes_glossary.pdf
- [24] Maximilian Schwarzmüller (2020), Docker & Kubernetes: The Practical Guide, udemy.com/course/docker-kubernetes-the-practical-guide/learn/lecture/22627559#overview
- [25] Business Browdaway(2020), Bob Hayes, Usage of Programming by Data Scientists: Python Grows while R Weakens, Accessed 14 August 2021, businessoverbroadway.com/2020/06/29/usage-of-programming-languages-by-data-scientists-python-grows-while-r-weakens/
- [26] Statista (2021), Shanhong Liu, Most used programming languages among developers worldwide, as of 2021, Accessed 14 August 2021,

www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

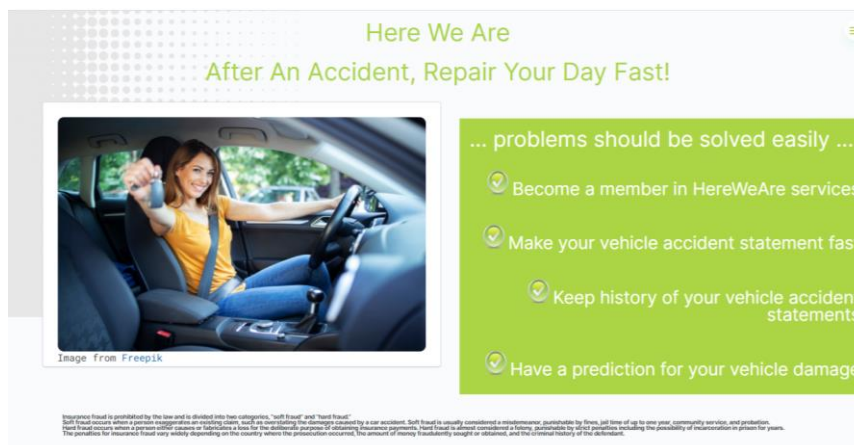
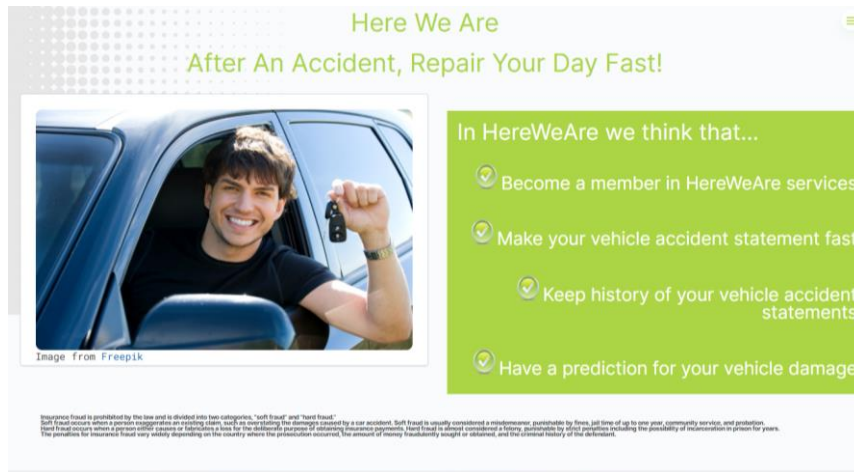
- [27] BoTreeTechnologies(2020), Parth Barrot, Pros and Cons of PythonL A Definite Python Web Development Guide, Accessed 14 August 2021, Tntra, www.botreetechnologies.com/blog/pros-and-cons-of-python/
- [28] Artima (2003), Bill Venners, The Making of Python A Conversation with Guido van Rossum, Part I, Accessed 4 September 2021, www.artima.com/articles/the-making-of-python
- [29] Exyte (2020), Vasilisa Sheromova, A brief history of Python, Accessed 4 September 2021, exyte.com/blog/a-brief-history-of-python
- [30] The Register(2018), Simon Sharwood, Python creator Guido van Rossum sys.exit()s as language overlord, Accessed 4 September 2021, www.theregister.com/2018/07/13/python_creator_guido_van_rossum_quits
- [31] The Netflix Tech Blog(2020), Kevin Glisson, Marc Vilanova, Forest Mosen, Introducing Dispatch, Accessed 4 September 2021, Netflix, netflixtechblog.com/introducing-dispatch-da4b8a2a8072
- [32] Uber Engineering(2019), Piero Molino, Yaroslav Dudin, Sai Sumanth Miryala, Ludwig v0.2 Adds New Features and Other Improvements to its Deep Learning Toolbox Accessed 4 September 2021, Uber, eng.uber.com/ludwig-v0-2/
- [33] Educative (2021), Erin Schaffer, Python REST API tutorial: Getting started with FastAPI, Accessed 4 September 2021, www.educative.io/blog/python-fastapi-tutorial
- [34] Stack Overflow (2021), 2021 Developer Survey, Accessed 4 September 2021, insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021
- [35] Exastax (2017), Top 5 Use Cases of Tensorflow, Accessed 4 September 2021, www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/
- [36] Usenix (2016), Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: A system for large-scale machine learning, Accessed 5 September 2021, Google Brain, www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
- [37] Google (2018), Google Git, Refactor Dependencies so keras_support can be imported directly. chromium.googlesource.com/external/github.com/tensorflow/tensorflow/+v1.10.0
- [38] Wikipedia, TensorFlow, Accessed 5 September 2021, en.wikipedia.org/wiki/TensorFlow

- [39] Auth0 (2017), Bruno Krebs, SQLAlchemy ORM Tutorial for Python Developers, Accessed 5 September 2021, auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/
- [40] Imaginary Cloud (2021), Mariana Berga, Tiago Franco, SQL vs NoSQL: When to use?, Accessed 7 September 2021, www.imaginarycloud.com/blog/sql-vs-nosql/
- [41] Postgres Tutorial, What is PostgreSQL? Accessed 7 September 2021, www.postgrestutorial.com/what-is-postgresql/
- [42] React, A JavaScript Library for Building User Interfaces, Accessed 7 September 2021, reactjs.org/
- [43] Rising Stack (2021), Rising Stack Engineering, The History of React.js on a Timeline. Accessed 7 September 2021, blog.risingstack.com/the-history-of-react-js-on-a-timeline/
- [44] Stack Overflow (2021), 2021 Developer Survey, Accessed 7 September 2021, insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021
- [45] Redux, Dan Abramov and the Redux documentation authors, React-Redux, Accessed 7 September 2021, react-redux.js.org/
- [46] California Data Science, What is machine learning and why it is important? Accessed 12 September 2021, californiadatascience.com/machine-learning
- [47] Towards Data Science, Teemu Kanstrén (2020), A look at Precision, Recall, and F1-core. Accessed 24 September 2021, towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652
- [48] Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016), Deep Learning, MIT Press
- [49] IBM Cloud Education (2021), Supervised vs. Unsupervised Learning: What's the difference, Accessed 12 September 2021, www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning
- [50] IBM Cloud Education (2021), Convolutional Neural Network, Accessed 12 September 2021, www.ibm.com/cloud/learn/convolutional-neural-networks
- [51] Keras Backend, Accessed 12 September 2021, faroit.com/keras-docs/1.2.0/backend/
- [52] Machine Learning Mastery (2019), Jason Brownlee, Use Weight Regularization to Reduce Overfitting of Deep Learning Models, Accessed 12 September 2021, machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/

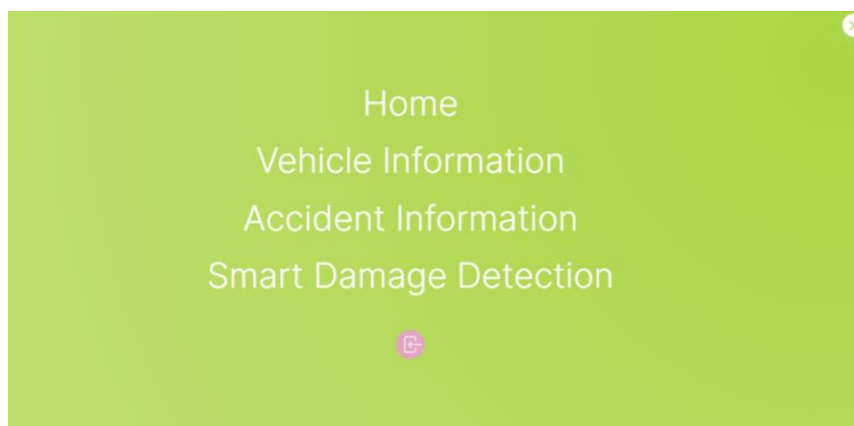
- [53] John Macintyre, Lazaros Iliadis, Ilias Maglogiannias, Chrisina Jayne (2019), Engineering Applications of Neural Networks Page 106, Springer Nature Switzerland AG 2019
- [54] Intel (2019), Adam-Milton-Barker, Inception V3 Deep Convolutional Architecture For Classifying Acute Myeloid/Lymphoblastic Leukemia, Accessed 14 September 2021, software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html
- [55] Ting Neo, Accessed 30 August 2020, [Access to the image dataset](https://opendatacommons.org/licenses/by/1.0/) is made available under the Open Data Commons Attribution License: opendatacommons.org/licenses/by/1.0/.
- [56] Peltarion, Accessed 15 May 2021, peltarion.com/knowledge-center/documentation/terms/dataset-licenses/car-damage
- [57] Sebastian Ruder (2016), An overview of gradient descent optimization algorithms, Accessed 22 September 2021, ruder.io/optimizing-gradient-descent/
- [58] Kingma Diederik, Ba Jimmy (2014), Adam: A Method for Stochastic Optimization, Cornell University, Accessed 23 September 2021, arxiv.org/abs/1412.6980
- [59] Towards Data Science, Richmode Alake (2020), Deep Learning: Understanding The Inception Module, Accessed 24 September 2021, towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652

8. USER INTERFACE

Here is the welcome page of 'Here We Are'! Crediting freepik.com for website images.



Select burger button on the upper-right side of the page, to open navigation.

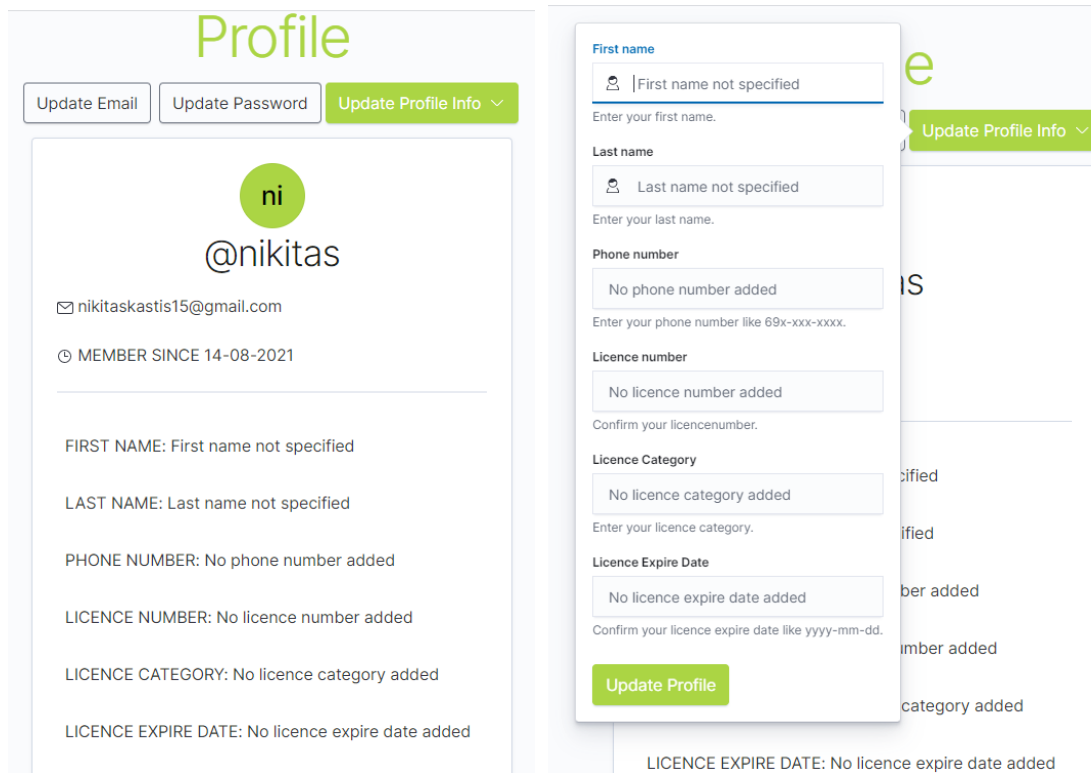


Select purple button, to go to login page. If you do not have an account, select 'Sign up here'.

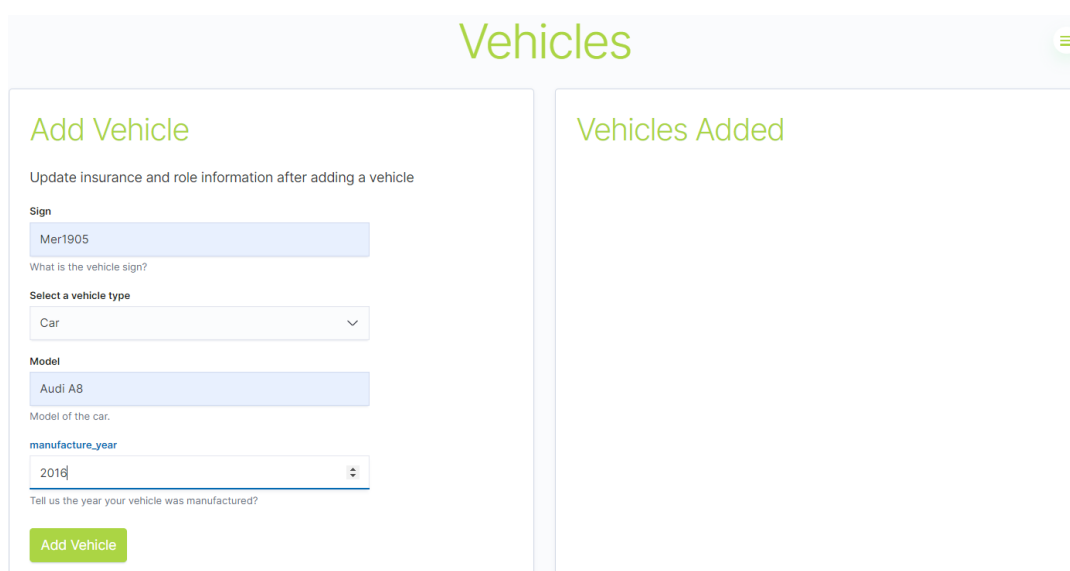
The image shows two side-by-side screenshots of web forms. The left screenshot is titled 'Login' and features an 'Email' field with 'user@gmail.com', a 'Password' field with masked characters, a 'Submit' button, and a link 'Need an account? Sign up here.' The right screenshot is titled 'Sign Up' and features an 'Email' field with 'user@gmail.com', a 'Username' field with 'your_username', a 'Password' field, a 'Confirm password' field, a checkbox for 'I agree to the terms and conditions.', a 'Sign Up' button, and a link 'Already have an account? Log in here.'

Complete the form and press Sign Up.

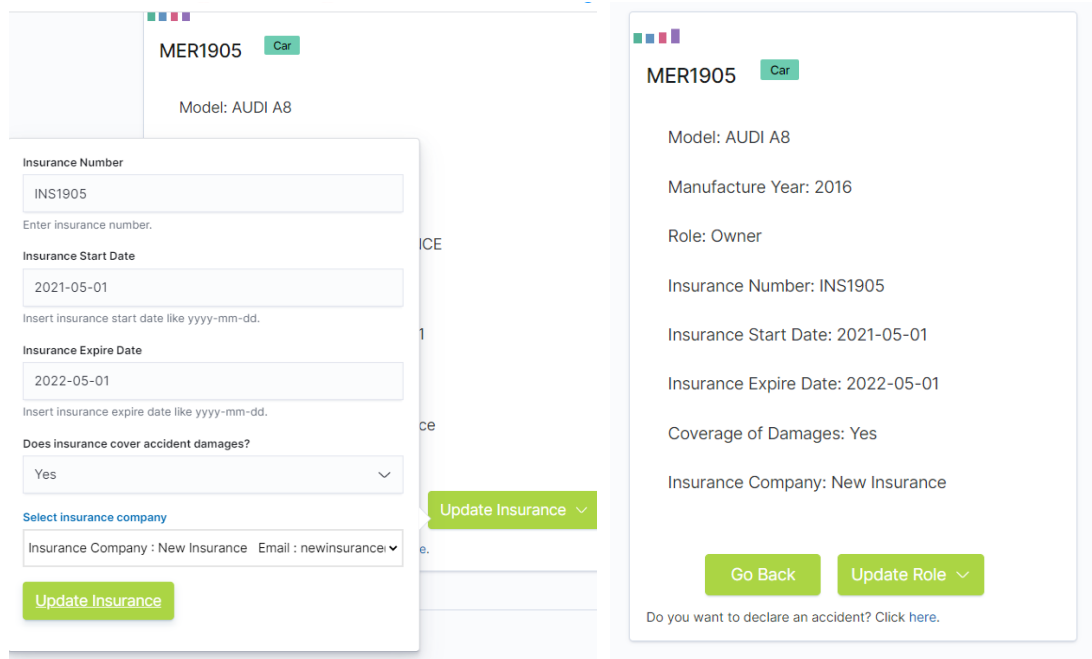
App directs you to Profile Page. Selecting "Update Profile Info" you can update your personal information. Remember to check information for each input. Press "Update Profile" button and, thanks to virtual DOM, update time is minimum.



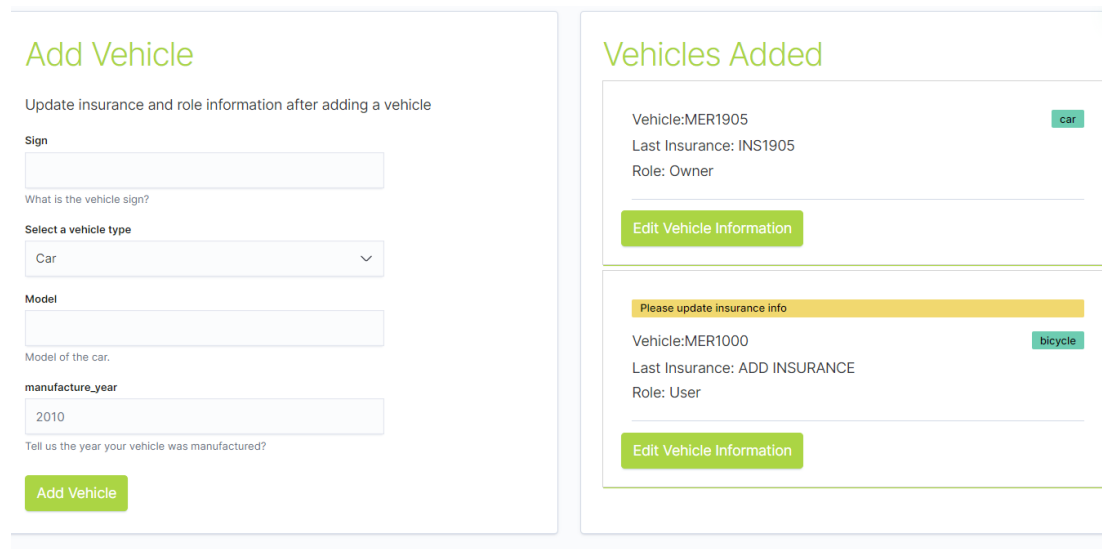
Through navigation, go to Vehicle Information, and add vehicles. On the right side of the screen, you can see the vehicles you have added.



So, if you have filled in the form correctly, when you press Add Vehicle, you are redirected to the vehicle page, to update info about the insurance and if you are owner of the vehicle.



Once you update insurance info, 'Update Insurance' button is hidden, till the insurance expires. In case you don't have an updated insurance, then an information badge appears to let you know that you should update your insurance.



In case you want to make a vehicle accident statement select from navigation 'Accident Information'. On the right part you can see your accident history.

Accidents

Declare Accident

Select a vehicle

Select your vehicle

Select your vehicle

Vehicle sign : MER1905	Insurance Number : INS1905
Vehicle sign : MER1000	Insurance Number : ADD INSURANCE

2021-08-14 02:08:00

Please enter your accident datetime like YYYY-MM-DD hh:mm:ss.

City

Declare the city that the accident took place.

Address

Declare the accident exact address.

Injuries

Declare if there were injuries in the accident.

Road Problems

Declare if there were road problems that led to the accident.

Declare Accident

Accidents Declared

Search by last name, address, date, city

In case you have not entered your information, the form declare accident does not appear. Additionally, you cannot select a vehicle that does not have a valid and updated insurance.

So, if you have filled in the form correctly, when you press Declare Accident, you are redirected to the accident page, to make your accident statement, add sketch, photos, and the other driver(s) that were in the accident.



14/8/2021, 2:08:00 π.μ.

VEKOU 8,GALATSI

Injuries: None
Road problems: None



Accident Statements

Driver Info

Name: Nikitas Kastis
Phone: 697-211-2020
Email: nikitaskastis@gmail.com
Licence: ME1905
Licence Category: C+
Licence Exp.Date: 2060-12-31

Vehicle Info

Vehicle Sign: MER1905
Vehicle Model: AUDI A8
Manufacture Year: 2016
Vehicle Type: Car
Driver Role in Vehicle: Owner

Insurance Info

Number: INS1905
Start Date: 2021-05-01
Expire Date: 2022-05-01
Damage Coverage: Yes
Company: New Insurance
Company Email: newinsurance@inc.com

Driver has to update accident statement.

Update Statement

Add Vehicle To Accident

Driver Full Name

Add driver's name you to accident.

Driver Email

Declare the driver's email.

Vehicle Sign

Declare the vehicle sign.

Insurance Number

Declare the vehicle insurance number.

Insurance Email

Declare the e-mail for the insurance company.

You can update re-entering info, using the same driver e-mail or vehicle sign. You can not add driver email or vehicle sign twice.

Remove Vehicle From Accident

Remove driver

Remove Driver

PENDING STATEMENTS



Pending Statements

Go Back

So, the form has all your data, and you have to Update Statement.

Accident Statement

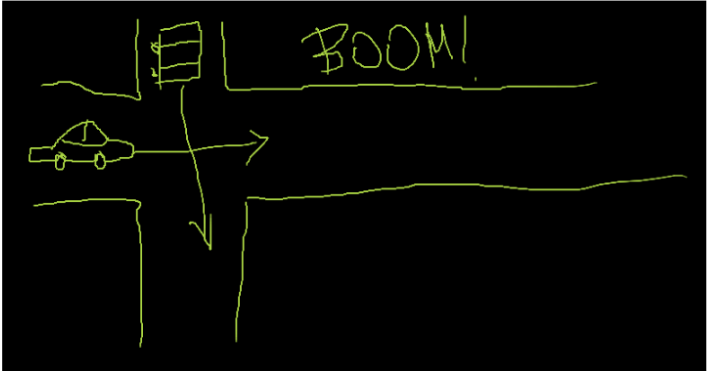
Write your comments. Who is the one to blame for the accident?

Select the reason of the accident

Hit when vehicle was parked
▼

Update Statement


Once you complete the statement, 'Add Accident Photo' and 'Draw Accident' buttons and the link for damage detection appear.



Update Accident Drawing

Draw the accident and press 'Update Accident Drawing', and you will see it in your statement right away.

Caused by: Hit while vehicle was driving on the right in the cross junction
Comments: He hit me!
 Smart damage Detection not used. [Click here.](#)



Select Image
▼

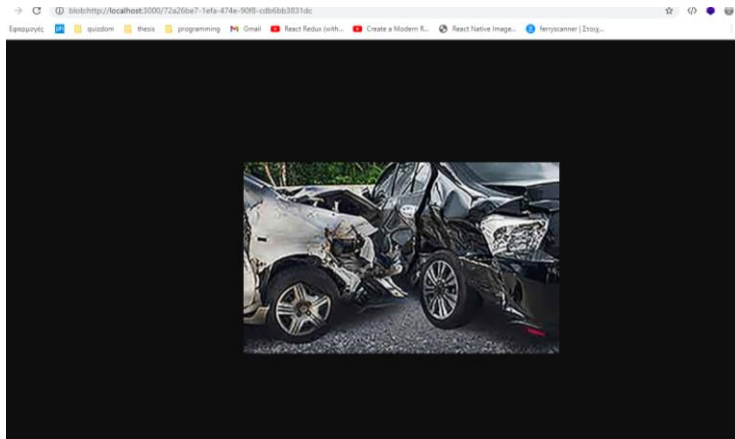
Select the accident image you want to check

Add Accident Photo

Draw Accident
▼

Update Statement
▼

Remember to add a photo depicting all vehicles included to the accident



Next, add accident photos, one-by-one. If you want to view one, you can select the image in the Select Image Option and press the “Open Image” button that just appeared and the selected image opens in a new window.

To add the other driver's data,

A screenshot of a web form titled "Add Vehicle To Accident". The form contains several input fields: "Driver Full Name" (Regina Phallange), "Driver Email" (kenadamsfriend@tibidabo.com), "Vehicle Sign" (MER1994), "Insurance Number" (INS1994), and "Insurance Email" (newinsurance@inc.com). There is a green "+" button next to the insurance email field. Below the form, there is a yellow warning message: "You can update re-entering info, using the same driver e-mail or vehicle sign. You can not add driver email or vehicle sign twice." Below the form, there is a section titled "Remove Vehicle From Accident" with a dropdown menu labeled "Remove driver" and a "Remove Driver" button.

 Pending Statements


PENDING STATEMENTS

Pressing “+” button, pending statements show the driver that has not used the page to make a response to the accident statement. You can add another driver if you want, or delete one, if they have not made an accident statement.

A screenshot of a web page showing a "Pending Statements" section. The section is titled "Pending Statements" and contains a box labeled "Driver Info" with the following details: Name: Regina Phallange, Email: kenadamsfriend@tibidabo.com, Vehicle Sign: MER1994, Insurance Number: INS1994, Insurance Co. Email: newinsurance@inc.com. Below the box is a "Go Back" button.

To use damage detection, either select the link in the accident statement or use navigation.

Smart Damage Detection




Tips for prediction
You can use estimate damage only for cars.
Select close-up photos to the damage.
Let us know if the prediction was right.


[Select Photo To Detect Damage](#)

[Go Back](#) [Refresh](#)

Press "Select Photo to Detect Damage" and add a photo of your damaged car. For example, let us upload the photo below:



0012.JPEG



Tips for prediction
You can use estimate damage only for cars.
Select close-up photos to the damage.
Let us know if the prediction was right.


Damage in Picture: damage on the front part

Am I correct?

[Yes](#)

[No](#)

The happy detective asks if the detection system is correct. And he is correct! So let's press "Yes". Then he asks if you want to add the prediction to an accident statement you have made.



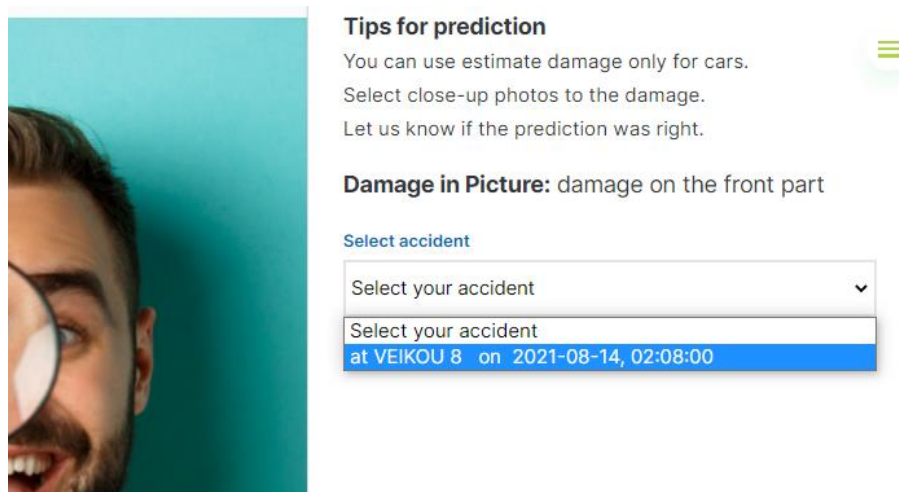
Damage in Picture: damage on the front part

You want to add prediction to an accident statement you have made?

[Yes](#)

[No, thanks.](#)

Pressing yes, you have to select in which accident you can add the prediction.



Tips for prediction

- You can use estimate damage only for cars.
- Select close-up photos to the damage.
- Let us know if the prediction was right.

Damage in Picture: damage on the front part

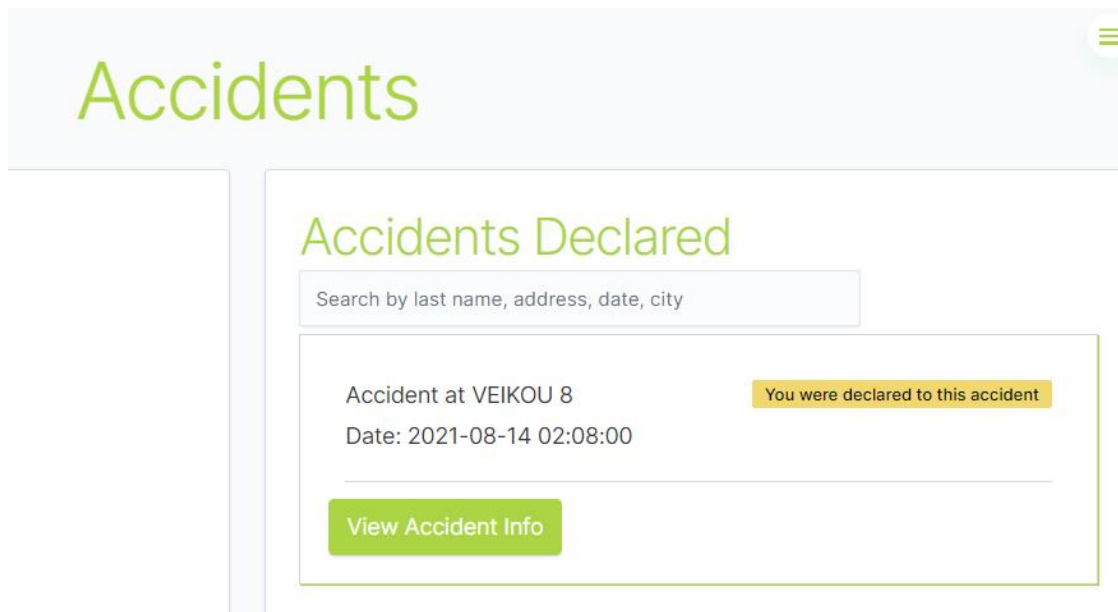
Select accident

Select your accident

Select your accident
at VEIKOU 8 on 2021-08-14, 02:08:00

Confirm and you are redirected to accident statement.

Let us see now how the other driver updates the statement. Say that they have an account and have already added the vehicle in their account with the insurance updated.



Accidents

Accidents Declared

Search by last name, address, date, city

Accident at VEIKOU 8 You were declared to this accident

Date: 2021-08-14 02:08:00

[View Accident Info](#)

Accident information page has already the accident and has a warning that the user should add an accident statement.

Pressing the “View Accident Info” they go to the same page, where they cannot – of course- edit anything in the statements of the other drivers. But they can see the exact same data, the sketch, the photos, everything.



14/8/2021, 2:08:00 π.μ.

VEIKOU 8,GALATSI

Injuries: None
Road problems: None



Accident Statements

Driver Info

Name: Nikitas Kastis
Phone: 697-211-2020
Email: nikitaskastis@gmail.com
Licence: ME1905
Licence Category: C+
Licence Exp.Date: 2060-12-31

Vehicle Info

Vehicle Sign: MER1905
Vehicle Model: AUDI A8
Manufacture Year: 2016
Vehicle Type: Car
Driver Role in Vehicle: Owner

Insurance Info

Number: INS1905
Start Date: 2021-05-01
Expire Date: 2022-05-01
Damage Coverage: Yes
Company: New Insurance
Company Email: newinsurance@inc.com

Caused by: Hit while vehicle was driving on the right in the cross junction

Comments: He hit me!

Smart damage detection problem: damage on the front part.



Select Image

Select the accident image you want to check

PENDING STATEMENTS



Pending Statements

Driver Info

Name: Regina Phallange
Email: kenadamsfriend@tibidabo.com
Vehicle Sign: MER1994
Insurance Number: INS1994
Insurance Co. Email: newinsurance@inc.com

Add Statement

In case your vehicle is wrongly written, communicate with Nikitas Kastis, who declared the accident or his insurance company. Thank you.

Go Back

They can find their data as the first driver stated in Pending statements, and they can add a statement. Select the button, and the same accident statement opens. Adding the data and pressing "Create Statement" button, they are no more in pending statements, and the page looks like this:



14/8/2021, 2:08:00 π.μ.

VEIKOU 8,GALATSI


Injuries: None
Road problems: None



Accident Statements

Driver Info	Vehicle Info	Insurance Info
Name: Nikitas Kastis Phone: 697-211-2020 Email: nikitaskastis15@gmail.com Licence: ME1905 Licence Category: C+ Licence Exp.Date: 2060-12-31	Vehicle Sign: MER1905 Vehicle Model: AUDI A8 Manufacture Year: 2016 Vehicle Type: Car Driver Role in Vehicle: Owner	Number: INS1905 Start Date: 2021-05-01 Expire Date: 2022-05-01 Damage Coverage: Yes Company: New Insurance Company Email: newinsurance@inc.com

Caused by: Hit while vehicle was driving on the right in the cross junction
Comments: He hit me!
Smart damage detection problem: damage on the front part.



Select Image
 Select the accident image you want to check

Driver Info	Vehicle Info	Insurance Info
Name: Regina Phallange Phone: 697-222-2222 Email: kenadamsfriend@tividabo.com Licence: FRI1994 Licence Category: C+ Licence Exp.Date: 2031-12-31	Vehicle Sign: MER1994 Vehicle Model: TAXI Manufacture Year: 1994 Vehicle Type: Car Driver Role in Vehicle: User	Number: INS1994 Start Date: 2021-05-01 Expire Date: 2021-12-01 Damage Coverage: Yes Company: New Insurance Company Email: newinsurance@inc.com


Caused by: Hit while vehicle was driving on the right in the cross junction
Comments: I did not see him
Smart damage Detection not used. [Click here.](#)

Remember to add a photo depicting all vehicles included to the accident

PENDING STATEMENTS



Pending Statements

 Driver Regina Phallange has made accident statement

Now they can update their statement, the same way the first user did. The only difference is that they cannot add or remove other drivers.

When the form is completed, each driver can declare his accident statement as completed.

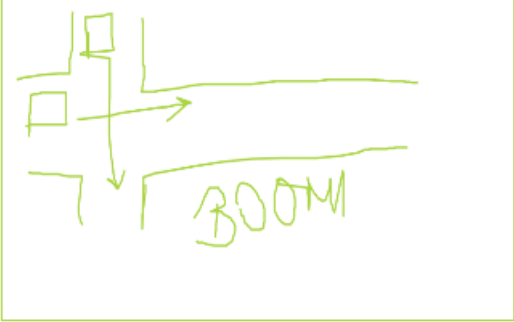
22/9/2021, 7:09:00 μ.μ. PATISSION 31,ATHENS

Injuries: NONE **Road problems:** NONE

Accident Statements

Driver Info Name: Nikitas Kastis Phone: 697-222-2020 Email: nikitaskastis15@gmail.com Licence: ME1905 Licence Category: C+ Licence Exp.Date: 2060-12-31	Vehicle Info Vehicle Sign: ME1905 Vehicle Model: BMW Manufacture Year: 2016 Vehicle Type: Car Driver Role in Vehicle: Owner	Insurance Info Number: INS1905 Start Date: 2021-09-21 Expire Date: 2021-10-21 Damage Coverage: Yes Company: NEW INSURANCE Company Email: newinsurance@inc.com
--	--	--

Caused by: Hit while vehicle was driving on the right in the cross junction
Comments: HE HIT ME

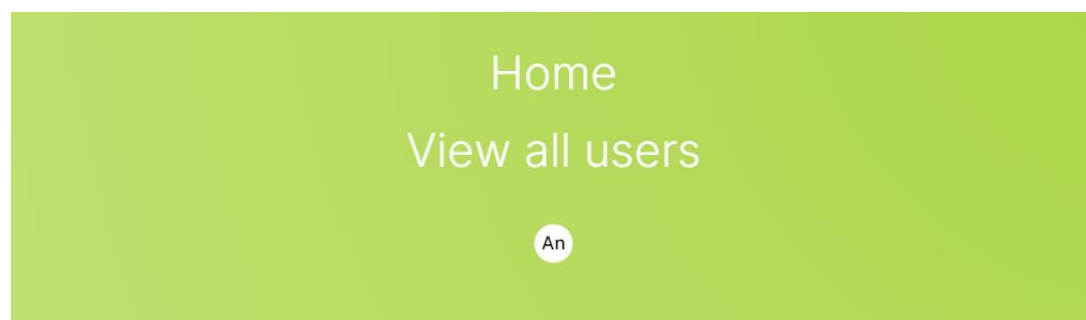


Select Image

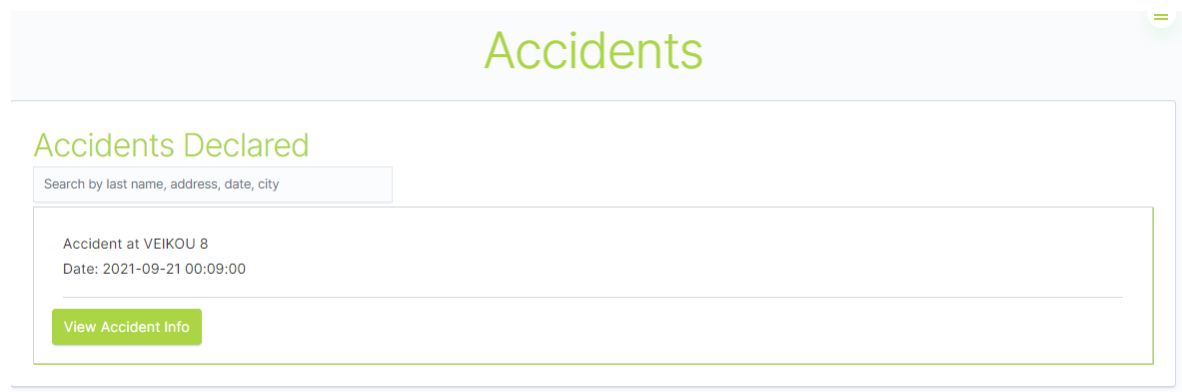
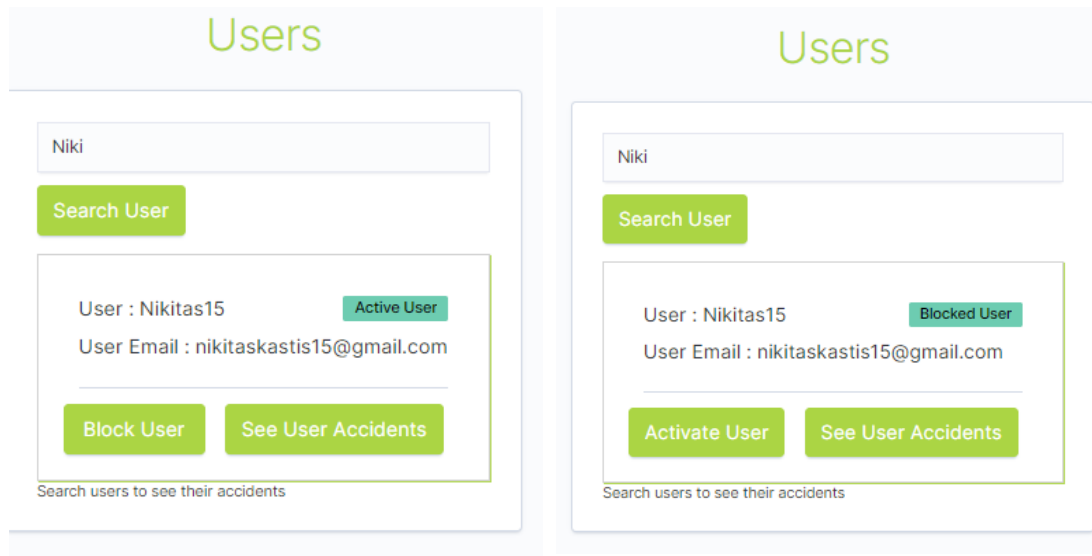
Select the accident image you want to check

Statement declared

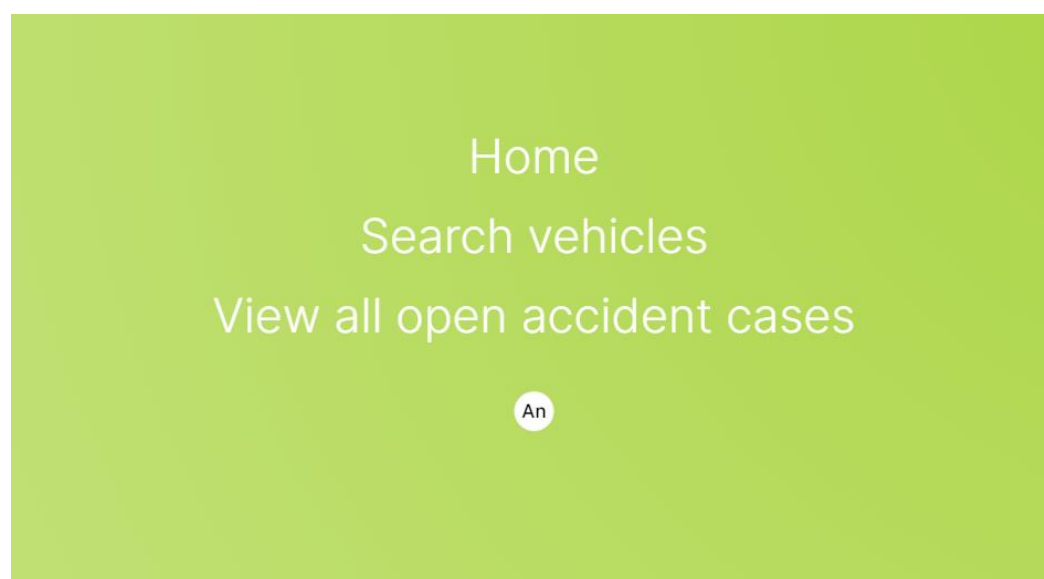
If a master user is logged in, navigation is different.



By searching a user, master may block him or see user's accidents. If the user is blocked, then master can activate user's account.



If an insurance company is logged in, navigation is different:



By searching a vehicle, insurance company may block him or see user's accidents. If the user is blocked, then master can activate user's account.

Vehicles

ME1905

[Search Vehicles](#)

Vehicle: ME1905 car

[See Vehicle Accidents](#)

Search vehicles to see their accidents

Accidents

Search by last name, address, date, city [Get all open cases](#)

Accident at PATISSION 31, ATHENS

Date: 2021-09-22 19:09:00

[View Accident Info](#) [Close case](#)

Pressing "Get all open cases" button, insurance company can view all its open cases of accidents. There is a "search by last name of driver, address, date, city" field that filters accidents according to user input.

Accidents

Search by last name, address, date, city [Get all open cases](#)

Accident at VEIKOU 3, GALATSI

Date: 2021-09-23 09:09:00

[View Accident Info](#) [Close case](#)

Accident at PLOUSIOU 5, EKALI

Date: 2021-09-22 20:09:00

[View Accident Info](#) [Close case](#)

Accident at PATISSION 31, ATHENS