Princess Sumaya University for Technology
King Hussein School for Computing Sciences

Baleghni

Prepared By:
Rinata Bakeer (20170169)
Mohammad Abu-Hilal (20160643)
Rami Habash (20170165)

Supervised By:
Dr. Anas Abu Taleb

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Computer Science

First-2020

Baleghni

Declaration of Originality

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is clearly cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university's regulations and is subject to legal actions.

Names and Signatures of team members:

Rinata Bakeer.

Rami Habash

Mohammad Abu-Hilal

Baleghni

Acknowledgments

First and foremost, we would like to state that we are grateful for being part of Princess Sumaya University that always guarantees to give the highest levels of education and opportunities for us.

We would like to express our gratitude and thanks for all the people with whom our project was successfully completed. Also, we would like to sincerely thank our dearest supervisor Dr. Anas Abu Taleb, for his guidance, patience, and advices that helped us throughout the project.

Special thanks are extended to Dr. Ammar Odeh, who kept providing us care, support, and knowledge that were a huge part in shaping our ideas into a project.

Finally, we sincerely thank all our families, friends and colleagues that gave us a continuous emotional support and motivation that kept us progressing and moving forward especially during the current hard days.

Baleghni

Summary

Traffic accidents are becoming very common nowadays. As more people are engaged with driving, the probability of an accident to happen is increasing. The result of such increase will demand a lot of effort and time from the police, mainly due to human errors.

The main purpose of our project is to provide an application that would be used on daily basis to report a car accident but in handy useful way. Also, it aims to ensure that in case of a car accident, the process of requesting, reaching and communicating with a policeman is efficient and fast until the arrival of the policeman and reporting finishes.

Our application will not only solve the traffic issues resulted from accidents, but also reduce wasted effort and time, as well as improving our country's roadways by sending detailed analysis regarding the accidents to the government.

Our methodological approach is to build this application to provide an efficient rapid car accident reporting experience for all drivers, and to analyze the car accident reports to improve streets, areas and driving rules.

Baleghni

List of Abbreviations

Table 1: Table of Abbreviations

Abbreviation	Meaning
UML	Unified Marked Language
UI	User Interface
ERD	Entity Relationship Diagram
FK	Foreign Key
PK	Primary Key

Baleghini

Table of Contents

Chapter 1 Introduction **10**

1.1 Overview **10**

1.2 Problem Statement **11**

1.2.1 Problem Description 11

1.2.2 Project Outcome 11

1.2.3 Target Audience 11

1.3 Related Work **11**

1.4 Document Outline **12**

Chapter 2 Project Plan **13**

2.1 Project Deliverables **13**

2.2 Project Tasks **13**

2.3 Roles and Responsibilities **15**

2.4 Risk Assessment **15**

2.4.1 Risk Identification and Analysis 15

2.4.2 Risk Response 16

2.5 Cost Estimation **16**

2.6 Project Management Tools **17**

Chapter 3 Requirements Specification **18**

3.1 Stakeholders **18**

3.2 Platform Requirements **18**

3.3 Functional Requirements **19**

3.4 Non-Functional Requirements **23**

3.5 Other Requirements **24**

Chapter 4 System Design **25**

4.1 Logical Model Design **25**

4.1.1 Use Case Diagrams 25

4.1.2 Class Diagrams 31

4.1.3 Object Diagram 32

4.1.4 Activity Diagrams 33

4.1.5 System Architecture Diagrams 35

4.1.6 Sequence Diagrams 37

4.1.7 State Diagrams 39

4.2 Physical Model Design **41**

4.2.1 UI Design 41

Baleghni

4.2.2 Database Design 46
4.2.3 Reports Design 48

4.3 Test Plan 53

Chapter 5 Implementation 55

5.1 Programming Languages, Tools and APIs 55
5.1.1 Front End 55
5.1.2 Back End 55

5.2 Overview 56

5.3 Main Algorithms and Coding Conventions 57
5.3.1 General User 57
5.3.1.1 Sign in using Email/Password 57
5.3.2 Driver User 58
5.3.2.1 Register Driver 58
5.3.2.2 Driver Request Police 59
5.3.2.3 Driver Cancel Request 61
5.3.3 Police User 63
5.3.3.1 Police Finish Request 63
5.3.3.2 Police Change Status 64
5.3.4 Dispatcher 65
5.3.4.1 Assign Police Agent to Request 65

5.4 Features 66

Chapter 6 Testing 67

6.1 Testing Approach 67
6.1.1 Functional Testing 67
6.1.2 Non-functional Testing 68

6.3 Testing Samples 69

6.2 Testing Tools 72

6.3 Tested Features 72

Chapter 7 Conclusions and Future Work 74

Appendix A Users' Manual 75

Driver 75

Dispatcher 81

Police Agent 84

Appendix B Document Changes 89

Appendix C Code Documentation 93

References 94

Baleghni

Table of Figures

FIGURE 1: GANTT CHART 13
FIGURE 2: USE CASE DIAGRAM FOR GENERAL USER FUNCTIONALITIES 25
FIGURE 3: USE CASE DIAGRAM FOR DRIVER-SIDE INTERACTION IN REQUESTS 28
FIGURE 4: USE CASE DIAGRAM FOR POLICE/DRIVER INTERACTION AFTER REQUEST 30
FIGURE 5: CLASS DIAGRAM FOR BALEGHNI 31
FIGURE 6: OBJECT DIAGRAM FOR BALEGHNI REQUEST 32
FIGURE 7: ACTIVITY DIAGRAM FOR REQUESTING A POLICE AGENT 33
FIGURE 8: ACTIVITY DIAGRAM FOR AFTER RECEIVING A REQUEST 34
FIGURE 9: SYSTEM ARCHITECTURE FOR REQUEST PROCESSING 35
FIGURE 10: SYSTEM ARCHITECTURE FOR DATA ANALYSIS PROCESS 36
FIGURE 11: SEQUENCE DIAGRAM FOR LOGIN 37
FIGURE 12: SEQUENCE DIAGRAM FOR REQUEST PROCESSING 38
FIGURE 13: STATE DIAGRAM FOR DRIVER REQUESTING 39
FIGURE 14: STATE DIAGRAM FOR POLICE SIDE AFTER REQUEST RECEIVAL 40
FIGURE 15: LOGIN INTERFACE 41
FIGURE 16: SIGN UP PAGE INTERFACE 41
FIGURE 17: HOME PAGE INTERFACE 41
FIGURE 18: HISTORY PAGE INTERFACE 42
FIGURE 19: CONTACT US PAGE INTERFACE 42
FIGURE 20: DRIVER PROFILE PAGE 42
FIGURE 21: DRIVER PROFILE INTERFACE WITH EDIT VEHICLE 43
FIGURE 22: POLICE LOGIN PAGE INTERFACE 44
FIGURE 23: REQUESTS PAGE INTERFACE 44
FIGURE 24: POLICE HOME PAGE INTERFACE 45
FIGURE 25: POLICE PROFILE PAGE INTERFACE 45
FIGURE 26: BALEGHNI DATABASE ENTITY RELATIONSHIP DESIGN 46
FIGURE 27: BALEGHNI DATABASE SCHEMA 47
FIGURE 28: VEHICLE ACCIDENT HISTORY FOR AN OWNER REPORT 48
FIGURE 29: ANNUAL ACCIDENT ANALYTICS PER AREA REPORT 49
FIGURE 30: ACCIDENT REQUESTS HISTORY ON SPECIFIC DATE REPORT 50
FIGURE 31: DRIVER ACCIDENTS REPORT 51
FIGURE 32: UPCOMING YEAR CAR ACCIDENTS PREDICTION REPORT 52
FIGURE 33: USER MANUAL - LOGIN PAGE 75
FIGURE 34: USER MANUAL - SIGN UP PAGE 76
FIGURE 35: USER MANUAL FORGOT PASSWORD PAGE 77
FIGURE 36: USER MANUAL - ONE TIME LOCATION PERMISSION REQUEST 77
FIGURE 37: USER MANUAL - HOME PAGE 78
FIGURE 38: USER MANUAL -DRIVER HISTORY PAGE 79
FIGURE 39: USER MANUAL - DRIVER REQUEST DETAILS SAMPLE. 79
FIGURE 40: USER MANUAL - DRIVER PROFILE SCREEN 80
FIGURE 41: USER MANUAL - DISPATCHER LOGIN PAGE 81
FIGURE 42: USER MANUAL - DISPATCHER ASSIGNMENT PROCESS 82

Baleghni

FIGURE 43: USER MANUAL - DISPATCHER MAP PAGE 82
FIGURE 44: USER MANUAL - DISPATCHER REGISTER POLICE PAGE 83
FIGURE 45: USER MANUAL - DISPATCHER SIGN OUT IN MAP PAGE 83
FIGURE 46: USER MANUAL - POLICE HOME PAGE 84
FIGURE 47: USER MANUAL - POLICE HOME PAGE WITH ONLINE MODE 84
FIGURE 48: USER MANUAL - POLICE RECEIVES NEW REQUEST 85
FIGURE 49: USER MANUAL -POLICE HISTORY PAGE 87
FIGURE 50: USER MANUAL - POLICE REQUEST DETAILS SAMPLE. 87
FIGURE 51: USER MANUAL - POLICE PROFILE 88
FIGURE 52: DISPATCHER TABLE IN FIREBASE 90
FIGURE 53: CANCELLED REQUEST IN FINISHED REQUESTS TABLE IN FIREBASE 90
FIGURE 54: COMPLETED REQUEST IN FINISHED REQUESTS TABLE IN FIREBASE 91
FIGURE 55: PENDING REQUESTS IN SENT REQUESTS TABLE IN FIREBASE 91
FIGURE 56: ACCEPTED REQUESTS IN SENT REQUESTS TABLE IN FIREBASE 92
FIGURE 57: DRIVER IN USERS TABLE FIREBASE 92
FIGURE 58: POLICE IN USERS TABLE FIREBASE 92

Table of Tables

TABLE 1: TABLE OF ABBREVIATIONS 5
TABLE 2: TABLE OF RELATED WORKS COMPARISONS 12
TABLE 3: DOCUMENT OUTLINE 12
TABLE 4: PROJECT TASKS DESCRIPTION 14
TABLE 5: ROLES AND RESPONSIBILITIES TABLE 15
TABLE 6: RISK IDENTIFICATION AND ANALYSIS: 15
TABLE 7: RISK RESPONSE: 16
TABLE 8: STAKEHOLDERS 18
TABLE 9: CLIENT-SIDE REQUIREMENTS 18
TABLE 10: SERVER-SIDE REQUIREMENTS 18
TABLE 11: FUNCTIONAL REQUIREMENTS P1 19
TABLE 12: FUNCTIONAL REQUIREMENTS P2 20
TABLE 13: NON-FUNCTIONAL REQUIREMENTS 23
TABLE 14: TEST PLAN TABLE 53
TABLE 15: FEATURES TABLE 66
TABLE 16: TESTED FEATURES 72
TABLE 17: DOCUMENT CHANGES TABLE 89

Balaghni

Chapter 1
Introduction

1.1 Overview

Baleghni is a cloud-based and easy-to-use mobile application that focuses on reporting traffic accidents in much-improved way, the application will ensure that the location is sent accurately instead of having to describe it over the phone, which helps the police agents to get the exact destination for the accident through optimal route which leads to saving a lot of time and effort for both the drivers and the police, the users can also optionally send photos of the accident and the vehicle included in the accident; sending photos can make it easier for the agent to identify the accident, or in case of hit and run, it will make it much easier for the culprit to be identified by giving the police a better judgement about the culprit's car and maybe the number plate.
Users that can use this application are classified into two categories, drivers and police agents, for the drivers, the data that is required is their name, national ID or passport number, and phone number, the difference for the police agent is that their police identification id is also required .
When a driver reports an accident through their mobile, it will send all the important details (Name, Location, phone number) for the nearest police agent which will have the privileges of viewing their profile, the driver should also be able to retrieve the police agent's information. This will ensure optimal and efficient work.
The system also aims to reduce the amount of accidents through the data that is collected , for an instance, if a certain road gets a lot of accidents, more measures can be taken in order to prevent or reduce accidents in that specific road in the future, an example for that is that a lot of accidents happen on the desert road in Jordan, if the government knows the exact location of the accidents, they can triangulate the most dangerous areas and they can take more actions to prevent the loss of lives and resources.

Baleghni

1.2 Problem Statement

1.2.1 Problem Description

Since the number of cars in Jordan has increased reaching millions within the last couple of years, causing more car accidents to occur as well, which requires more time and effort to report. The traditional report process requires preparation for report forms; filling, auditing and analyzing with a high human error. Moreover, most of the time, requesting kroka takes a lot of time due to many factors: police arriving lately to the scene, people don't know how to describe their place, police line busy, lost data transfer between police.

1.2.2 Project Outcome

The outcome of this project is a mobile application that aims to provide a more efficient accident reporting service.

1.2.3 Target Audience

This mobile application targets every Jordanian driver who have a driving license and a mobile phone. Users first have to sign up or login, afterwards, they will be able to request a traffic police agent to the scene, this process will be much more efficient due to sending an accurate location that the agent can use to reach the destination in the quickest route possible, saving time and effort for both the driver and the police.

1.3 Related Work

We would like to clarify these applications similar to ours is currently not available in Jordanian's market. But there are some in Saudi Arabia, Dubai and United States.

Najim, an application that is available in Saudi Arabia and Emirates, which enables drivers/insurance agents to report an accident by taking pictures of the accident and order a police officer right away by using location services. [1]

Dubai police (**شرطة دبي**), an UAE government app. That is not only for reporting accidents, it includes all kind of services a driver need from the police. [2]

Accident Reporter – Track Vehicle incidents, an application that is available in the United States, just like Najm app but it only allows the driver to report to your insurance company not the government's traffic department. [3]

Baleghni

Table 2: Table of Related Works Comparisons

	Najim	Dubai Police	Accident Reporter	Baleghni
Arabic Language	YES	YES	NO	YES
Reduces Battery Life	YES	YES	YES	NO
Generates accurate statistical reports for authorities	NO	NO	NO	YES
Smart Guidelines for taking pictures/ recording videos	NO	YES	YES	YES
Sends current location fast	NO	YES	YES	YES
Covers all areas (countries)	NO	NO	NO	YES
Provide Guidelines as videos or pictures	YES, but doesn't work	YES	YES	YES
Checking the report status	YES	YES	YES	YES
Voice Assistant for elders	NO	YES	NO	YES
Fast to send requests	NO	YES	NO	YES

1.4 Document Outline

Our documentation is structured as follows:

Table 3: Document Outline

Chapter Title	Description
Chapter 1: Introduction	Provides a concise overview of the project's idea and the problem's solution, as well as the related work and the targeted audience for this project.
Chapter 2: Project Plan	Detailed description of the project's flow, deliverables, tools required and the tasks with their distribution amongst team members, it also contains the risk assessment for each task and the cost estimation of the project.
Chapter 3: Requirements Specification	The functional and non-functional requirements and required software and hardware specifications with the declaration of stakeholders.
Chapter 4: System Design	Detailed analysis for the system components for both the high- and low-level design using UML diagrams and such.

Chapter 2
Project Plan

2.1 Project Deliverables

- Source code of the implementation
- documentation of the project
- graphs and diagrams
- working prototype of the system

2.2 Project Tasks

Our project is subdivided into 3 high-level tasks:

1. Research and Analysis Phase: it includes all the research and analysis that is required to start our project and the problems that we are going to solve by implementing it.
2. Design Phase: Designing all the charts and layouts that will determine our development strategy.
3. Implementation Phase: Implementing our idea into a real-life application.

Figure 1: Gantt Chart

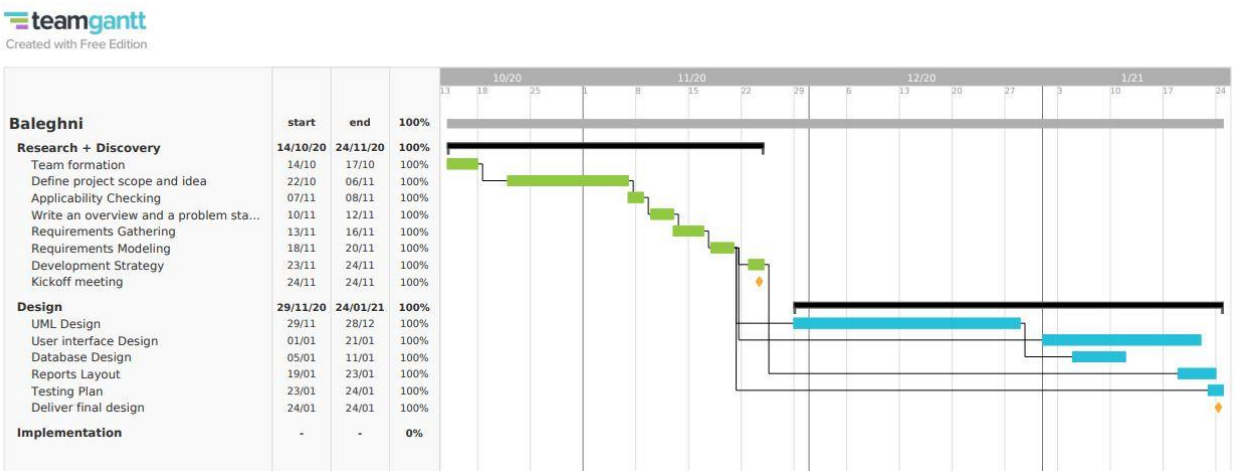


Table 4: Project Tasks Description

Task ID	Task Name	Description	Time Duration	Dependencies	Assigned To	Progress
		Phase 1: Research and Analysis				
A1	Define Project Scope	Finding the idea to be made.	10 Days	----	All	100%
A2	Applicability Checking	Checking if project idea is applicable.	2 Days	A1	Rinata	100%
A3	Write An overview and a problem statement	Giving an abstract overview of our idea and providing a precise description of the problem and how to solve it.	3 Days	A2	Mohammad and Rami	100%
A4	Requirement Gathering	Searching and discovering the system requirements by competitor analysis and defining features.	4 Days	A3	All	100%
A5	Requirement Modeling	Subdividing our requirements into functional and non-functional requirements and giving a brief explanation of both.	3 Days	A4	Rami and Mohammad	100%
A6	Development Strategy	Identifying and selecting how to reach our approach and allocating the resources (software) in order to achieve our goal.	3 Days	A5	All	100%
		Phase 2: Design				
D1	UML Design	Designing different diagrams that show how our system functions.	1 month	A5	Mohammad and Rami	100%
D2	User Interface Design	Designing our application's different interfaces that our users will work with	2 weeks 4 days	A5	Rinata	100%
D3	Database Design	Designing the databases' models that will be used in our application	1 Week	D1	Rinata and Rami	100%
D4	Report Layout	Illustrating the initial generated analysis report by the system, including Driver history and annual car accident analysis report.	5 days	A6	Rinata	100%
D5	Testing Plan	Examining the functionality of the application using Blackbox testing method (without peering into its internal work)	2 days	A5	Rinata	100%

2.3 Roles and Responsibilities

Table 5: Roles and Responsibilities Table

Team Member	Responsibilities
Rinata Bakeer (Project Manager)	Big Data, Database, Design, Documentation
Rami Habash	Cloud and Database, Documentation
Mohammad Abu Hilal	Mobile Application Development, Documentation

2.4 Risk Assessment

2.4.1 Risk Identification and Analysis

This table contains all the risks that may occur to some of our tasks, by providing the probability in the range of low-medium-high and the impact on the process.

Table 6: Risk Identification and Analysis:

Risk ID	Task ID	Task Name	Risk	Probability	Impact
			Phase 1: Research and Analysis		
R1	A1	Define Project Scope	Unfinished explanation of the idea and problem.	Low	Weak comprehension of our project's scope and how to start our project.
R2	A2	Applicability Checking	Not solving the problem, we stated to solve and being unacceptable by stakeholders or in the real-world.	High	Project scope changing.
R3	A4	Requirement Gathering	Gathering incomplete and ambiguous requirements that are required for our system.	Medium	This leads to unusable and unstable deliverables features.
R4	A6	Development Strategy	Developing a strategy that acts as a hindrance leading to not accomplishing our goal.	low	Starting with a new development strategy from scratch.
			Phase 2: Design		
R5	D1	User Interface Design	Doesn't fulfill the user's desirable interface.	low	Less users will use our application.
R6	D2	Database Design	Risk in designing the database schema	high	Causes problems in our application (requests, analysis, etc..)

2.4.2 Risk Response

Table 7: Risk Response:

Risk ID	Risk	Response
R1	Unfinished explanation of the idea and problem.	Revising the scope, and then either stay or change the idea depending on the result of the revision.
R2	Not solving the problem, we stated to solve and being unacceptable by stakeholders or in the real-world.	Change the project scope.
R3	Gathering incomplete and ambiguous requirements that are required for our system.	Do more research and change the requirements gathering method.
R4	Developing a strategy that acts as a hindrance leading to not accomplishing our goal.	Starting with a new development strategy from scratch.
R5	Doesn't fulfill the user's desirable interface.	Read the users' comments about the layout and how to make it better to suit them or do a survey that will help with choosing a new user interface.
R6	Risk in designing the database schema	Redesigning the database schema.

2.5 Cost Estimation

We estimate the starting cost to be in a range of 824-3000+.

Includes:

- Google Development Account which is a one-time 25$ fee.
- IOS Development Account which is an annual 99$ fee.
- Google Map API which has a starting cost of 200$ monthly payment as well as a dynamic cost depending on number of requests.
- External Hosting Server to be able to store data and send requests between the users. Starting at 500-2000
- Maintenance $200 per month.

2.6 Project Management Tools

The tools that we have used and enabled us to manage, synchronize, and clarify our project work more efficiently:

- Microsoft Word: a documentation tool that we used to create, write, edit and format our project documentation.
- GitHub: a software platform that is used to host all our documentations, codes, and progress reports.
- Team Gantt: used to create the Gantt chart.
- Adobe XD: used to create UI designs.
- Draw.io: used to create all UML designs.
- Dbdesign.com: used to create a database third normalization scheme.
- Google drive: to submit the final documentation.

Baleghni

Chapter 3
Requirements Specification

3.1 Stakeholders

Table 8: Stakeholders

ID	Stakeholders	Description
S1	Municipality of Greater Amman	They are responsible for overseeing the application, providing guidelines and knowledge about Kroka rules, data that can be used.
S2	Insurance companies	Making sure that the validity of driver's insurance data is correct.
S3	Drivers	Uses the application to report an accident (Main Actor).
S5	Police Agents	Uses the application to Accepts Driver requests.
S4	Ministry of Transport	They will benefit from the reports generated by our system.

3.2 Platform Requirements

- **Client-Side**

Table 9: Client-Side Requirements

Requirement	Description	Priority
PRC1	Mobile Phone	Essential
PRC2	Internet Connection	Essential
PRC3	Access to Location	Essential
PRC4	Access to Camera	Essential

- **Server-Side**

Table 10: Server-Side Requirements

Requirement	Description	Priority
PRS1	Internet Connection	Essential
PRS2	Networking Server	Essential
PRS3	Large Database Storage	Essential

3.3 Functional Requirements

The two following tables shows the functional requirements in our system, including the process of that function in the second table.

Table 11: Functional Requirements p1

ID	Requirement	Description	Priority
FR1	Sign Up	Any user will be able to make an account of their own.	Essential
FR2	Login	User will enter email and password.	Essential
FR3	Forgot password	Request a password change through the email or mobile phone.	Essential
FR4	View Profile	Users will be able to view their profile showing their information.	Essential
FR5	Edit Profile	Users will be able to edit their information.	Essential
FR6	Add a vehicle	Users should be able to add vehicles to their profile in which they will be able to choose from when they are requesting.	Essential
FR7	Update a vehicle	Users should be able to update the vehicles they previously added on their profile.	Essential
FR8	Delete a vehicle	Users should be able to delete the vehicles that are already added to their profile.	Essential
FR9	Change password	Users will be able to change their password through a button.	Essential
FR10	Request a police agent	The driver should be able to report an accident.	Essential
FR11	Attach an image on request	Extra option for the driver to add an image upon the request of a police agent.	Recommended
FR12	Choose a vehicle	Users should be able to pick a vehicle when they want to do a request if it was already added to their profile previously.	Essential
FR13	Driver confirm request	Drivers should be able to confirm their request after attaching the location and optionally, vehicle and images.	Essential
FR14	Driver cancel request before confirming it	Drivers should be able to cancel their request before confirming it.	Essential

FR15	Driver cancel request after confirming it	Drivers should be able to cancel their request within 5seconds of confirming it.	Essential
FR16	Display notification for police agents upon requests arrival	Show notification for chosen free police agent informing them about the request.	Essential
FR17	Status police change	Status of every police agent would change based on accept if they accepted a request, arrived if they arrived at the location of the request, end request at the end of the request.	Recommended
FR18	Display popup for driver showing their request has been accepted	The driver should receive a popup notification showing that their request has been accepted.	Essential
FR19	Display Driver/agent information	The information of both driver and police agent upon request acceptance	Essential
FR20	Map Tracking	The police agent should be able to track the location of the accident.	Essential
FR21	Language Control	Users should be able to change the application language between English and Arabic.	Essential
FR22	Display request history	The users should be able to check the history which shows their previous requests.	Essential
FR23	Search History by Date	Users should be able to search in their history using the start and end dates as filters.	Essential
FR24	Customer service	The user should be able to contact customer service if any help is needed.	Recommended

Table 12: Functional Requirements p2

Baleghi

ID	Input	Output	Constraint	Process
FR1	User inputs Id, name, phone number, email, password	Error message appears if there is an error while signing up, otherwise the registration will proceed.	It must not exist in the account database; password length must be minimum 8-long.	The user will be able to select between driver and police agent, then will enter the required information. Police agent will also require POLICE ID. Admins will be able to sign up and given privileged settings through verification. Police agents will also require verification.
FR2	Email, password	Error message appears if non-existing email or wrong password, otherwise proceed.	Email and password must be matching with database entry.	The user will enter their email and password, if the information match database data, they will proceed and will be redirected to the appropriate window corresponding to their type. Else will display an error message. Each of the admins, drivers and agents will have a different default dashboard.
FR3	Email or phone number	An email containing a password resetting link will be send to the user if the input email was correct.	Email must be matching in database entry.	When the user clicks on ‘’forgot password’’, the user will enter their email or phone number and then click confirm. If email was selected, an email will be sent back to the email if it was correct, otherwise to the mobile phone; containing a code that can be used to change the password.
FR4	None	Redirected to profile page.	None	The user will be able to view their profile with a click on the main dashboard, the user will be taken to a new page showing their information, there is also a ‘’change password’’ button on that page.
FR5	None	Edited information.	None	Through the profile page, the user will be able to edit their information, excluding national ID.
FR6	Click on add vehicle	Vehicle added.	Vehicle wrong info.	The user will click on Add vehicle from their profile, then they will proceed to fill the information and click on finish.
FR7	Click on update vehicle	Vehicle information updated.	Vehicle wrong info.	The user will click on Update vehicles from their profile, then they will proceed to update the information of the chosen vehicle and click on save button.
FR8	Click on delete vehicle	Vehicle deleted.	None	The user will click on Delete vehicles from their profile, then highlight the vehicles and proceed to click delete button.
FR9	None	The user will be redirected to another page in which they can change their password.	Password length greater than 6.	Through the profile page, The user will be able to click on ‘’change password’’ it will redirect them to another page in which they will be asked to enter the current password, and the new password with its

				confirmation, if the password is valid then it will be confirmed and the user will get back to the main page, otherwise error and must enter the passwords again.
FR10	The driver will input the location of the accident	A request will be sent the nearest police agents containing driver information	Free nearby agent.	The driver can order a request for a police agent, in which they have to put the location of the accident and more information such as the car plate number and so, then the request will be sent to all nearby police agents to be accepted by one
FR11	Click on attach an image	Image attached to the request.	None	Driver should be able to attach an image of the accident to the request before the confirmation.
FR12	Click on choose vehicle	Vehicle chosen for request.	None	While requesting, the user will click on Choose vehicle then will proceed to pick a vehicle out of the previously added ones, or none.
FR13	Driver click on confirm button	Driver's request will await 5 seconds and then be sent out for police agents.	None	When the driver clicks on confirm request, the request will be sent after 5 seconds to all nearby agents.
FR14	Driver click on cancel button	Driver's request will be cancelled, return to main dashboard.	None	Driver should be able to cancel the request before confirming it, returning them to the main dashboard.
FR15	Driver click on cancel button	Driver's request will be removed.	None	Driver should be able to cancel the request within 5 seconds of confirming it, will return to the home dashboard upon cancelling.
FR16	Driver requests an agent	A notification will appear to nearby police agents about this request.	No free nearby agent.	Whenever a driver orders a request, the system will search for free police agents, repeatedly widening the range after specific time-period until accepted. Upon finding and linking, police's and request's statuses will be updated. After that, the police's and the driver's information will be shared with each others.
FR17	None	Status change of police	None	The status of the police agents will be automatically updated upon accepting or ending requests.
FR18	None	A notification will be received by the driver upon request acceptance.	None	Whenever a police agent accepts a driver's request, a notification will pop up for the requesting driver showing that the request has been accepted.
FR19	Request accepted.	The information of the driver and the agent will be shown to each other respectively.	None	When a request gets accepted by a police agent, they will receive the requestor's information. The driver will also receive the information of the accepting agent.

FR20	Request accepted.	Police agent will have the accident's location. Driver will have the police agent's location.	None	Police agent will track the location that was attached to the accepted request. Drivers can track the location of the dispatched police agent that is on the way to the accident scene.
FR21	Click on change language button	Language will be changed.	None	The users will click on the change language option, two choices will appear, English and Arabic. If the different language is chosen, then the application language will change to the selected one.
FR22	Saving request history in database	Show request history.	None	The users will have an option to go into ''request history'' which will show all the user's requests log, or, for the police agents, they see the log of accepted requests.
FR23	Input Date	Requests within the dates inserted will be viewed.	No requests within that time period.	The user will enter the dates in which they would like to search. Can be either in-between or day dates. Any request within that time stamp will be viewed.
FR24	None	None	Free Customer service agent	Users can request help from our customer service agents through a click on the ''help'' option found in the extra menu.

3.4 Non-Functional Requirements

The two following tables shows the non-functional requirements in our system,

Table 13: Non-Functional Requirements

ID	Requirement	Description
NFR1	Security	Provide high-level encryption over the data of users which will not be visible or shared with other users, as well as sending verification messages to the user's emails and phones on logins to ensure their safety.
NFR2	Easy to use/Usability	The interface of the application must be easy to understand and work with even for people with little knowledge of technology.

Baleghi

NFR3	Reliability	The application system must be able to handle many requests without crashing having any problems such as lag or delays.
NFR4	Performance	The proposed system aims to have as smooth and accurate experience for the users as possible. Requests should be sent and processed quickly. This will save cost (time, effort, money) for the users. Our system will handle a thousand of requests at a time.
NFR5	Transparency	The users will know everything that is happening, between the driver and the dispatched agent, the drivers can track the agent's location as well as being provided with their information. The dispatched agent will know the requesting driver's information.
NFR6	Scalability	The system should handle a huge number of users and requests, it can also be accessed from all mobile devices.
NFR7	Availability	The system is available every hour of every day of the week 24/7.

3.5 Other Requirements

Other requirement will be based on the following contributors' rules: -

- Ministry of Transport.
- Government.
- Municipality of Greater Amman.
- Insurance companies.

Baleghni

Chapter 4

System Design

4.1 Logical Model Design

4.1.1 Use Case Diagrams

4.1.1.1: Use Case Diagram for General User Functionalities

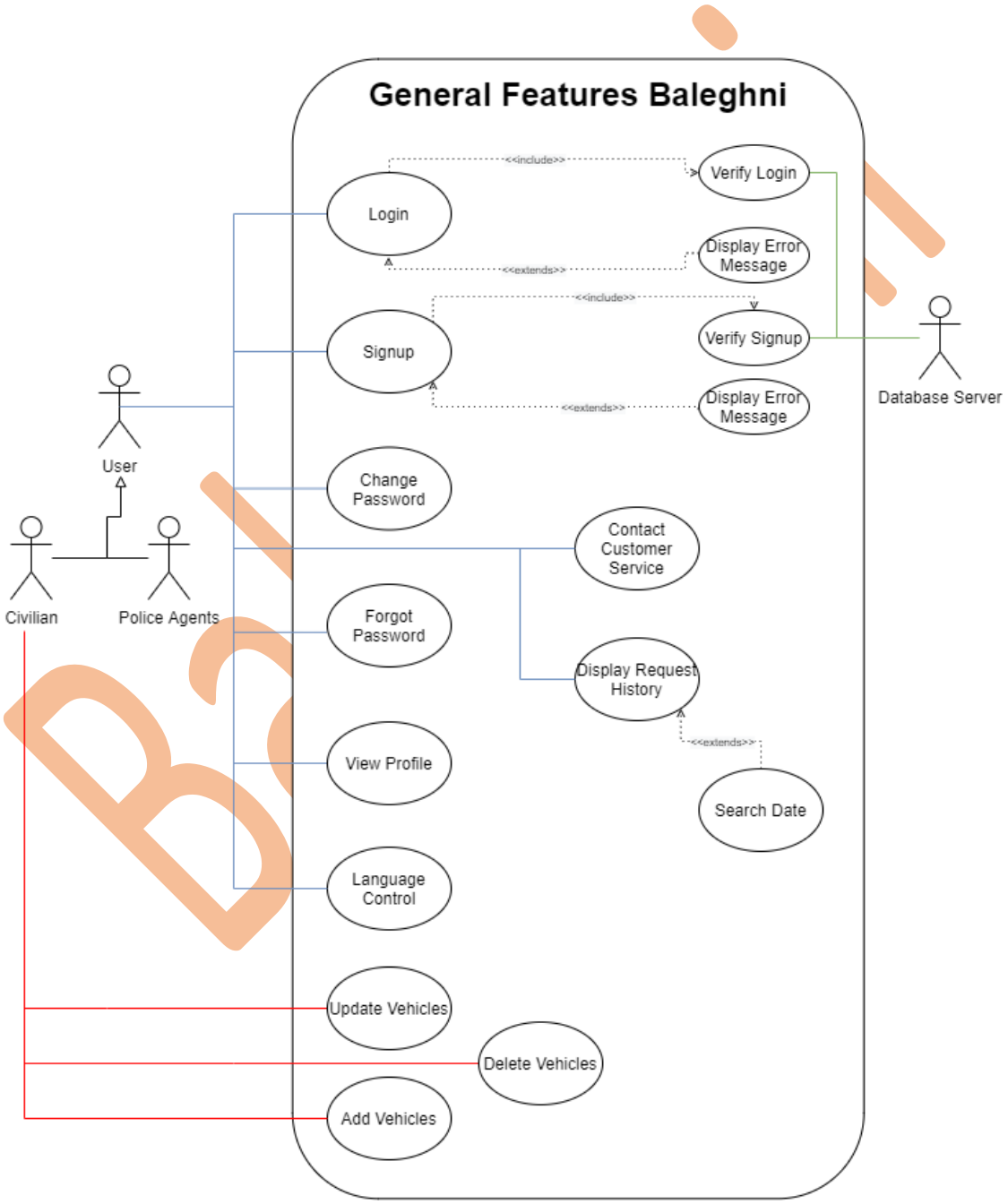


Figure 2: Use Case Diagram for General User Functionalities

Figure 2 represents the user's general functionalities in the system, this includes both the driver and police users.

- Logging in into the system includes verification of the email and password used.
- Signing up includes verification to make sure that the data were not used in another account as well the validity of national ID.
- Changing password while logged in.
- Change their password if they are not logged into the system.
- View their profile.
- Change the language between Arabic and English.
- View their request history.
- Contacting customer service for help.
- Drivers will be able to add, update or remove vehicles from their profile.

1-Use Case: Login

Actors: Driver, Police Agent, Database server
Purpose: For users to login into the system
Overview (Success scenario):
Driver/Police Agent inserts their login information (email and password), the database server checks for validity. On completion, the user gets into the system.
Type: Primary
Cross References: FR2
Typical course of actions:

Actor Actions	System Response
1-User inserts email and password.	2-System authenticates the user.
	3-System Authorizes the users if they are valid.
	4-Successful login, application displays main dashboard to the user.

Alternative: If the user enters wrong email or password, system returns error popup.

Baleghni

2-Use Case: Display Request History

Actors: Driver, Police Agent, Database server
Purpose: For The users to check the history which shows their requests history.
Overview (Success scenario):
The user clicks on Display Request History button, the system should display the history for the user,
the system should display the history based on the date.
Type: Secondary
Cross References: FR23
Typical course of actions:

Actor Actions	System Response
1-User clicks on display request history.	2-The system will display the start and end dates.
3-The user fills the start and end dates.	4-Filter the data from the database server
	5-System displays History information from the database server.
6-User views the information	

Alternative: if the user inserts wrong date (start or end dates) show an error message

Baleghni

4.1.1.2: Use Case Diagram for Driver-Side Interaction in requests

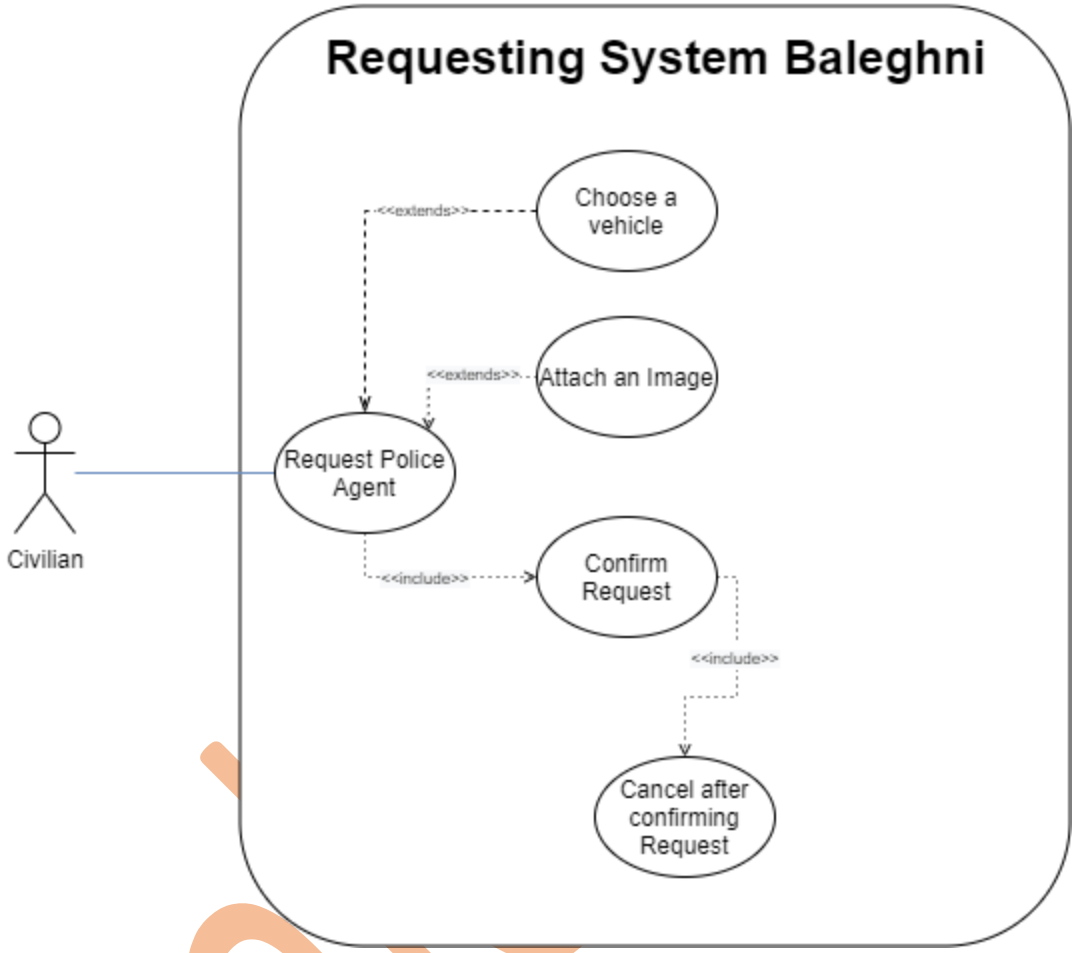


Figure 3: Use Case Diagram for Driver-Side Interaction in requests

Figure 3 represents the Requesting part of Baleghni System that are strictly for Driver users.

- Request a police agent for an accident scene.
- Optionally attach an image of the accident in the request.
- Optionally choose a vehicle.
- Confirm the request.
- Cancel the request before the limited amount passes after previously confirming it.

Use Case: Request Police Agent

Actors: Driver, Police Agent, Database server
Purpose: For the Drivers to send a request for a police agent
Overview (Success scenario):
The Driver will request a police agent, the request will be sent for all nearby police agents, the dispatched police agent will receive the driver's location and information. On completion, the agent will be able to finish the case and the request data will be saved in the database.
Type: Primary
Cross References: FR10
Typical course of actions:

Actor Actions	System Response
1-Driver sends a request.	2-System sends alert to nearby police agents.
	3-System will insert a new case in the database under process.
4-Police agent accepts the request.	5-System sends the driver's location and information to the dispatched police agent, a notification will be sent for the driver that a police agent is on the way.
6-Dispatched police agent will reach the destination and confirm their arrival.	7-System will update the status of the case in the database as under progress.
8-The police agent can end the request once they finish.	9-System will update the request as finished and will save it in the database server.

Baleghni

4.1.1.3: Use Case Diagram for Police/Driver Interaction after request

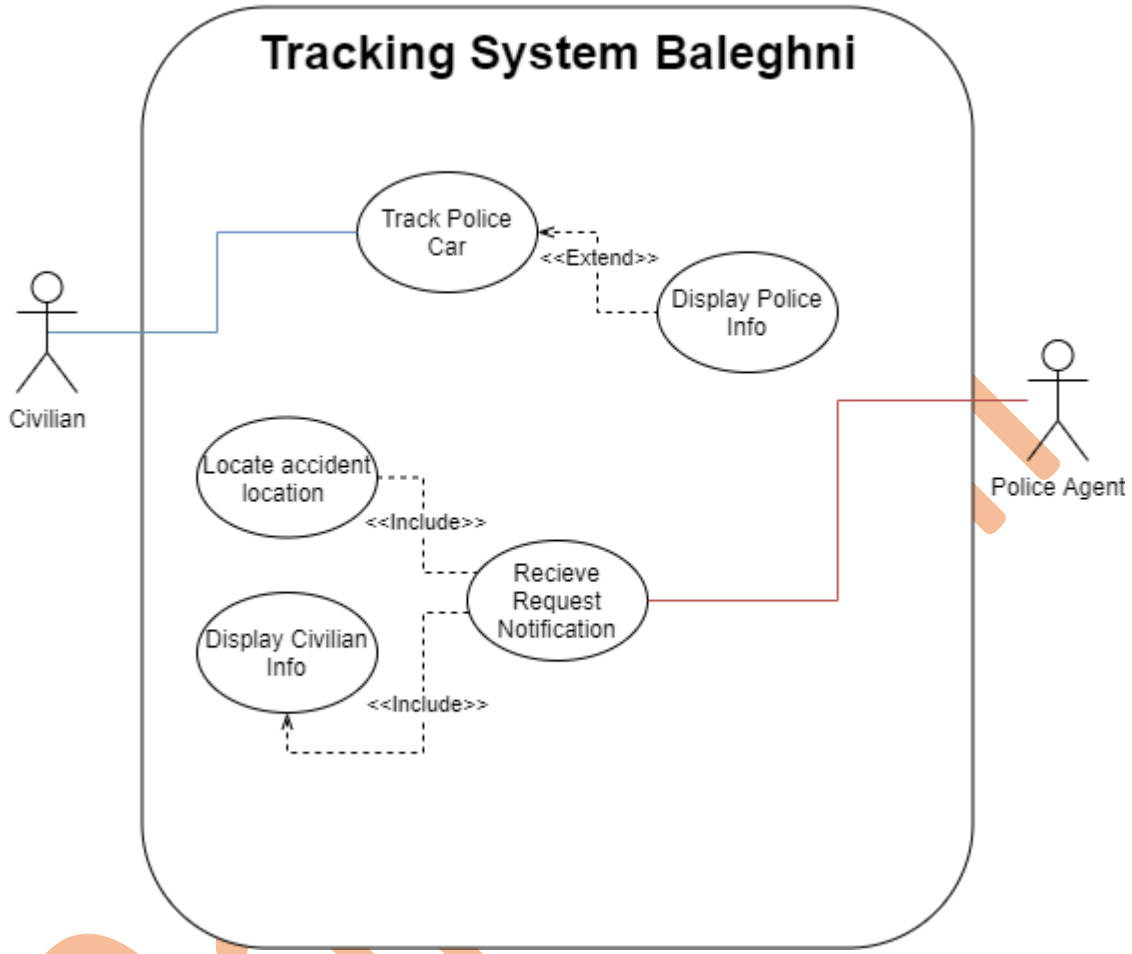


Figure 4: Use Case Diagram for Police/Driver Interaction after request

Figure 4 represents the Receiving part of Baleghni System. Which happens after a request is sent from the driver.

- Police agent receives notification about the request.
- The police will receive the requesting driver's information and the location of the accident.
- The driver will also be able to view the police's information as well as track their location.

4.1.2 Class Diagrams

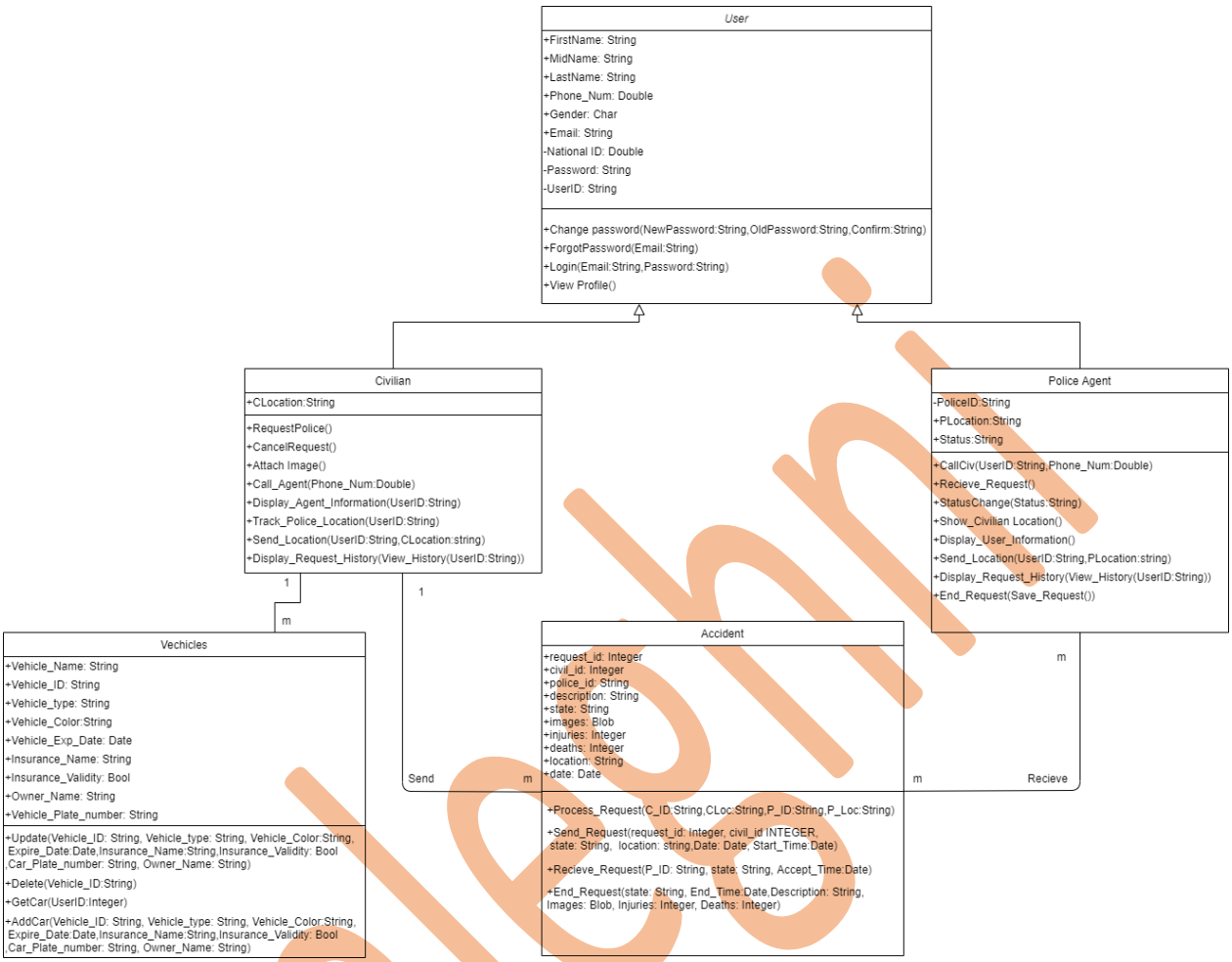


Figure 5: Class Diagram for Baleghni

Figure 5 models the hierarchy of the classes structure that is found in Baleghni system, showing their attributes and operations.

Baleghni has two types of users which are both essential to carry out the main purpose behind the system, which is reporting an accident, they are:

- **Drivers**
- **Police Agents**

Each user can have from 0 up to multiple vehicles linked to their account in which they can choose from when ordering a request.

Both types of users are linked together through the Accident class.

Baleghni

4.1.3 Object Diagram

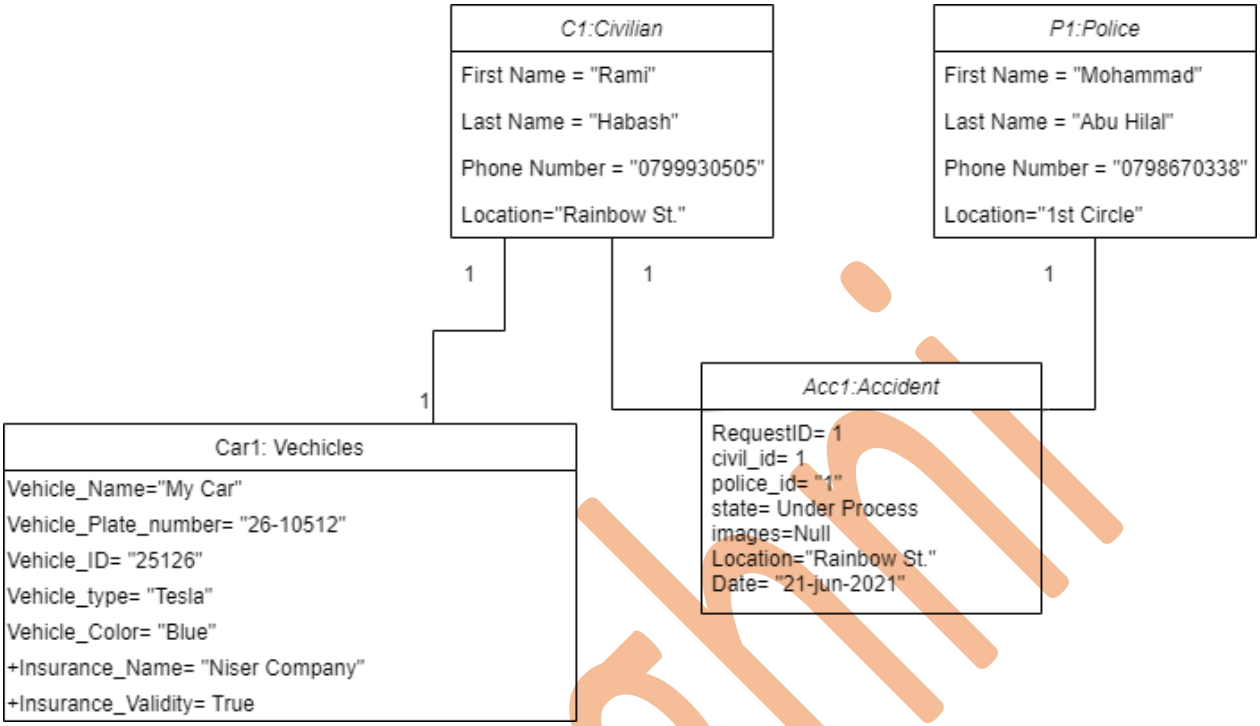


Figure 6: Object Diagram for Baleghni Request

Figure 6 shows a sample of how the classes in our system interact with each other.

The driver object interacts with the police object through the Accident class after choosing the vehicle which is associated in the accident that they had.

Baleghni

4.1.4 Activity Diagrams

4.1.4.1: Activity Diagram for Requesting a Police Agent

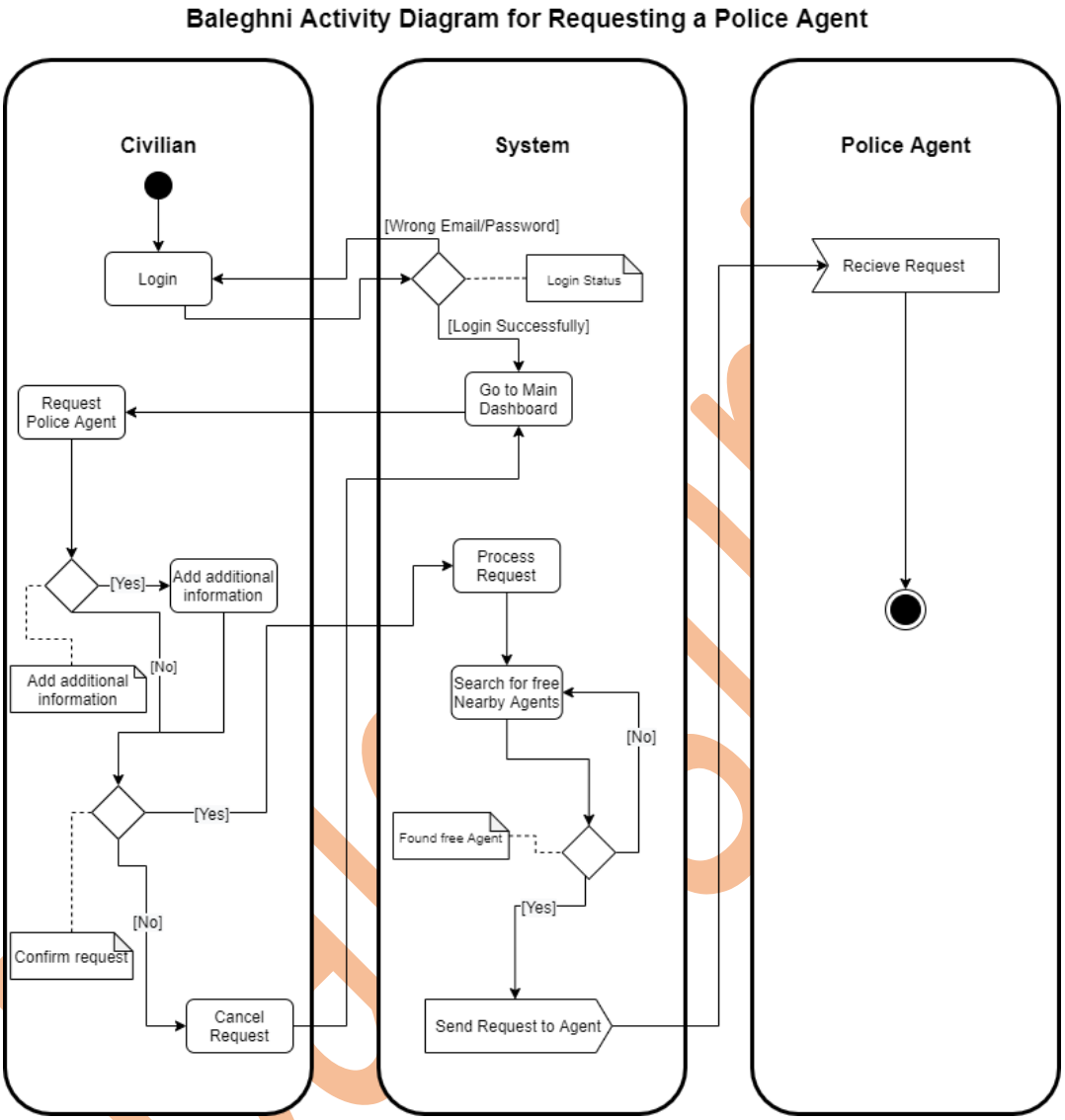


Figure 7: Activity Diagram for Requesting a Police Agent

Figure 7 shows the process and flow of sending a request for a police agent where the driver needs to be logged in, then the main dashboard is displayed. The driver clicks on the button "Request Police Agent'', they can optionally add additional information for the request which consists of (*attaching images*, *choosing vehicle*). After confirming, they will have a limited time to cancel it; if it is cancelled, then they will go back to the main dashboard, else, the system will process the request and search for free nearby agents, the search will keep going until a free agent was found. Once a free agent is found, the system will send the request to the agent found.

4.1.4.2: Activity Diagram After a Police Agent Accepts a Request

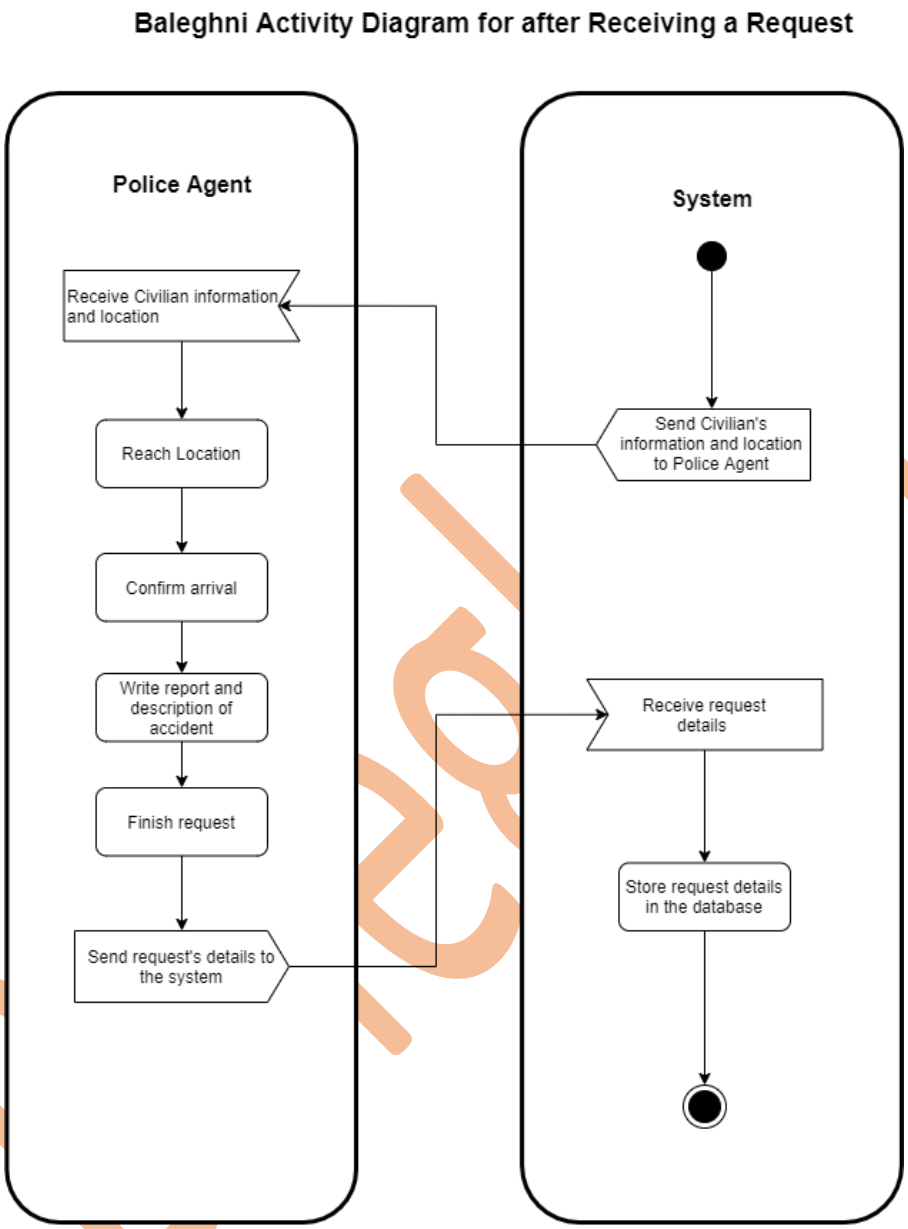


Figure 8: Activity Diagram for after Receiving a Request

Figure 8 shows the processing after the request gets accepted by an agent. The system will send the requesting driver's location and information in which the agent will receive. The agent will reach the location and confirm their arrival. The agent will then write the report and description of the accident before finishing the request. The request details will be sent back and received in the system which will then be stored in the database.

4.1.5 System Architecture Diagrams

4.1.5.1 System Architecture for Request processing

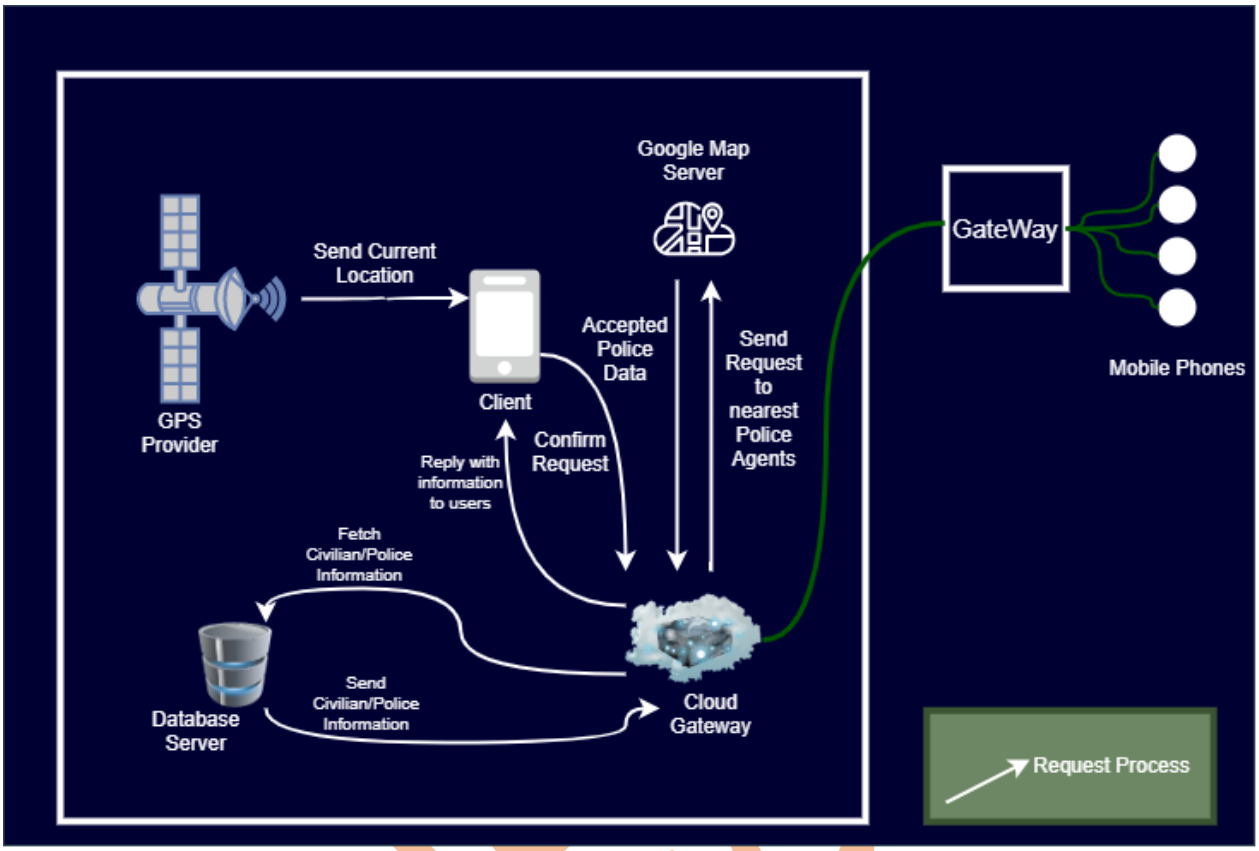


Figure 9: System Architecture for Request processing

Figure 9 represents the processing part of our requests.

To begin with the location of the driver that is provided by the GPS provider will be selected by default by our system, the driver will confirm their request and it will be sent to our cloud server, after that, the system will search for the nearest police agent using the google map server which includes the police agents and their locations, the request will be sent to the nearest agent to the driver's location, afterwards a police agent will accept the request and their ID will be sent back to the cloud which will then be used to fetch their information along with the requesting driver's information from the database. Which will then be sent back to the cloud and back to the users.

4.1.5.2 System Architecture for Data Analysis process

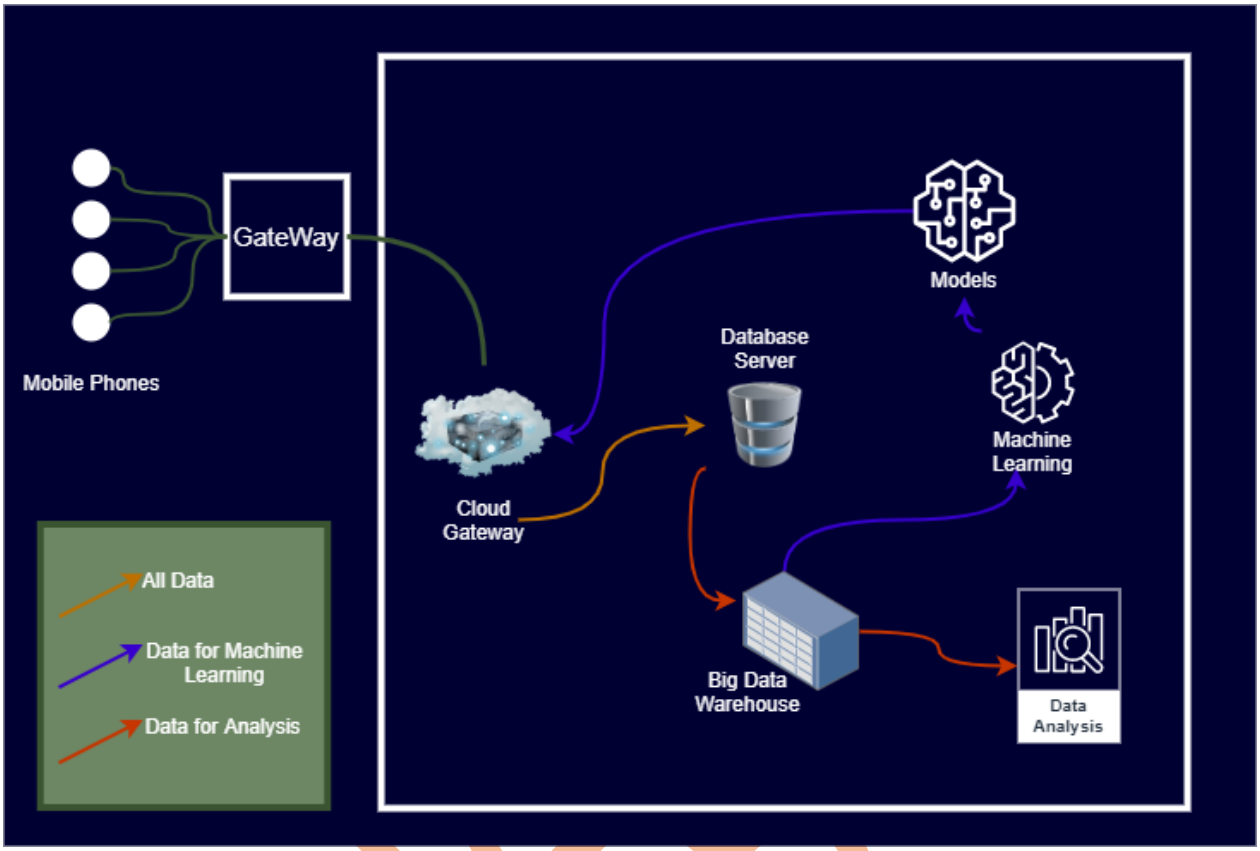


Figure 10: System Architecture for Data analysis process

Figure 10 represents the processing part of data analysis.

In order to get all the data from mobile phones into the database server a gateway will be used, which as a "gate" between the two "networks". Then, the cloud gateway will as a bridge to allow the data to be transferred between the local mobile phones and the remote cloud-based database server which will hold all of the data. The data in the database server will be segmented into two parts: Data to be analyzed and data to be used in machine learning.

The data that are required to be analyzed (accident history, number of injuries, deaths, accidents areas, etc) will be passed to the data analysis center to generate annual full reports related to car accident, number of deaths, injuries, etc. On the other hand, the Machine language data which will help in determining the car accidents causes which will be later on useful in finding solutions, and predicting the upcoming year's number of accidents, injuries, and the areas with the highest number of accidents.

4.1.6 Sequence Diagrams

4.1.6.1 Sequence diagram for Login

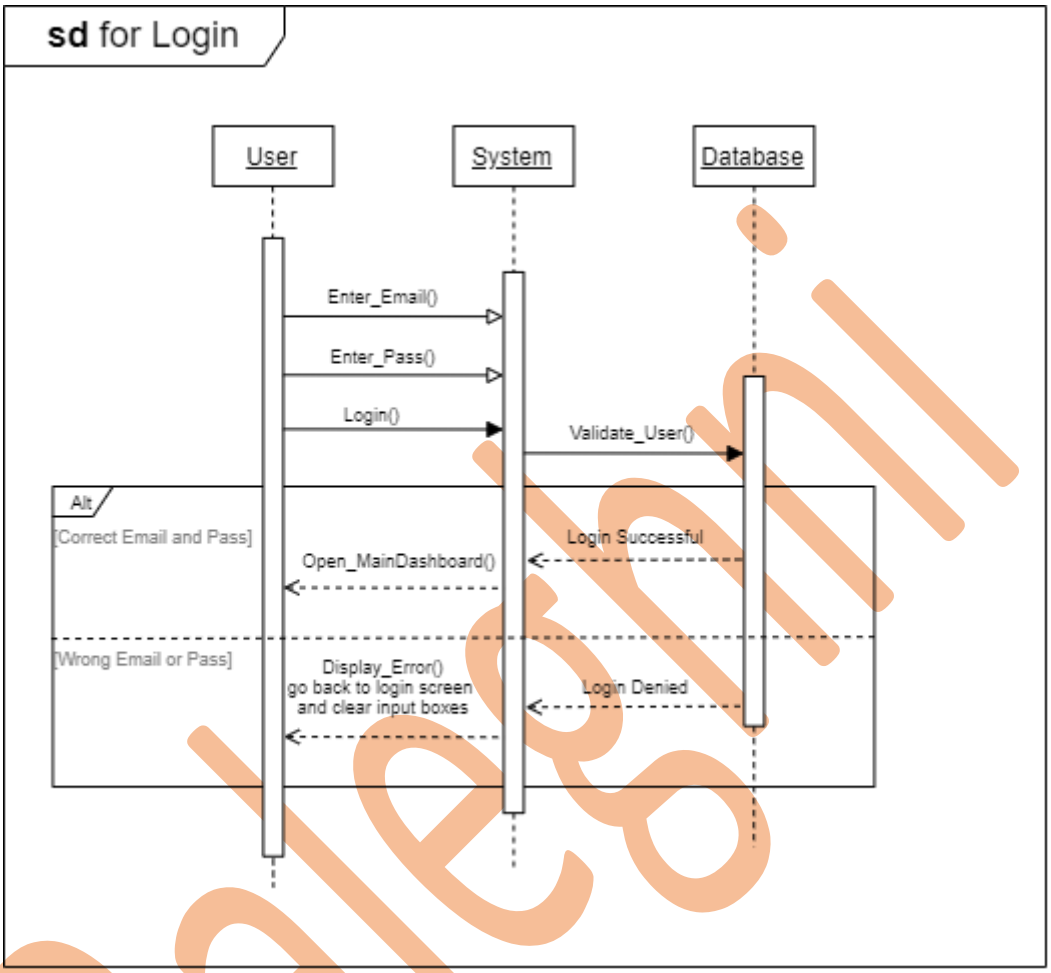


Figure 11: Sequence diagram for Login

Figure 11 represents the logging in part in our system. The user must enter their email and password, followed by clicking on the (login) button, then the system will validate the inserted information from the database, if they are correct, then the system will display the main dashboard, else, if one or both of the input information are not correct, an error message will be displayed, and the system will display the login page again.

Baleghni

4.1.6.2 Sequence diagram for Request processing

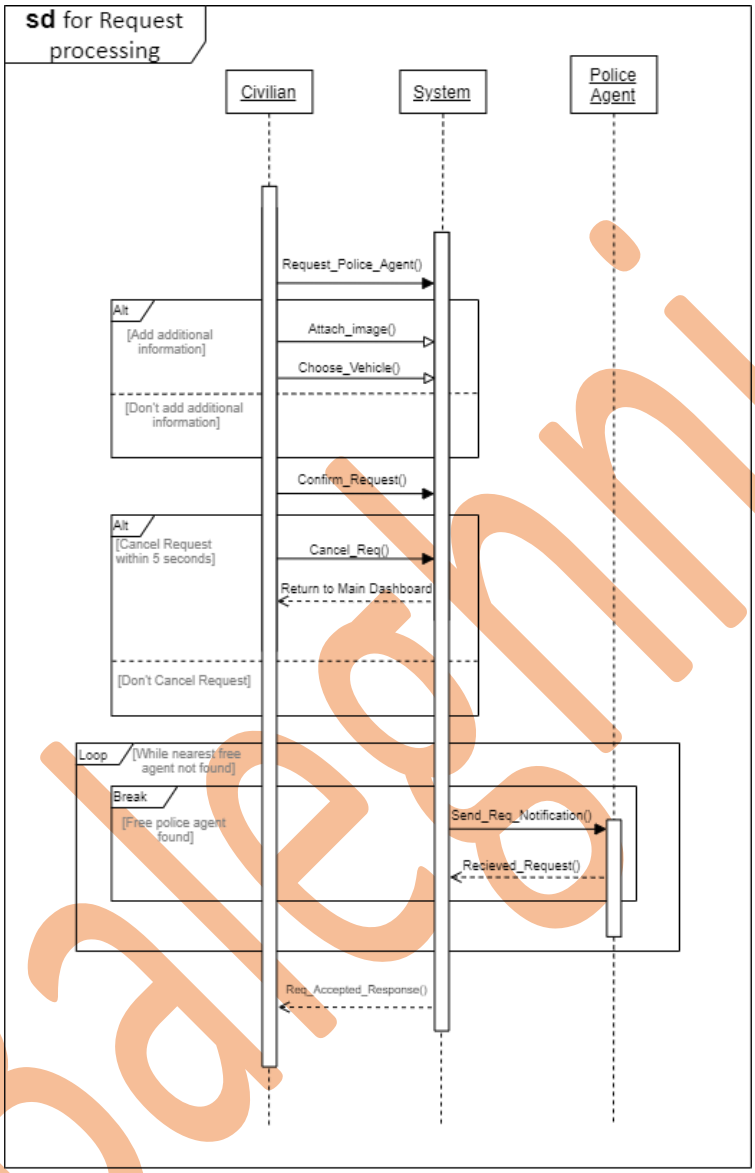


Figure 12: Sequence diagram for Request processing

Figure 12 represents the process of requesting a police agent until one agent accept the request. The user will click on (request police agent) button, there they will have an option add additional information which consists of (*attach images, choose a vehicle*), afterwards, they can click on (confirm request) button, and will have 5 seconds to cancel the request by clicking on (cancel request) button, if they cancel, the system will take them back to the main dashboard, otherwise, the request will proceed and the system will start and keep searching for the nearest free police agent until one is found, a notification will be sent for the found agent, the system will send a notification for the requesting driver showing that their request has been accepted.

Baleghni

4.1.7 State Diagrams

4.1.7.1 State Diagram for Driver Requesting

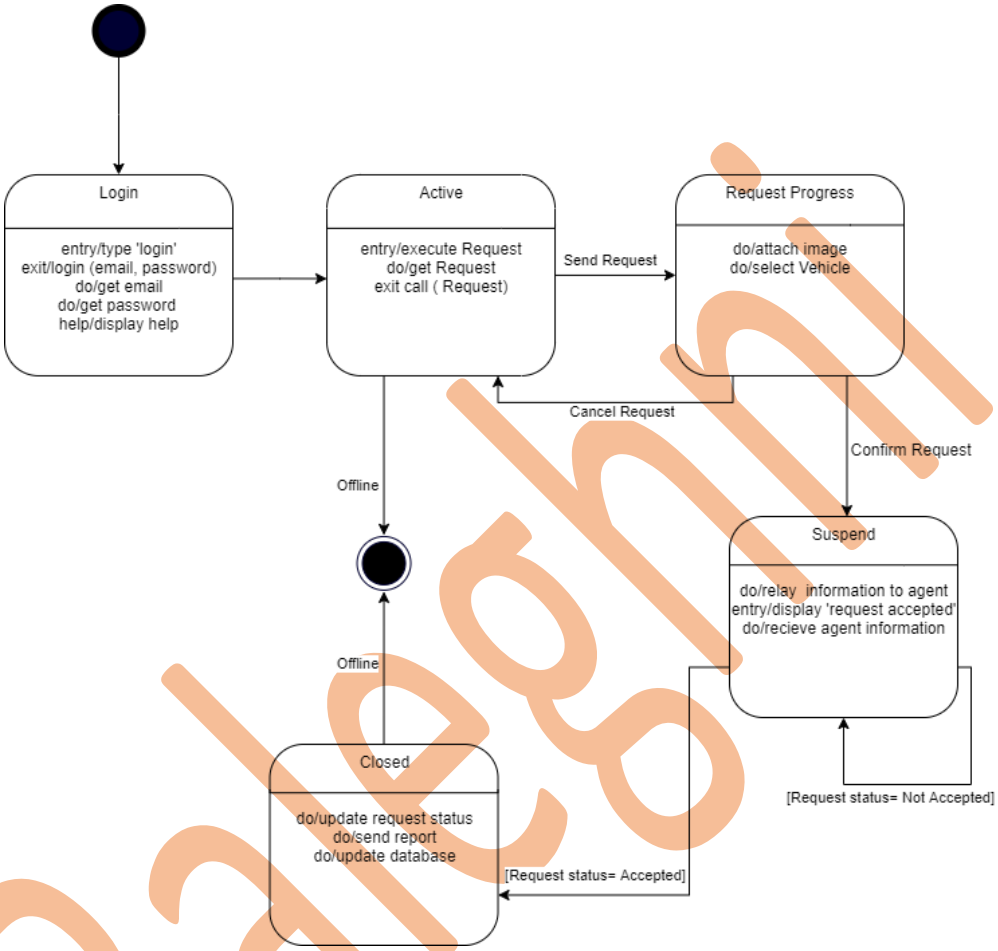


Figure 13: State Diagram for Driver Requesting

Figure 13 represents the process of Requesting, to start with, the requesting driver must be logged in, after being logged in, while active, they'll be able to do a send request option, after that, they can add extra information for the request; (attach an image, select a vehicle), the user can either cancel the request and return to the main page or confirm the request, after confirming, the request will be suspended until an agent is found. The driver will receive a notification that the request has been accepted, they will receive the agent's information and the system will send the driver's information for the agent. When the request ends, the case will be closed, and the agent will send the report and close the request which will update its status and the database.

Baleghni

4.1.7.2 State Diagram for Police side after Request receival

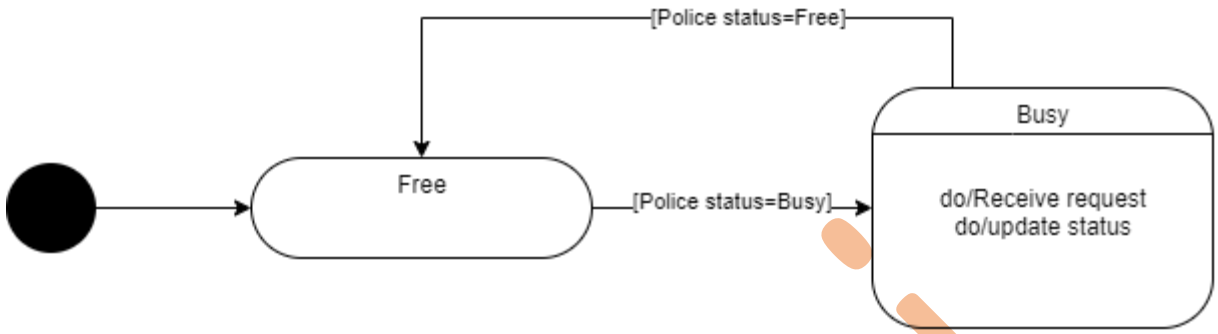


Figure 14: State Diagram for Police side after Request receival

Figure 14 shows the police states while they are active, when they are free, they will receive requests of nearby drivers, after that, the agent's status will change into 'busy', after finishing the request, their status will change back to 'free' and the process will keep repeating while they are active.

Baleghini

4.2 Physical Model Design

4.2.1 UI Design

4.2.1.1 Driver interfaces: [4]

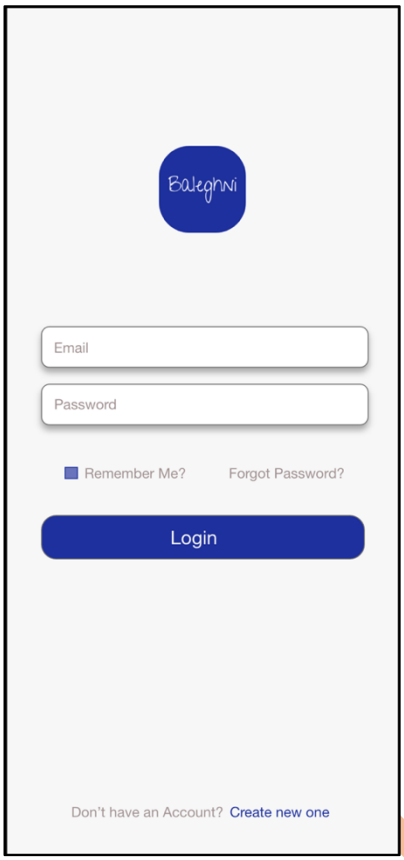


Figure 15: Login Interface

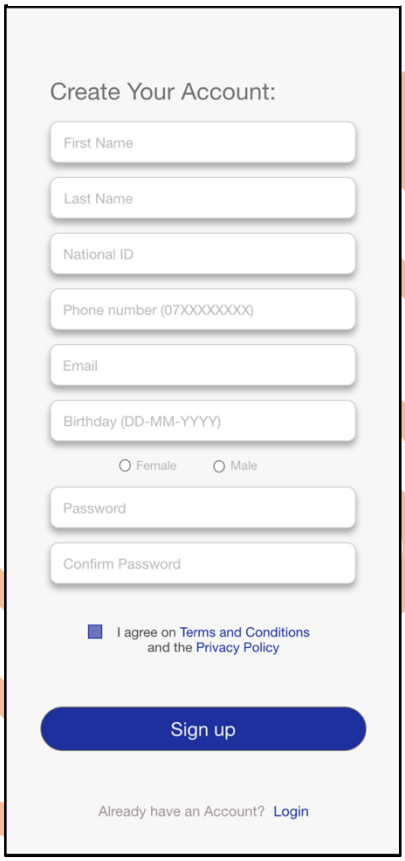


Figure 16: Sign up Page Interface

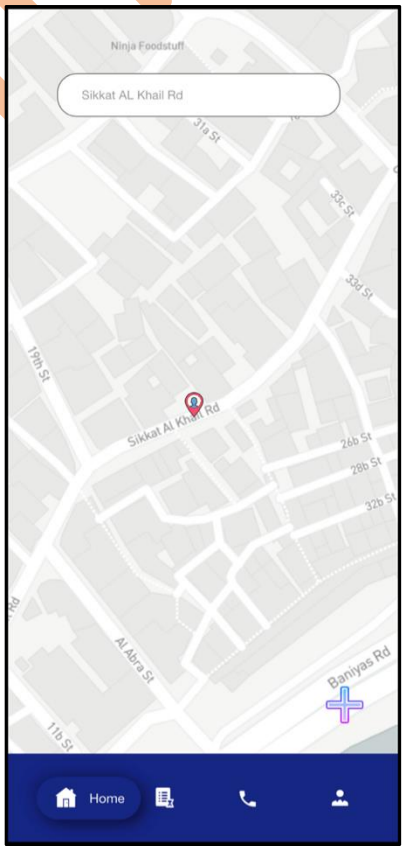


Figure 17: Home Page Interface

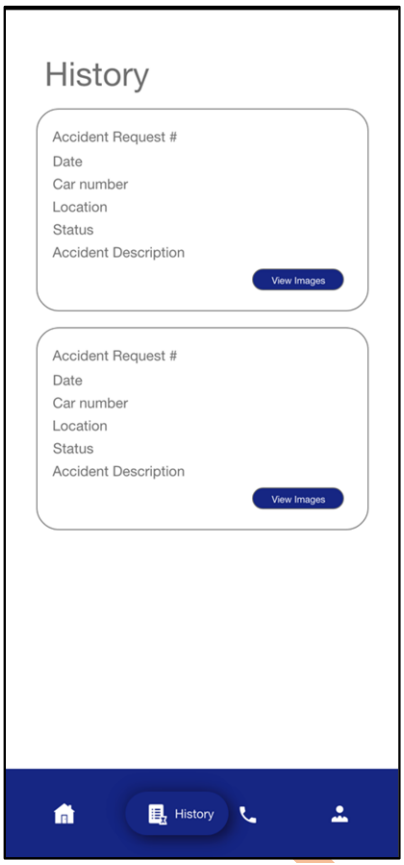


Figure 18: History Page Interface

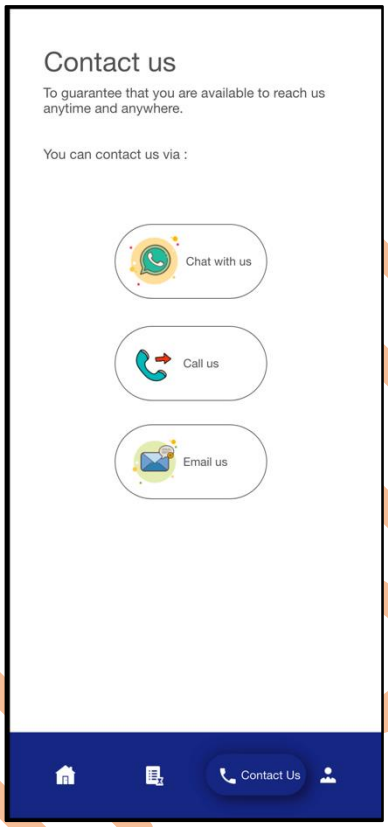


Figure 19: Contact us Page Interface

Figure 20: Driver Profile Page

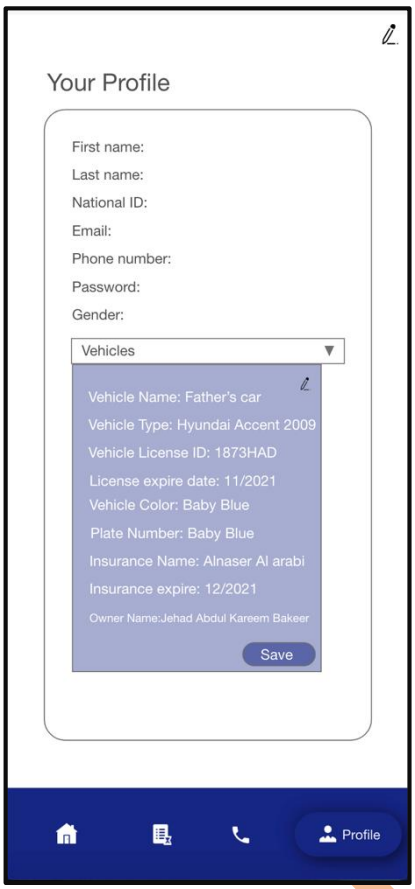


Figure 21: Driver Profile Interface with Edit Vehicle

Baleghni

4.2.1.2 Police Interface [4]

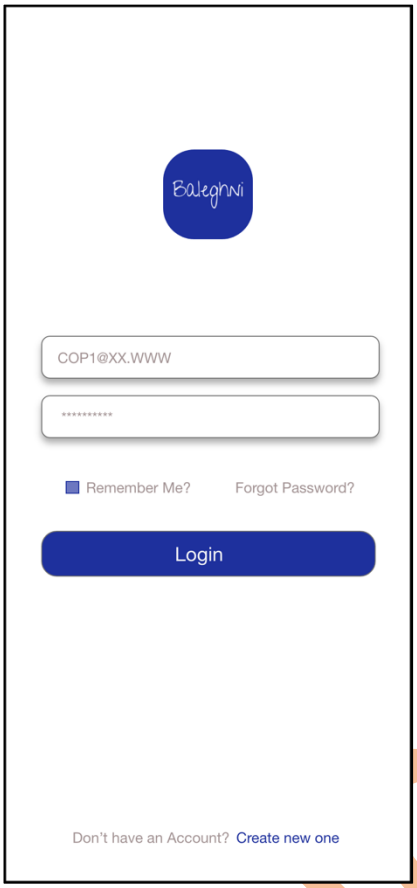


Figure 22: Police Login Page Interface

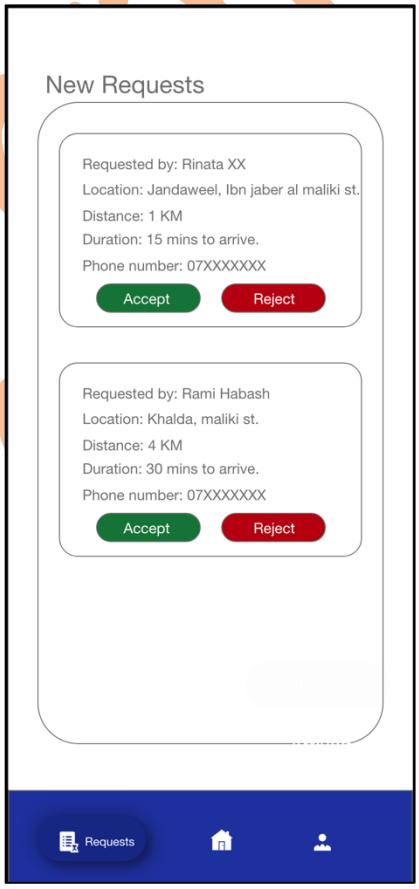


Figure 23: Requests Page Interface

Balegni

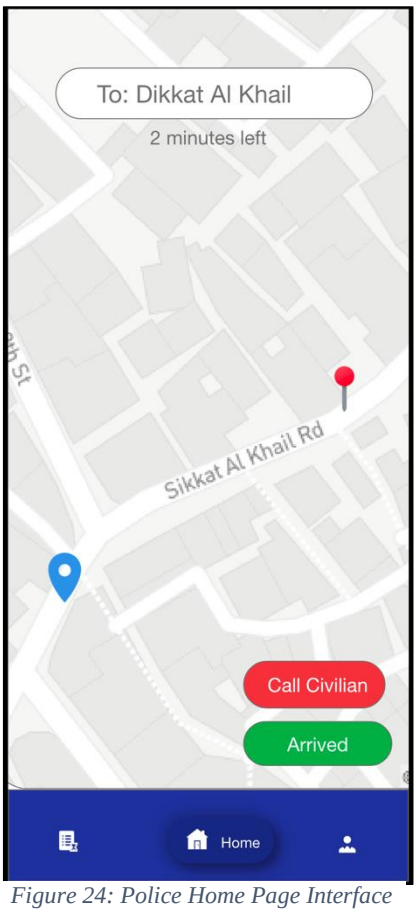


Figure 24: Police Home Page Interface

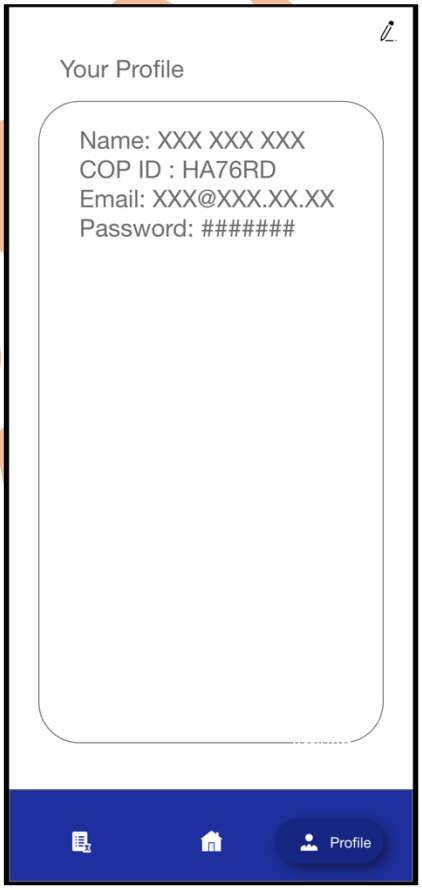


Figure 25: Police Profile Page Interface

Baleegh

4.2.2 Database Design

4.2.2.1 Database Entity Relationship Diagram [5]

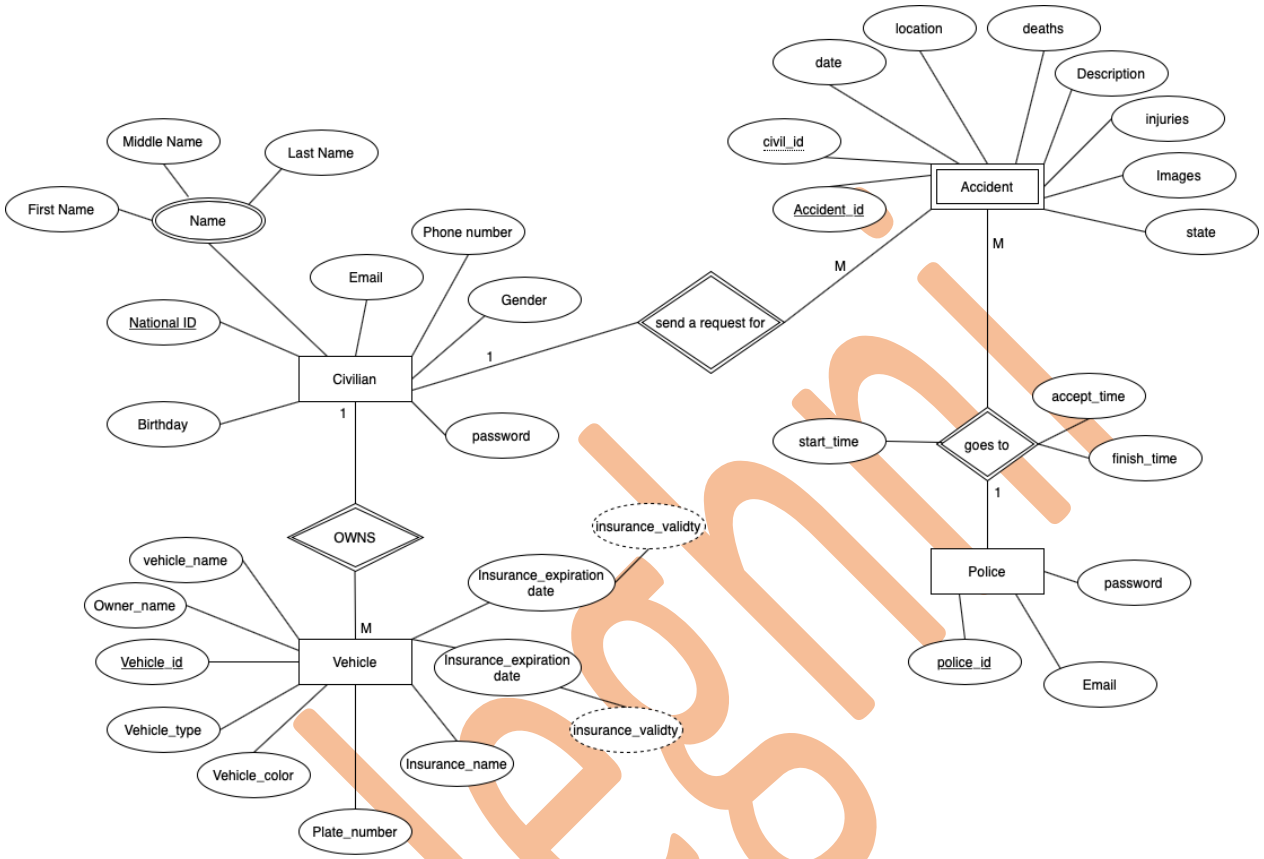


Figure 26: Baleghni Database Entity Relationship Design

Figure 25 shows the ERD which represents the relationships between the entities and their attributes that are stored in our database and used by the Baleghni system.

Baleghni

4.2.2.2 Database Schema

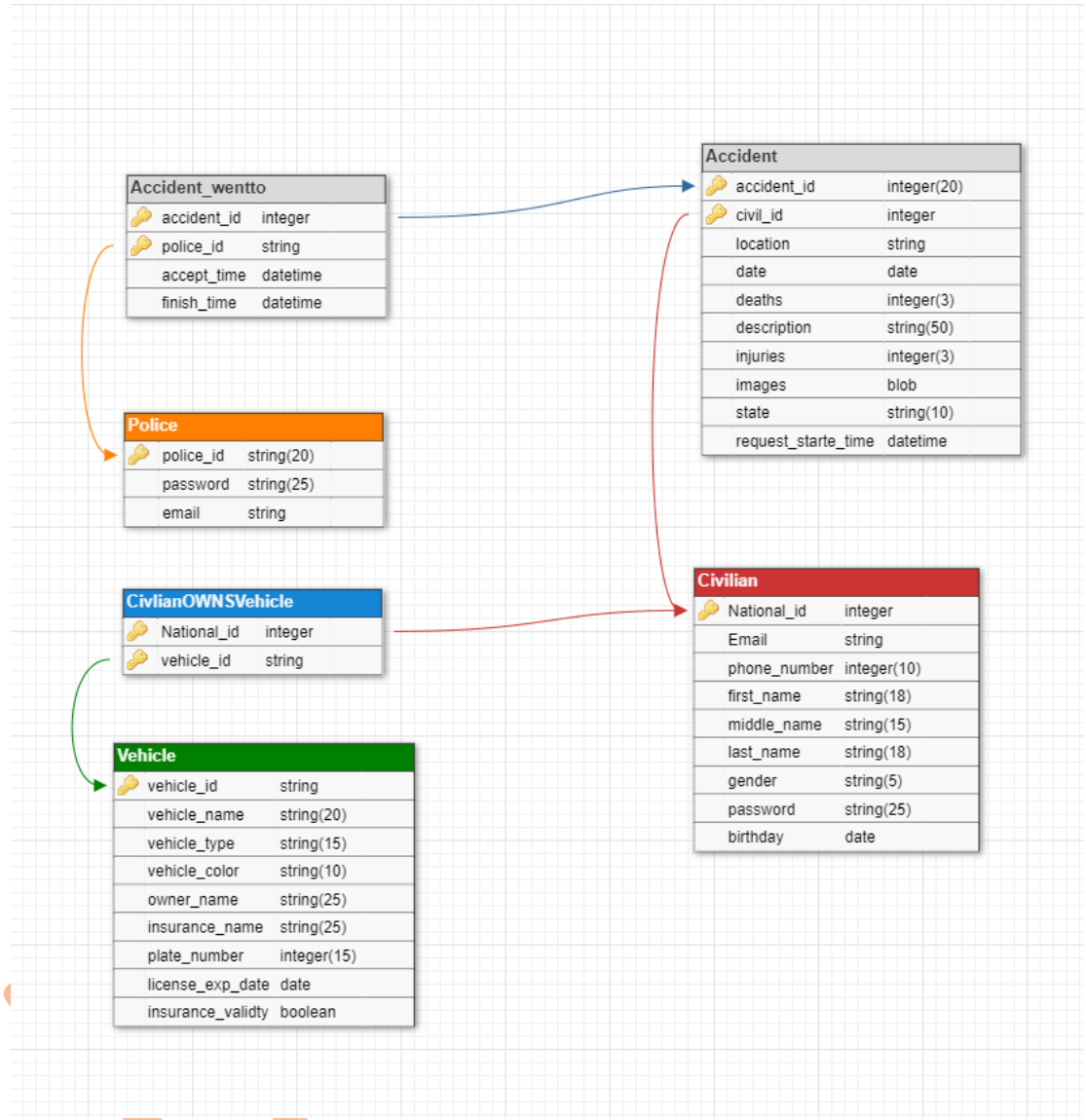


Figure 27: Baleghni Database Schema

Figure 26 shows the database schema of our system in the Third Normal Form (3NF).

4.2.3 Reports Design

4.2.3.1 Vehicle Accident History for an Owner Report

Accident History

Owner national id: #####

Owner name: Omar XXX

Vehicle ID: 235312

Vehicle Type: Car

Vehicle Color: Color

Vehicle brand: BMW

Accident ID	Date	Location	Time	Damages images

Figure 28: Vehicle Accident History for an Owner Report

4.2.3.2 Accident Analytics Per Area Report

Annual Accident Analysis per Area

Area ID	Reason	# of injuries	# of deaths	Damaged Public Property
				Tree

Car accident cause and the high percentage of it : SPEED with 70%

Most vehicle types with accidents = CAR

Current year total number of injuries in the area = ###

Current year total number of deaths in the area = ###

Current year total number of accidents in the area = ###

Predicted number of Injuries in upcoming year = #

Predicted number of deaths in upcoming year = #

Figure 29: Annual Accident Analytics Per Area Report

4.2.3.3 Requests History based on a specific date Report

Accident Requests History on Specific Date

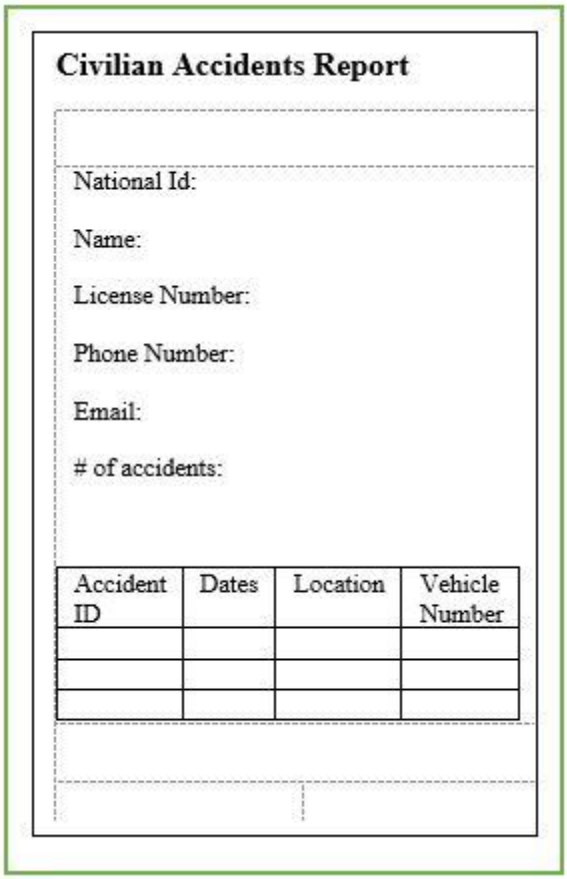
Date:
DD/MM/YYYY

Request ID	Start time	Acceptance Time	End Time	Police ID	Location

Figure 30: Accident Requests History on Specific Date Report

Baleg

4.2.3.4 Driver Accidents Report



Civilian Accidents Report

National Id:

Name:

License Number:

Phone Number:

Email:

# of accidents:

Accident ID	Dates	Location	Vehicle Number

Figure 31: Driver Accidents Report

4.2.3.5 Upcoming Year Car Accidents Prediction Report

Upcoming Year Car Accidents Prediction Report

Predicted total # of accidents in upcoming year = #

Predicted total # of BUS accidents in upcoming year = #

Predicted number of Injuries in upcoming year = #

Predicted number of deaths in upcoming year = #

Predicted area with highest car-accident percentage: AREA1

Predicted car-accident cause with highest percentage:

ALCOHOL

Predicted vehicle model with highest car-accident percentage: KIA SEVIA

Figure 32: Upcoming Year Car Accidents Prediction Report

4.3 Test Plan

Table 14: Test Plan Table

Test ID	Function tested	Function ID	Input	Expected Output	Actual Output	Result
TC1	Sign up	FR1	User enters valid national id, name, phone number in form of (07XXXXXXX), and valid email.	Successful Sign up.	Sign up has succeeded.	Successful
TC2	Sign up	FR1	User enters invalid national id or name, or invalid phone number not in form of (07XXXXXXX) or invalid email.	Unsuccessful sign up, error message – Enter a valid national id/name/ phone number/email.	Sign up has failed to complete.	Successful
TC3	Login (for Driver and Police)	FR2	Existing username. Correct password.	Successful Login.	User login was successfully.	Successful
TC4	Forgot Password	FR3	Valid email/phone number, and valid password.	Successful changing password.	Changing password was successful.	Successful
TC5	Add vehicle	FR6	User enters a valid vehicle id, plate number, owner name.	Successful adding vehicle into database.	Adding Vehicle information into database is successful.	Successful
TC6	Update Vehicle	FR7	Updating existing vehicle information with invalid vehicle id, or plate number or owner name.	Unsuccessful vehicle's update with error message.	Updating vehicle information in the database is unsuccessful.	Successful
TC7	Request a police agent.	FR10	User requests a police agent for an accident and confirms the request.	Request is obtained successfully and searching for a police officer is in progress.	Request is obtained successfully and placed within the database in order for the system to send a nearby police.	Successful
TC8	Driver cancels request before confirming.	FR14	User clicks on cancel button.	Cancel request is obtained successfully.	Cancel request is obtained successfully, and the system will process this cancelation by removing it from database and reassigning police into free statue.	Successful

TC9	Attach an image for the accident.	FR11	User (Driver /Police) attaches a jpg, jpeg image type.	Successful adding Image.	Attached image is successfully added.	Successful
TC10	Attach an image for the accident.	FR11	User (Driver /Police) attaches an image with other types of jpg and jpeg.	Unsuccessful adding Image, error message appears.	Attached image is unsuccessful added.	Successful
TC11	Display request history	FR23	User enters an valid date form.	Successful history display request and is obtained.	Successful history display.	Successful
TC12	Search Accident History by date.	FR24	User enters an invalid date. For example: 22/2/2200 or a string such as HAKOONA	Unsuccessful history display request, error message appears.	Unsuccessful history display.	Successful

Baleghni

Chapter 5
Implementation

5.1 Programming Languages, Tools and APIs

Several programming languages were used to achieve our main goal in this project. And these programming languages were chosen for being the best and most appropriate for best functionality.

To begin with ,Visual Studio Code was used as the editor due to being lightweight as well as having many extensions that can be used to help improve our development experience, along with connecting it with GitHub which was a big part of a team working together.

5.1.1 Front End

Our project divides into a mobile application, and a website. Therefore, for the application, we picked React Native as our main language after seeing a lot of reviews stating that it one of the top languages currently available. React Native has many interesting features such as unified coding for both IOS and Android, instead of being separate. And it is much easier for testing as it includes real-time data fetching and generating UI based on any code changes. On the other hand, for the website, basic HTML/CSS/JS was used to fully develop what was required along with a bootstrap template.

5.1.2 Back End

Our project's main Back-end tool used was Firebase, a Back-end-as-a-service (BAAS) provided by Google which offers a variety of tools and services, such as user authentication, databases, a server that can be deployed and many other useful time saving services, which improves the development of both mobile and web applications.

For the APIs that were used.

Node.js was the base of our development, as it was used to access 'npm' and completely manage the packages and their dependencies that were required throughout the whole development period.

Expo API is a framework for universal React applications, it contains many packages and tools that can be used to improve the performance of our application, mainly, its most important feature that we used for testing is accessing through multiple devices at the same time, in world-wide-distance manner.

Google Maps API integrated a copy of google map into our Web application

Maps JavaScript API was used to help customize the map that the was created in our Web application.

Baleghi

Cloud Functions API is a serverless framework that allowed us to run backend code in response to events generated by Firestore triggers and requests. It either stores JavaScript or Typescript code; we chose Typescript since it is more complex and advanced, and it can all be saved in a controlled environment within the project's cloud storage.

Expo's Push API handles notification delivery to clients using Firebase Cloud Messaging, which we specified to activate based on the condition of their current request.

5.2 Overview

This is a simplified general overview of the classes used in our applications, estimated number of lines and classes.

Class Name	Class Description
react	Library for building general User Interface.
react-native	Library for building detailed User Interface.
react-native-maps	To view a customizable copy of google map in our application.
react-native-btr	Create navigation bottom panel.
firebase	Firebase Authentication Services.
@expo/vector-icons	Handles customized icon usages.
expo-permissions	Handles user-notification-permissions.
expo-notifications	Handles received request notifications.
expo-location	Accurate Location finder. Handles user-location-permissions.
expo-image-picker	Handles user-media-permissions. Handles opening gallery
HistoryData	Handles receiving request history for user.
RequestDetails	Handles showing details for history requests.
LoginScreen	Handles Email/Password Login page for Users
RegisterScreen	Handles Creating New Account Page for Drivers
ForgotPasswordScreen	Handles Resetting Password Screen for Users
HomeScreen	Handles Main page functionalities for Driver
PoliceHomeScreen	Handles Main page functionalities for Police
History	Handles Request History Page for Users
ProfileScreen	Handles Profile details page for Driver
PoliceProfileScreen	Handles Profile details page for Police
AdminLogin	Handles Admin Email/Password Login Page
Map	Handles Tracking both Requests and Active Police Page
RegisterPolice	Handles Register Police by Admin Page
Total Hand-Written Lines of Code	5099
Total Hand-Written Number of Classes	17+

Baleani

5.3 Main Algorithms and Coding Conventions

5.3.1 General User

5.3.1.1 Sign in using Email/Password

```
<Button
  title="Login"
  disabled={!canLogin}
  onPress={async () => {
    try {
      firebase
        .auth()
        .signInWithEmailAndPassword(Email, password)
        .then(async (userCredential) => {
          let snapshot: any = await firebase
            .firestore()
            .collection("Users")
            .doc(userCredential.user?.uid)
            .get();
          var checktype = snapshot.data().UserType;
          if (checktype != "Police") {
            if (userCredential.user?.emailVerified) {
              // Signed in and verified

              var user = userCredential.user;
              var loggedinuserid = user?.uid;
              await LoginCheck(loggedinuserid);
            } else {
              // Not verified
              firebase.auth().signOut();
              alert("Please verify your account through your email!");
            }
          } else {
            var user1 = userCredential.user;
            var loggedinuserid1 = user1?.uid;
            await LoginCheck(loggedinuserid1);
          }
        })
        .catch((e) => {
          alert(e);
        });
    } catch (e) {
      alert(e);
    }
  }}
/>
```

5.3.2 Driver User

5.3.2.1 Register Driver

```
<Button
  title="Signup"
  disabled={!canSignUp}
  onPress={async () => {
    try {
      var val1 = validateDataEntry(Pnumber, NationalId); //checks format validity of natioanl id,, phone number..
      var val2 = -1;
      if (val1 == 7) {
        //if correct format, checks if entries already used..
        let val = await AllowedDataEntry(Pnumber, NationalId);
        val2 = val;
      }

      if (val1 == 7 && val2 == 0) {
        firebase
          .auth()
          .createUserWithEmailAndPassword(Email, password)
          .then((UserCreds: any) => {
            if (UserCreds) {
              UserCreds.user.sendEmailVerification().then(() => {
                var user = UserCreds.user;
                var userID = user?.uid;
                // adding the driver(civilian) to database containing all users
                const usersRef = firebase
                  .firestore()
                  .collection("Users");
                usersRef
                  .doc(userID)
                  .set({
                    UserID: userID,
                    UserType: "Driver",
                    Created_At:
                      firebase.firestore.FieldValue.serverTimestamp(),
                    NationalID: NationalId,
                    First_Name: Fname,
                    Last_Name: Lname,
                    Phone_Number: Pnumber,
                    Email: Email,
                    Gender: Gender,
                    BirthDate: BirthDate,
                    Current_Request: "None",
                  })
                  .then(() => {})
                  .catch((error) => {
                    alert(error);
                  });
```

```
                alert([
                  // <Feather name="check-circle" size={24} color="green" />,
                  "Congratulations,Your account has been successfully created. Please verify it through your email!",
                ]);
                props.navigation.reset({
                  index: 0,
                  routes: [{ name: "Login" }],
                });
              });
            }
          })
          .catch((e) => {
            alert(e);
          });
      } else {
      }
    } catch (e) {
      alert(e);
    }
  }}
/>
```

5.3.2.2 Driver Request Police

```
const onReportAnAccident = React.useCallback(async () => {
  let lat = 0;
  let long = 0;


  if (customUserMapRegion) {
    // The user has moved the map manually, use their custom location
    lat = customUserMapRegion.latitude;
    long = customUserMapRegion.longitude;

  } else if (userGpsLocation !== null) {
    // The user didn't move the map, use the GPS location
    lat = userGpsLocation.coords.latitude;
    long = userGpsLocation.coords.longitude;

  }
  // Step 2: Send the request
  ProcessCheckup(LoggedUser, lat, long);
  setHasActiveReport(true);

}, []);
```

```
async function ProcessCheckup(LoggedUser: any, lat: any, long: any) {
  try {

      alert("Sending request in 5 seconds");
      sending = true;
      sendreqtimeout = setTimeout(function () {
        SendRequest(LoggedUser, lat, long);
      }, 5000);

  } catch (e) {
    alert(e);
  }
}
```

```
async function SendRequest(LoggedUser: any, lat: any, long: any) {
  try {

    setAccidentID("None");
    const Requestref = await firebase
      .firestore()
      .collection("Sent_Requests")
      .doc();
    setAccidentID(Requestref.id);

    Requestref.set({
      AccidentID: Requestref.id,
      Driver_User_ID: LoggedUser,
      Latitude: lat,
      Longitude: long,
      Request_Send_Time: firebase.firestore.FieldValue.serverTimestamp(),
      Status: "Pending",
    }).catch((error) => {
      alert(error);
    });

    await firebase.firestore().collection("Users").doc(LoggedUser).update({
      Current_Request: Requestref.id,
    });
    sent = true;

    alert("Your request has been sent, searching for police agent.");
  } catch (e) {
    alert(e);

  }

}
```

5.3.2.3 Driver Cancel Request

```
}
async function CancelRequest(LoggedUser: any) {
  try {
    getdata();
    let snapshot: any = await firebase
      .firestore()
      .collection("Sent_Requests")
      .doc(AccidentID)
      .get();

    if (snapshot.data() != null) sent = true;
    if (sent == true && snapshot.data()) {
      let refFinished = await firebase
        .firestore()
        .collection("Finished_Requests");

      refFinished.doc(AccidentID).set({
        AccidentID: AccidentID,
        Driver_User_ID: LoggedUser,
        Request_Send_Time: snapshot.data().Request_Send_Time,
        Status: "Cancelled",
        Latitude: snapshot.data().Latitude,
        Longitude: snapshot.data().Longitude,
        Request_Finish_Time: firebase.firestore.FieldValue.serverTimestamp(),
      });

      if (
        snapshot.data().Status == "Accepted" ||
        snapshot.data().Status == "Arrived"
      ) {
        //if request was accepted by police

        refFinished.doc(AccidentID).update({
          Police_User_ID: snapshot.data().Police_User_ID,
          Request_Accept_Time: snapshot.data().Request_Accept_Time,
        });
        if (snapshot.data().Status == "Arrived") {
          refFinished.doc(AccidentID).update({
            Police_User_ID: snapshot.data().Police_User_ID,
            Request_Arrival_Time: snapshot.data().Request_Arrival_Time,
          });
        }
```

```
        await firebase
          .firestore()
          .collection("Users")
          .doc(snapshot.data().Police_User_ID)
          .update({
            Current_Request: "None",
            Status2: "Free",
          });
      } else if (snapshot.data().Status == "Pending") {
        refFinished.doc(AccidentID).update({
          Status: "Cancelled",
        });
      }
      await firebase
        .firestore()
        .collection("Sent_Requests")
        .doc(AccidentID)
        .delete();
      await firebase.firestore().collection("Users").doc(LoggedUser).update({
        Current_Request: "None",
      });

      sent = false;
      alert("Request deleted after being sent");
    } else if (sent == false && sending == true) {
      //if request not sent, just remove the timer to send
      if (sendreqtimeout) {
        clearTimeout(sendreqtimeout);
        alert("Your request has been cancelled.");
        sent = false;
      }
    } else if (sent == false && snapshot.data() == null) {
      alert("You dont have any request");
    }
    sending = false;
  } catch (e) {
    alert(e);
  }
  getdata();
}
```

5.3.3 Police User

5.3.3.1 Police Finish Request

```
async function FinishRequest() {
  getdata();
  alert(AccidentID);
  let snapshot: any = await firebase
    .firestore()
    .collection("Sent_Requests")
    .doc(AccidentID)
    .get();

  if (snapshot.data()) {
    let refFinished = await firebase
      .firestore()
      .collection("Finished_Requests");

    refFinished.doc(AccidentID).set({
      AccidentID: AccidentID,
      Driver_User_ID: snapshot.data().Driver_User_ID,
      Status: "Finished",
      Request_Send_Time: snapshot.data().Request_Send_Time,
      Request_Accept_Time: snapshot.data().Request_Accept_Time,
      Latitude: snapshot.data().Latitude,
      Longitude: snapshot.data().Longitude,
    });

    refFinished.doc(AccidentID).update({
      Police_User_ID: LoggedUser,
      Request_Finish_Time: firebase.firestore.FieldValue.serverTimestamp(),
    });
    await firebase
      .firestore()
      .collection("Users")
      .doc(snapshot.data().Driver_User_ID)
      .update({
        Current_Request: "None",
      });
    await firebase.firestore().collection("Users").doc(LoggedUser).update({
      Current_Request: "None",
    });
    if(description!=""){
      refFinished.doc(AccidentID).update({
        AccidentDescription:description,
      })
    }else{
      refFinished.doc(AccidentID).update({
        AccidentDescription:"---",
      })
    }
```

```
    if(image!=""){
      refFinished.doc(AccidentID).update({
        AccidentImages:image
      })

    }
    await firebase
      .firestore()
      .collection("Sent_Requests")
      .doc(AccidentID)
      .delete();
  }
  //go back to normal screen
}
```

5.3.3.2 Police Change Status

```
function changestatus() {
  setActiveStatus(!activeStatus);
  if (!activeStatus) {
    firebase.firestore().collection("Users").doc(LoggedUser).update({
      Status: "Online",
      Latitude: location?.coords.latitude,
      Longitude: location?.coords.longitude,
    });
    setTimeout(() => {
      StartGeoLocation();
    }, 2000);
  } else {
    EndGeoLocation();

    firebase.firestore().collection("Users").doc(LoggedUser).update({
      Status: "Offline",
      Latitude: "---",
      Longitude: "---",
    });
  }
}
```

5.3.4 Dispatcher

5.3.4.1 Assign Police Agent to Request

```
async function assign() {
  try {
    var ReqSel = document.getElementById("AssignRequest").value;
    var PolSel = document.getElementById("AssignPolice").value;

    if (ReqSel != "Select a Request" && PolSel != "Select an Agent") {
      var ReqData = document.getElementById(ReqSel).cells;
      var PolData = document.getElementById(PolSel).cells;

      var DrivID = ReqData[0].innerText.split("ID:- ");
      var PolID = PolData[0].innerText.split("ID:- ");

      alert(
        "Request " +
          ReqData[2].innerText +
          " has been accepted by police " +
          PolData[2].innerText
      );
      const Policeref = await firebase
        .firestore()
        .collection("Users")
        .doc(PolID[1]);
      const PolicerefData = (await Policeref.get()).data();
      const Reqref = await (
        await firebase.firestore().collection("Users").doc(DrivID[1]).get()
      ).data().Current_Request;

      const Accidentref = await firebase
        .firestore()
        .collection("Sent_Requests")
        .doc(Reqref)
        .update({
          Police_User_ID: PolicerefData.UserID,
          Request_Accept_Time: firebase.firestore.FieldValue.serverTimestamp(),
          Status: "Accepted",
        });
      Policeref.update({
        Current_Request: Reqref,
        Status2: "Busy",
      });
    } else {
      alert("Please Assign Valid Units");
    }
  } catch (e) {
    alert(e);
  }
}
```

5.4 Features

Table 15 lists all the featured that were ONLY mentioned in chapter 2 and whether they were implemented or not.

Table 15: Features Table

Functional Requirement ID	Feature(s)	Implemented or Not
FR1	Sign Up	Implemented
FR2	Login using email and password	Implemented
FR3	Forgot password	Implemented
FR4	View profile	Implemented
FR5	Edit profile	Not Implemented
FR6-FR9	Editing vehicle details.	Not Implemented
FR10	Request a police agent	Implemented
FR11	Attach an Image upon request	Not Implemented
FR12	Choose vehicle upon request	Not Implemented
FR13	Driver confirms request	Implemented
FR14	Driver cancels request before confirming.	Implemented
FR15	Driver cancels request after confirming.	Implemented
FR16	Display notification for police agents upon requests arrival	Implemented
FR17	Police agent status change	Implemented
FR18	Displaying a popup for driver that his request has been accepted	Implemented
FR19	Display Driver/agent information	Implemented
FR20	Map tracking	Implemented
FR21	Language control	Not Implemented
FR22	Display request history	Implemented
FR23	Search History by Date	Not Implemented
FR24	Customer service	Implemented

Chapter 6
Testing

6.1 Testing Approach

Testing is considered one of the required steps of developing any software, because it helps in determining whether the system of software is functioning properly and as expected or not. the testing approach that is has been applied is divided into functional testing and non-functional testing.
There are two kinds of testing which have been used:

- Black box testing: this technique is tested among most of our system features. In which, the functionalities are being tested by focusing on the input and output without looking at the internal code, and having the requirements and features are examined. The expected outputs are known and are compared to the actual outputs.

- White box testing: this is a technique helps to improve the system design, usability and security in which the internal structure, design and coding are tested to verify the flow of the input-output and the commands by using multiple different print statements internally.

6.1.1 Functional Testing
In this testing approach, we are validating the Baleghni system among the implemented functional requirements. And several types of functional testing have been used, are listed below:

- Unit testing: each module has been tested separately to validate whether it is functioning properly or not. It was commonly used before attaching it to other system parts to check for errors and minimizing mistakes and flaws before continuing with the implementation.

- Integration testing: each individual unit are combined and tested as a group to check the functionality as a whole and to expose any errors between the integrated units. For example, when the driver either sends or cancels a request, we checked that the change was received by firebase and the dispatcher.

- Regression Testing: all our functions have been thoroughly tested at the conclusion of their creation and after weeks of development of our product. For example, after switching our application from automated search to dispatcher, we still needed to test all our earlier work to guarantee that the new code modifications didn't affect their current functionality.

- System Testing: in the system testing, we double-checked that everything was in order and that there were no problems or faults, all the integrated individual units amongst several different cases to evaluate the system compliance among its end-to end-features, requirements, packages after completing our project.

Baleghni

6.1.2 Non-functional Testing

For the non-functional requirements (mentioned in Chapter 2), the following testing has been done:

- Security: in Baleghni system, the purpose of testing for security is to identify all the threats and weaknesses of Baleghni system in order to secure the sensitive and important information for the driver, police agent, and the dispatcher, and how to make the system safeguarded against any intruder while transmitting the request to dispatcher by using the POST request, which makes for any unauthorized access hard.

- Reliability: the system was evaluated for its ability to fulfill its functions as predicted without crashing or causing delays by comparing the timestamps of the transmitted requests, with the requests themselves.

- Performance: Requests in our project are submitted in under a second and immediately updated in the database and for the dispatcher, and request responses are the same for both the drivers and the police agents.

- Transparency / Usability: our system offers a basic, easy-to-use user interface that can be utilized by people of any age or with no technical experience. To be sure, we tested it with members of our family of varied ages.

Baleghni

6.3 Testing Samples

Here are some examples of samples created for manual functional testing:

Test Case ID	**TC1**
Test Title	Creating new driver user account.
Related Function	FR1
Test Description	This test is to test the system capability to register a new user account, which includes: • Handle errors in unmatching entered inputs and notifying the user. • Sending a confirmation email to the user, in case of confirmation it must be able to authenticate and verify the user. • Fill the database with the user entered inputs without any change and matching the data types and forms rules. And not allowing any null value.
Pre-conditions	• There is no existing record for the new user. • The email should have a valid domain and has no spaces. • The password should be at least 6 letters. • National ID and the phone number must be a 10 digits number without any spaces or symbols and nonexistent in database. • The phone number should consist of 10 digits and starts with 079 or 077 or 078. • The difference between the current date and the entered birthdate should be at least 18 years and written in the form of DD/MM/YYYY.
Test Steps	1. User launches the application. 2. User clicks the sign-up button in the application. 3. The user writes an email without @. 4. The user writes phone number with spaces and starting with 073. 5. The user gets notifies on entering invalid email and phone number values. 6. The user enters all correct data and finishes signing up. 7. The driver receives a confirmation email. 8. The driver clicks on the link to complete and confirm the registration. 9. The driver information is listed and added to the firebase database.
Test Data Sample	National ID: 9980234277 First name: Rinata Last name: Bakeer Email: test@email.com Phone:0790000005 Birthdate: 20/12/1999 CreatedAt: May Wed 26 May 2021 4:48:27 UTC+3 Gender: Female User type: Driver Password: password123

Expected Results	• Step 1, the application should launch successfully. • Step 2, sign up is displayed successfully. • Step 5, the user is alerted that the email in an incorrect form, and that the phone number is unacceptable, and it is in an incorrect form, should be without spaces and starts with 079 or 078 or 078. • Step 6, all the entered inputs are in the correct form and the login page is displayed. • Step 7, the user receives an email with a working confirmation link. • After clicking on the link, the user account is validated and placed in "User" table in the database. •
Post-condition	User is created successfully and listed within Users table within the firebase.
Status	Pass

Baleghni

Test Case ID	**TC2**
Test Title	Sending a request and assigning a police agent test
Related Function(s)	FR10, FR16, FR17, FR18, FR19
Test Description	This test made for testing the system's capability in managing sending requests, assigning police agents, sending notifications, updating dispatcher map and tables, and not allowing any null or zero values.
Pre-conditions	• Non-Existing pending request for the Driver. • Free Police Agent status. • Permission to get location granted. • Permission to receive notification granted. • Location set accurately.
Test Steps	1. User logs into the application 2. Home page is displayed and showing the current driver location within the map by default. 3. Driver moves the location pointer for desired location 4. Driver presses "Report an accident" button. 5. The request is sent to the dispatcher with the needed information. 6. The request is marked up in the dispatcher Map page. 7. The dispatcher assigns a free police agent for the request. 8. The driver gets notified for the police agent assignment and information.
Test Data Sample	Accident ID: 0cb4AZ6224hMGqZuPIWQ Driver User ID: 66Kn8gffElOUjIf278EdZsUCAz53 Police_User_ID: lCcYRJu9QreBmZzo2SUwmww0oDL2 Latitude: 32.555597 Longitude: 34.152164 Request send time: 27 May 2021 4:48:27 UTC+3 Request_Accept_Time: 27 May 2021 4:49:21 UTC+3 Status: Accepted
Expected Results	• When the request is sent, the information related to the location (longitude and latitude) and the timestamp for the sent request must be correct, and visible in the Request table, the Requests drop down menu, and then marked up within the dispatcher map page. • Creating a new record for the sent request that should include information of the driver id, longitude, latitude, requesting time, and the status with "pending" value. • Dispatcher assigns free police agent for the request, will notify both the driver and the police about the update, the record will be updated to also contain police id, accept time, and update status to "Accepted"
Post-condition	Request is created successfully and listed visible to the dispatcher with all the needed information
Status	Pass

6.2 Testing Tools

- IOS emulator.
- Android emulator.
- Physical android/IOS devices.
- Visual studio code debugging mode.
- Postman application

6.3 Tested Features

All the system's functional requirements (features) which has been implemented are tested except for non-implemented ones. (See table 16 for more details on the tested features)

Table 16: Tested Features

Feature ID	Feature name	Details	Priority	Tested
FR1	Sign up	Creating new driver account and adding it to the database.	Essential	Yes
FR2	Login	To allow the user to login to the application using his email and password	Essential	Yes
FR3	Forgot Password	Setting up a new password for the driver in case forgetting it.	Essential	Yes
FR4	View Profile	Showing all the user information for them.	Essential	Yes
FR10	Request a police agent	Allowing the driver to send a request for a police agent using his current location	Essential	Yes
FR14	Driver cancels request before confirming it	Driver can cancel their request within 5 minutes before confirming	Essential	Yes
FR15	Driver cancels request after confirming it	Driver can cancel their request after confirming	Essential	Yes
FR16	Display notification for police agents upon requests arrival	Providing a notification for the police agents to notify them on arrival of a request that they have to go to.	Essential	Yes
FR17	Police agent status change	Allowing the police to show if they are free or busy.	Essential	Yes
FR18	Displaying a popup for driver that his request has been accepted	Providing a notification for the driver to notify them on request acceptance -	Essential	Yes

FR19	Display Driver/agent information	Provide both users, the driver and police agents of each other information (to make interaction easier and fast).	Essential	Yes
FR20	Map Tracking	Dispatcher can track the request and police agent's locations.	Essential	Yes
FR21	Language Control	Changing the app language from Arabic to English or vice versa.	Recommended	No
FR22	Display request history	Allowing the driver/police agents to see their current and previous requests details by Dates	Essential	Yes
FR23	Search History by Date	Allowing the driver to see his current and previous requests details in a by dates.	Recommended	No
FR24	Customer service	The driver/police agent could call the Baleghni customer service if they needed help.	Recommended	Yes
FR25	Registering a police agent	Allowing the dispatcher to create a new police agent account.	Essential	Yes

Baleghni

Chapter 7
Conclusions and Future Work

Because the Baleghni app presents a new idea for reporting traffic accidents in Jordan, which has the potential to change the way we deal with traffic accidents, it allows us to look further into the future. In this project, we have implemented an efficient reliable technology for accident reporting. We managed to make it user friendly, available for everyone and secured.

In terms of accident reporting in Jordan, our Baleghni model offers a lot of space for expansion. Many future aspects, such as the utilization of the potential of obtained data through data analysis, may be recognized, and achieved.

Some of the things that are considered future development include:
- By supplying dynamic traffic data and improving the algorithm employed, the main objective is to make this software more real-time and autonomous.

- Statistical reports would greatly enhance the system's capabilities and would aid in making important decisions about automobiles, highways, and other matters.

- Allowing the Driver to add/choose their vehicle when filing a request and automatically synchronizing the vehicle information with the police officer.

- Drivers/police agents will be able to evaluate each other once their request has been completed.

- A more advanced user registration system will be put in place.

- Instead of filling out the information on the sign-up form, scan an ID or a passport for details.

- Towing truck services are available.

Baleghni

Appendix A

Users' Manual

This section provides a full user manual description (textual and pictorial) for our Baleghni system and how to use it properly.

For installing Baleghni application on your phone, the user, whether they is a police agent or a regular driver, must install it from Play Store, for Android, or App Store for IOS devices. After the installation process is completed, the user can launch it by tapping on it.

Driver

1. After launching the application, the login page (in Figure 33) will appear first. And to be able to use the application, the user has to login using a valid email and password, then they will be directed to the Home page.

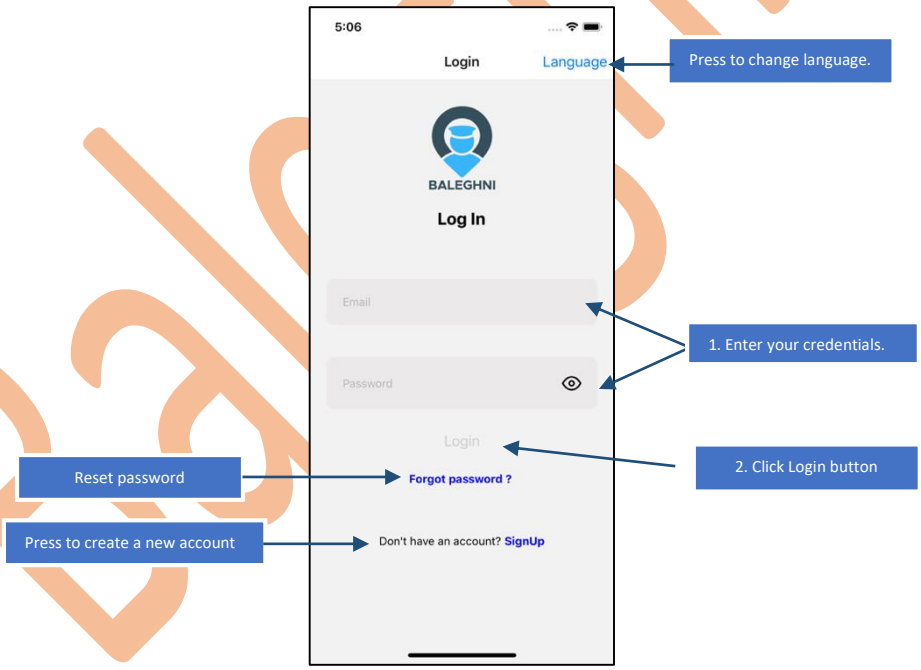


Figure 33: User Manual - Login Page

In case the user doesn't have an account, they can easily make a new account by pressing the "Sign Up" button in the Login page and be directed to the Sign-Up page (see figure 34). They must fill out their required information, and press Sign up button to complete the new account creation.
Also, the user can change the language of the application (between English and Arabic) by pressing the upper right corner "Language" button (see figure 33).

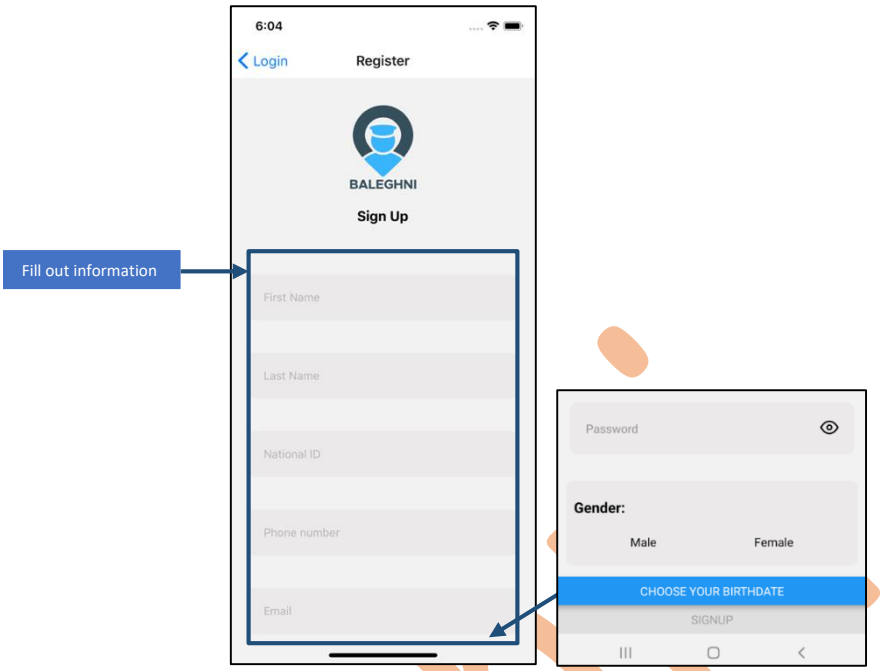


Figure 34: User manual - Sign up page

When the user fills out the information, they must take into their consideration:

- The first and last name must be of length greater than two.
- The national id must be a 10 digits number.
- The phone number must be 10 digits working Jordanian number which starts with 079 or 077 or 078.
- The email must be an existing working email with validate domain such as @gmail.com. (Any other form of domain such as @std.psut.edu.jo is acceptable).
- The password must be of length greater than 5.
- Choose Gender.
- The minimum accepted age is 18.

Baleghni

And in case of forgetting the password, they can press on the “Forgot password?” (See figure 35), then enter his email account and press on send. An email will be sent to them which provides a link for changing the user password.

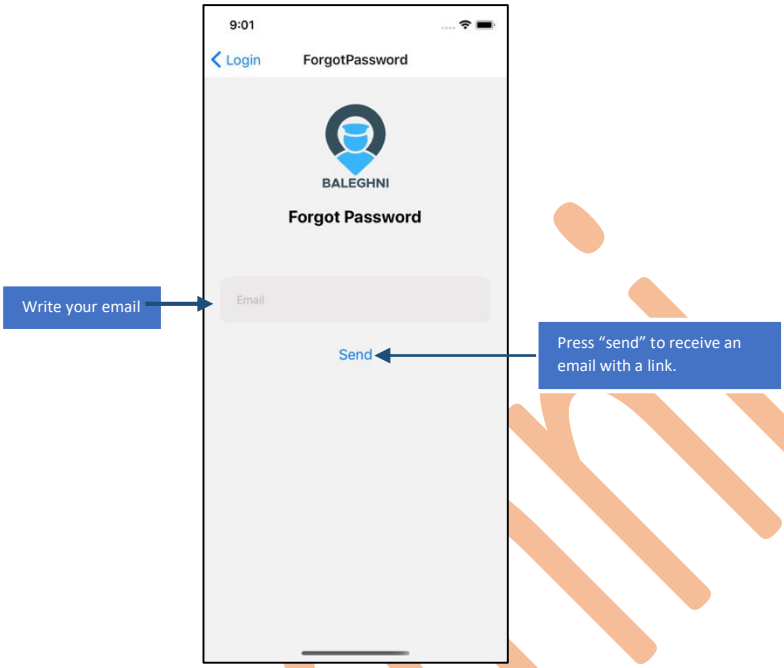


Figure 35: User manual Forgot password page

A prompt would appear for a new user opening the application for the first time to use the location services permission.

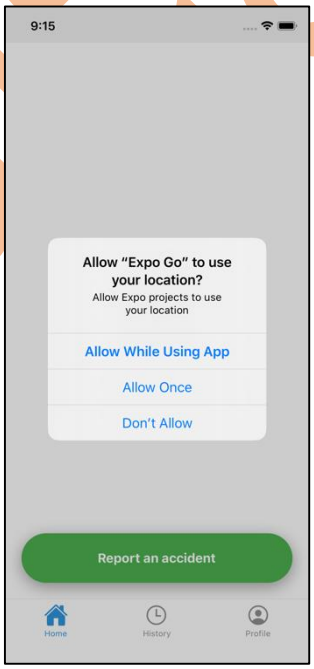


Figure 36: User manual - One Time Location permission request

2. After login in successfully, the home page will appear showing a map for the user with his current location pinned within. A new request can be made from the “Request an ambulance” button. The current location is synced and can be changed by moving the location pin or marker. (See figure 37)

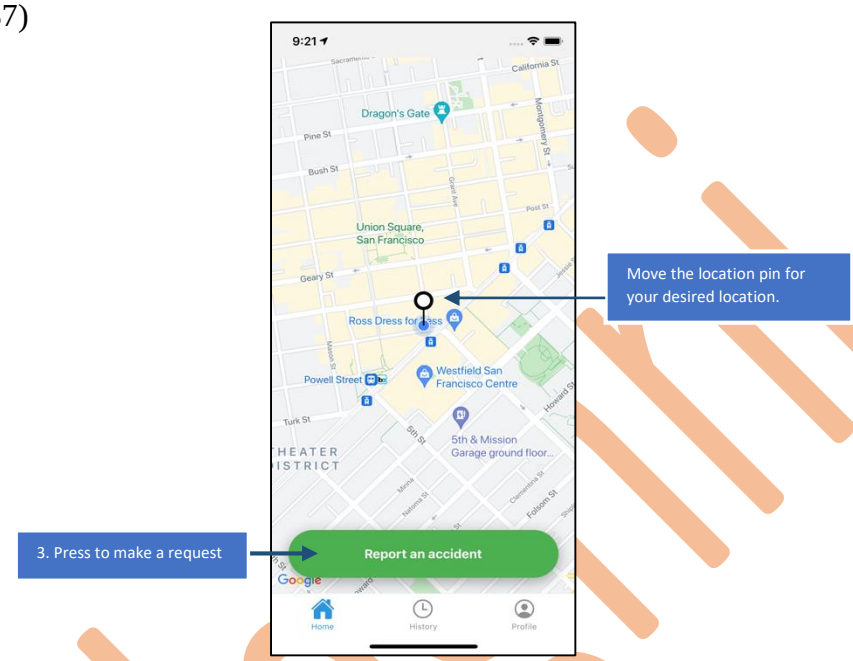


Figure 37: User manual - Home page

3. The user can press the “Report an accident” button after selecting their location or using default location (see figure 37).

Baleshini

4. If the Driver wanted to see their accidents requests history and their details, they can press on "History" within the navigator bar (see figure 38). Then the History page will list all the previous and current requests that have been made by the user.
5. To get more details on a request, the user can select it (see figure 39).

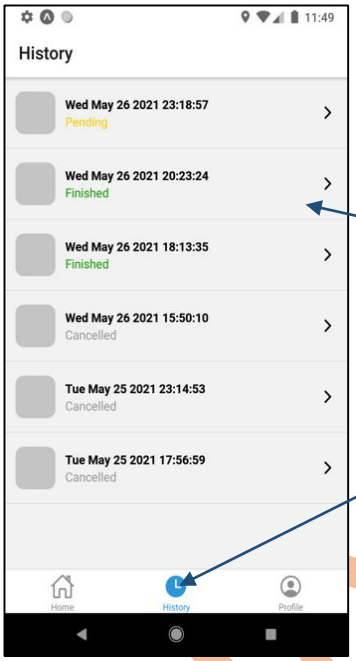


Figure 38: User manual -Driver History page

5. Press to open details

4. Press to open history

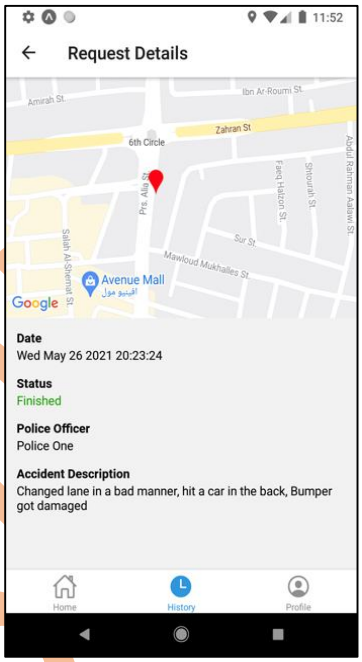


Figure 39: User manual - Driver Request Details Sample.

Baiech

If the Driver wants to either check their profile, call customer support, or logout from the application, they can simply go to the profile screen through the bottom navigation. (see figure 40)

Figure 40: User manual - Driver Profile Screen

Baleghini

Dispatcher

The interaction between the dispatcher and Baleghni system is through a website not a mobile application. The dispatcher will be able to log in, create new police agent profile, and track the requests and map them with police agents.

There is no sign-up page for the dispatcher, to avoid a lot of security issues related to the privileges that they have.

1. The dispatcher can login using their email and password that an admin have created for and allowed access for (see figure 41).

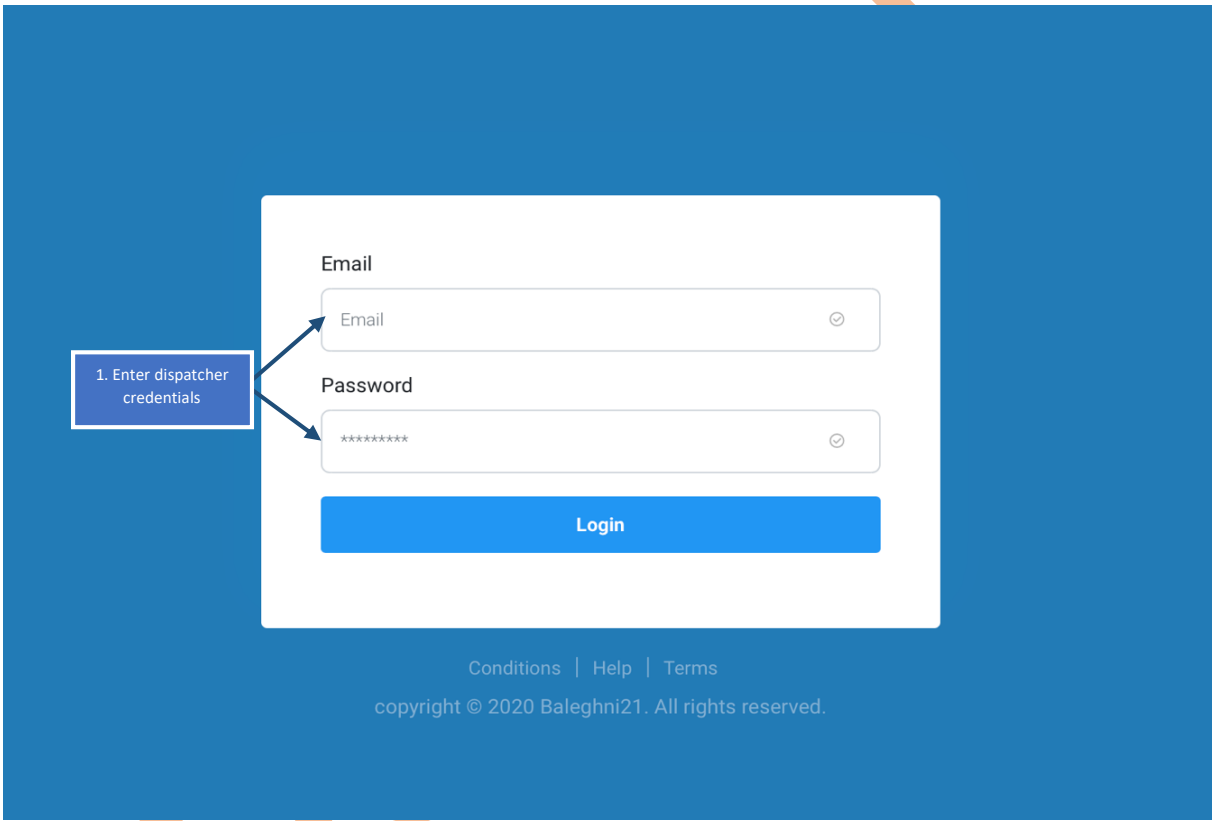


Figure 41: User manual - Dispatcher Login Page

The dispatcher will be directed to the Map page.

The dispatcher will be able to view all ongoing requests, as well as active police agents, then, will be able to assign a free police agent to a pending request. (see figure 42 and 43)

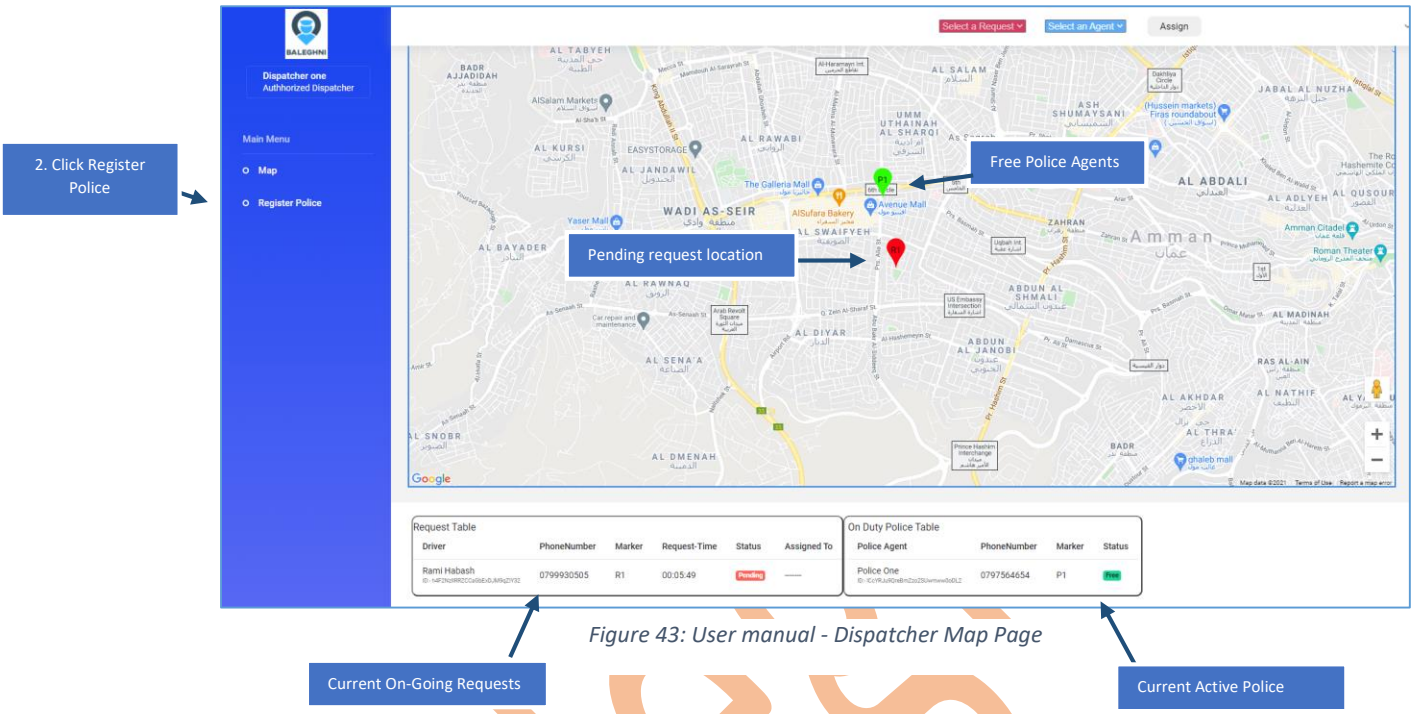


Figure 43: User manual - Dispatcher Map Page

Figure 42: User manual - Dispatcher Assignment Process

2. If the dispatcher wants to register new police agent account by clicking on “Register Police” on the left side. Then filling out his details including their full name, email, phone number, etc. (See Figure 43).

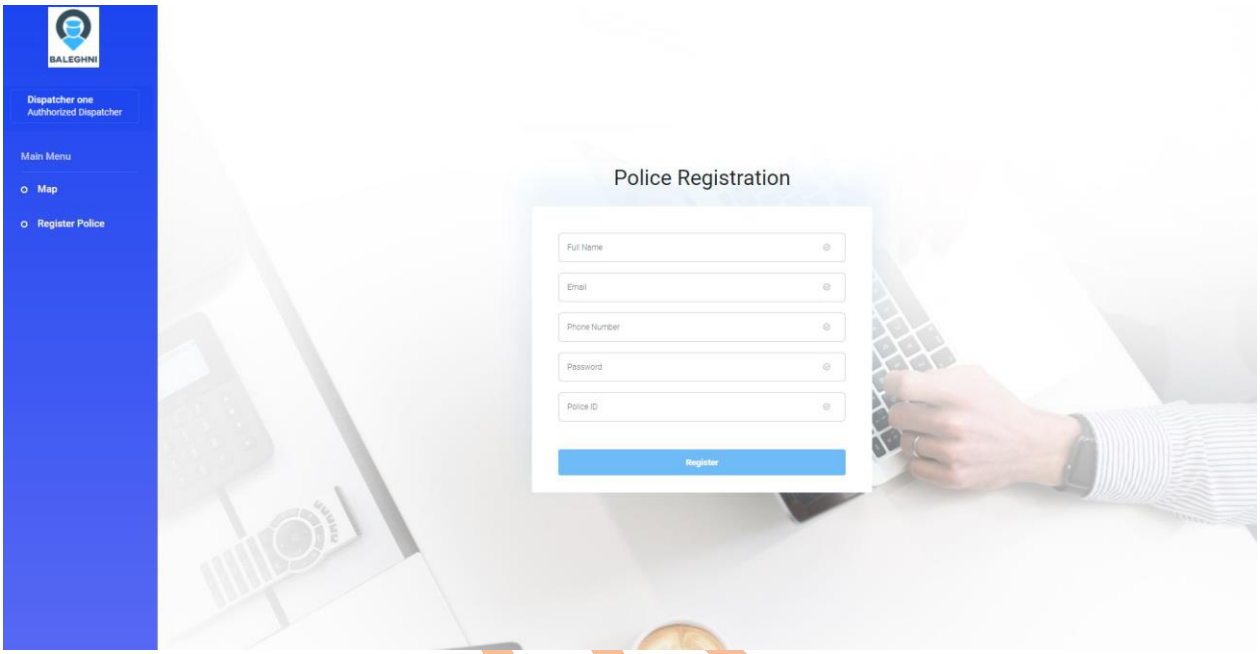


Figure 44: User manual - Dispatcher Register Police Page

3. To sign out from the website, the dispatcher can click on the upper left arrow within any page that the dispatcher in, then they can click on “Sign out” within the dropping menu. (See Figure 44)

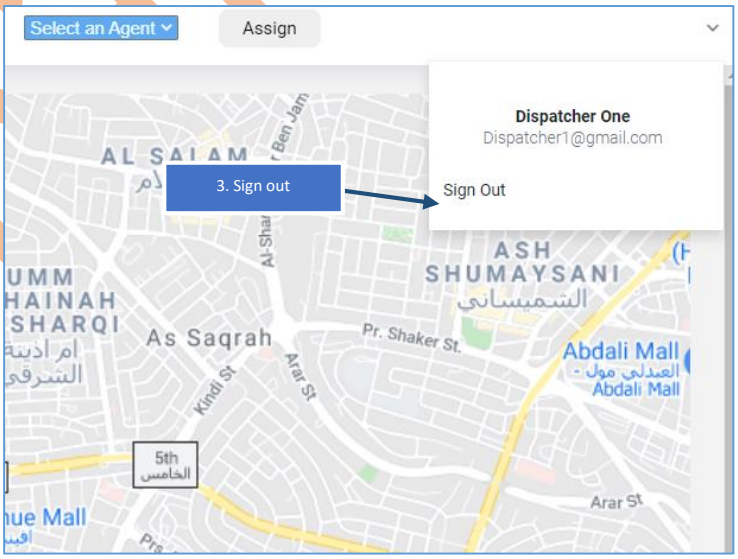


Figure 45: User manual - Dispatcher Sign out in Map Page

Police Agent

As a regular driver user, the police agent can use Baleghni through a mobile application.

1. After launching the mobile application, in the login page the police agent can login to the application using his email and password as the driver user. (See figure 33)

2. The police agent can't create their own account by themselves, the creation of a new police agent account only available through the dispatcher.

3. After login successfully, the police agent Home page will be launched providing his current location and a status switcher. The police agent can inform the dispatcher if they are available to receive new requests or not by pressing the upper right status switcher. (See figure 46)

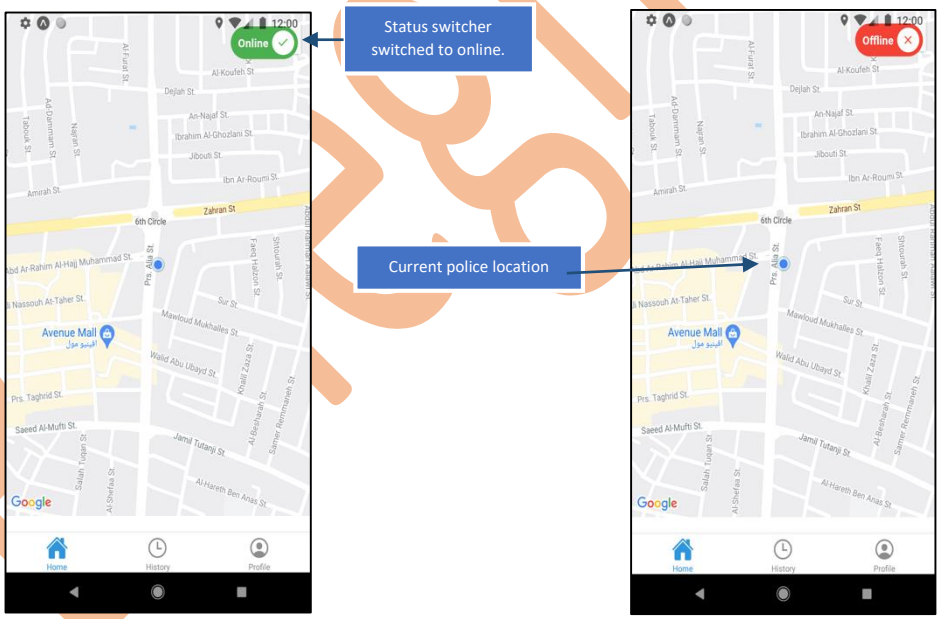


Figure 47: User manual - Police Home page with Online mode

Figure 46: User manual - Police Home page

4. If a new request has been assigned to the police agent by the dispatcher, they will be notified. And the Home page will automatically provide two buttons: “Call Civilian” if contacting the driver is needed, and the “Arrived” button which is pressed to notify the dispatcher the arrival of the police. (See figure 48)

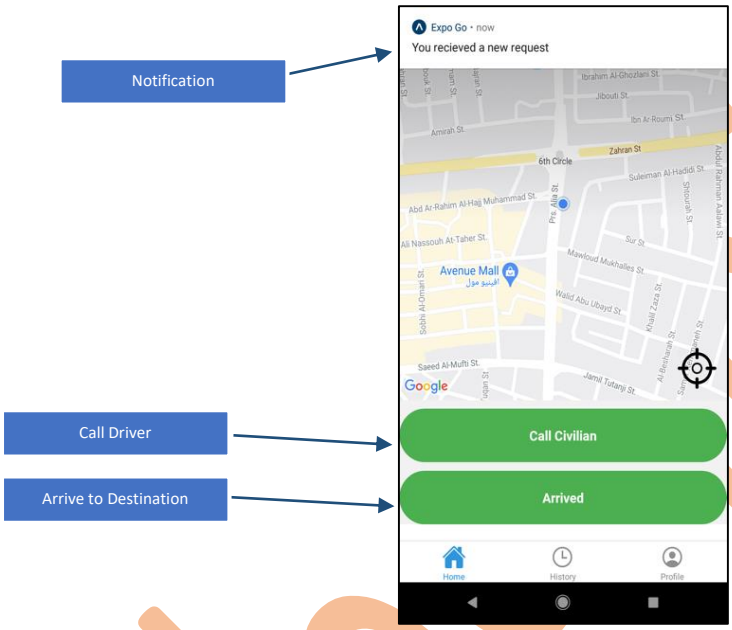


Figure 48: User manual - Police Receives New Request

5. After reaching the destination, the police agent will be notified that they successfully reached the accident location. The police officer can click on the “Description” button to add more details to the request. After completing to report the accident, the police agent presses on the “Finish Request” button. (See Figures 49 and 50)

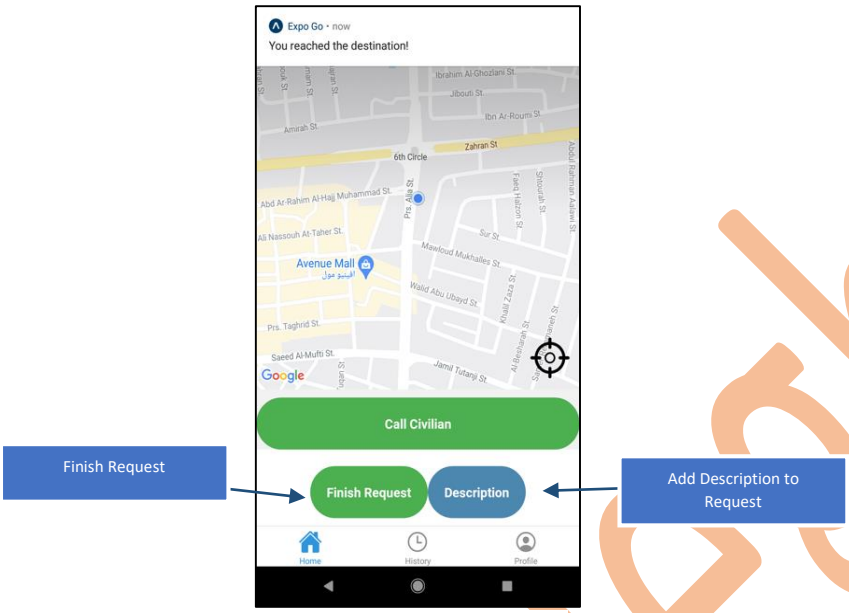


Figure 49: User Manual - Police Arrival page

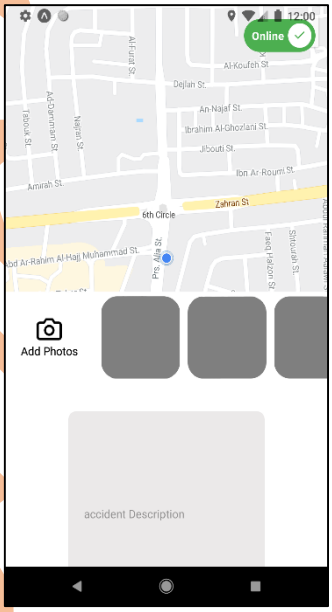


Figure 50: User Manual - Police Description page

Baleghi

6. If the Police wanted to see their accidents requests history and their details, they can press on "History" within the navigator bar (see figure 38). Then the History page will list all the previous and current requests that have been made by the user.
7. To get more details on a request, the user can select it (see figure 39).

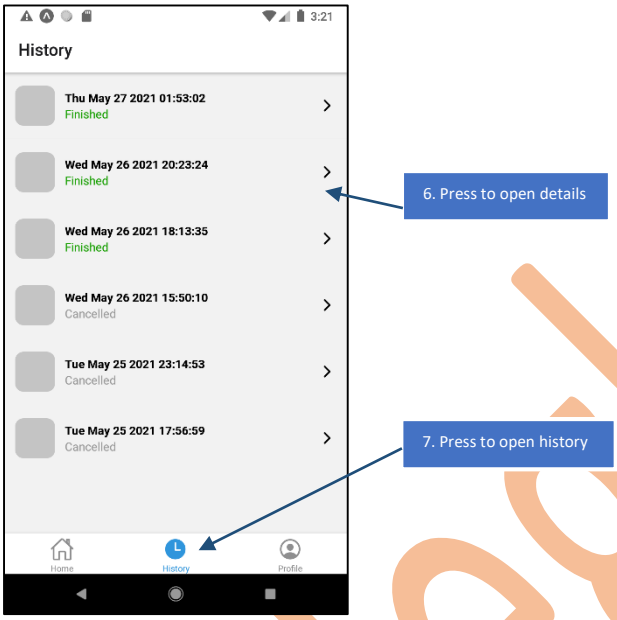


Figure 49: User manual -Police History page

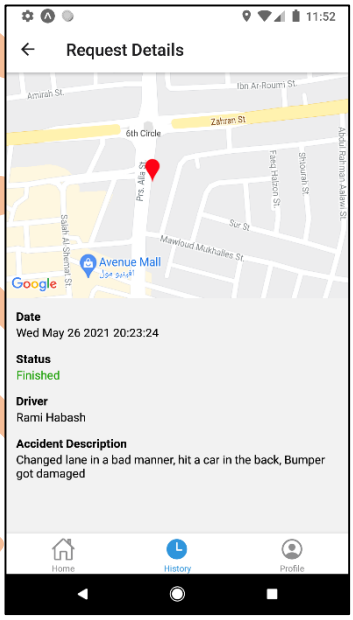


Figure 50: User manual - Police Request Details Sample.

Baleegh

To check the Police profile, to call customer support, or to logout from the application, they can simply go to the profile screen through the bottom navigation. (see figure 51)

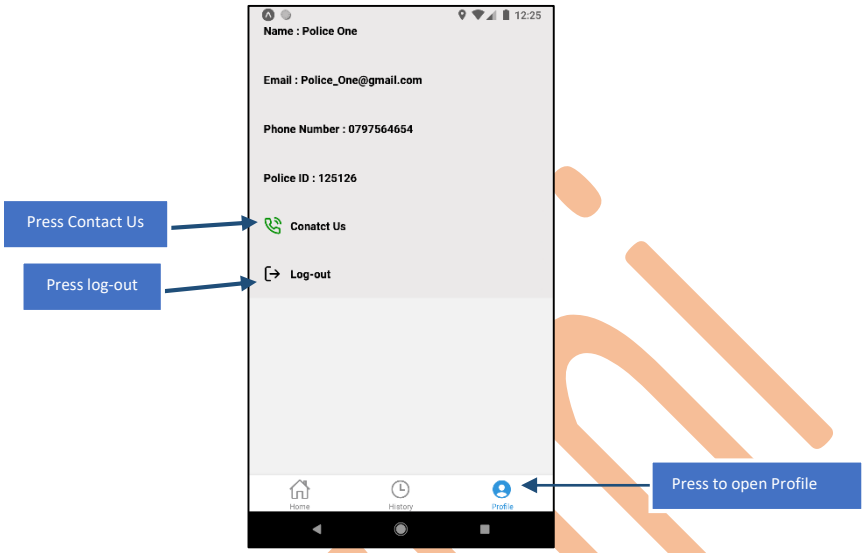


Figure 51: User Manual - Police profile

Baleghini

Appendix B
Document Changes

During the second semester, some modifications and changes have been done on the project's structure, system design, requirements and UI. Some changes can be removed, added or modified due to the reevaluation after implementing the Baleghni system.

The changes are listed within Table 17:

Table 17: Document Changes Table

Change ID	Change that has been done	Type of change	Reason for Changing
DC1	Functional requirements related to vehicles are considered to be future work.	Removed	Lack of data related to vehicles with car accidents history.
DC2	Dispatcher unit	Added	Implementing an automatic search system is complicated with short amount of time available.
DC3	Database Scheme	Modification	Cloud store works differently that traditional databases.
DC4	Dashboard website	Added	To have a dashboard that allows the dispatcher to do their tasks.
DC5	Project Costs	Modified	Google firebase has been used as an external hosting server.
DC6	Attach an Image upon request (Driver)	Modified	Attach an image upon finish request (Police)

Baleghni

New Database Scheme

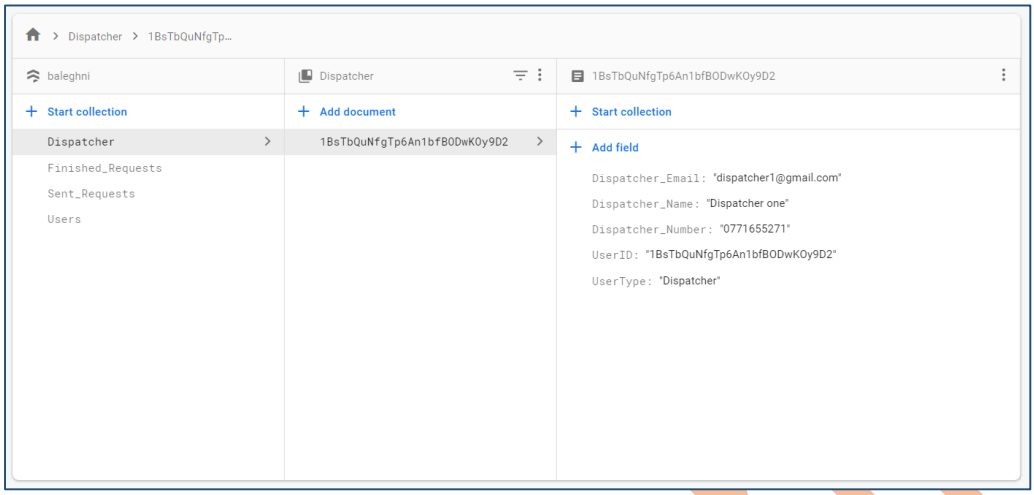


Figure 52: Dispatcher Table in Firebase

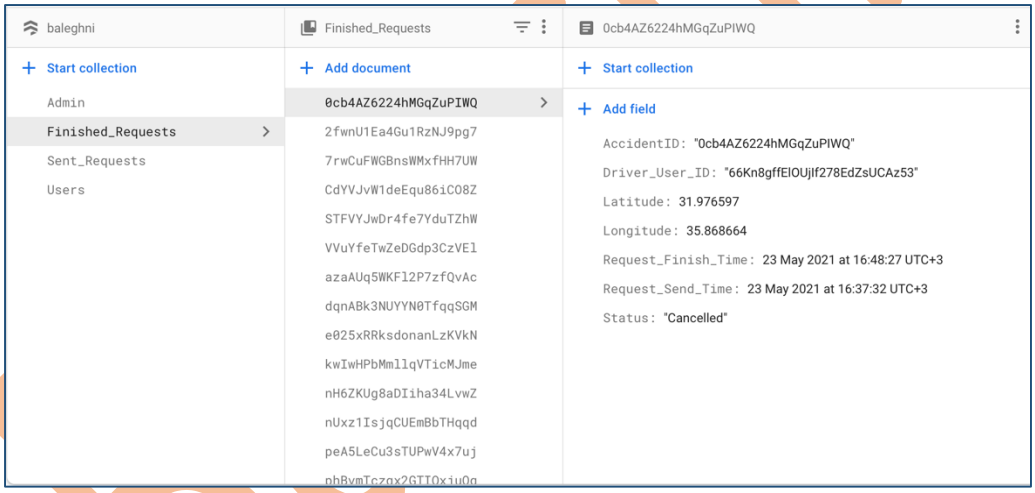


Figure 53: Cancelled Request in Finished Requests Table in Firebase

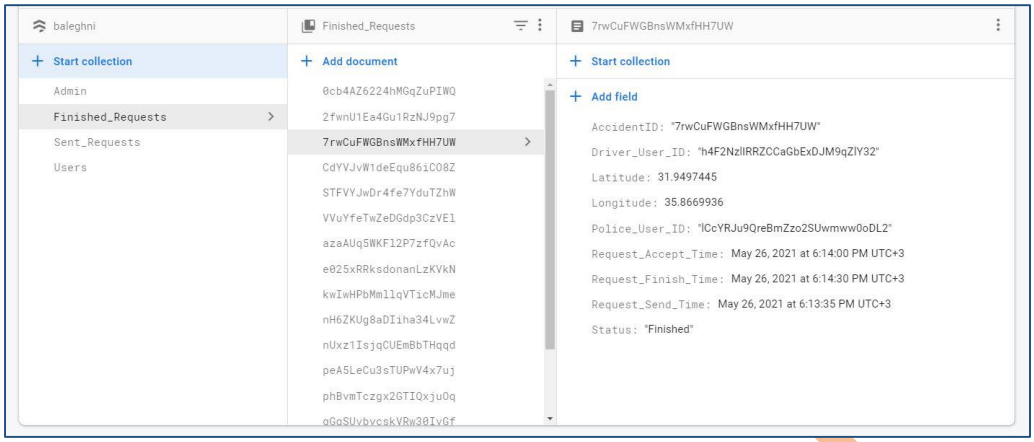


Figure 54: Completed Request in Finished Requests Table in Firebase

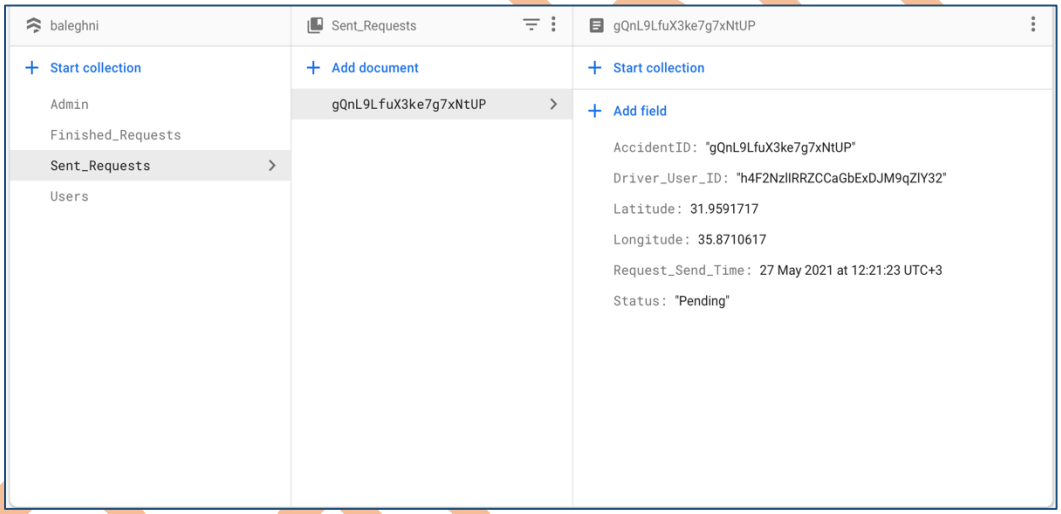


Figure 55: Pending Requests in Sent Requests Table in Firebase

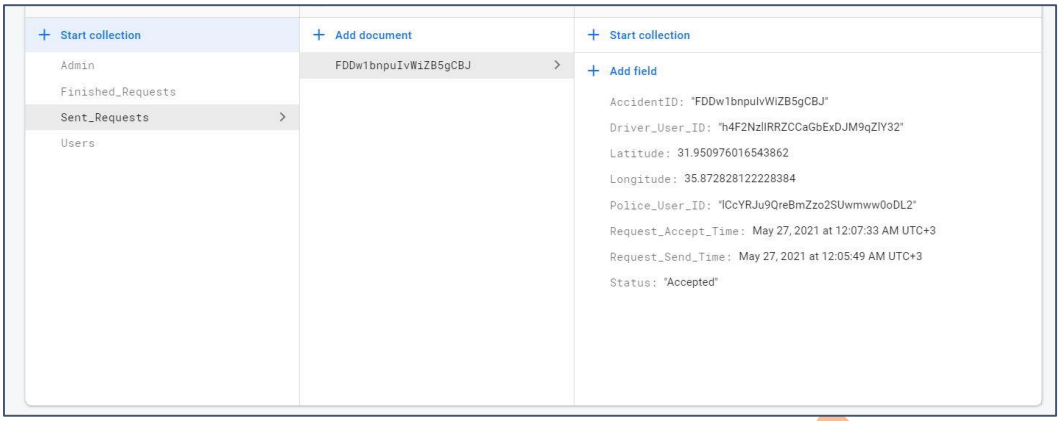


Figure 56: Accepted Requests in Sent Requests Table in Firebase

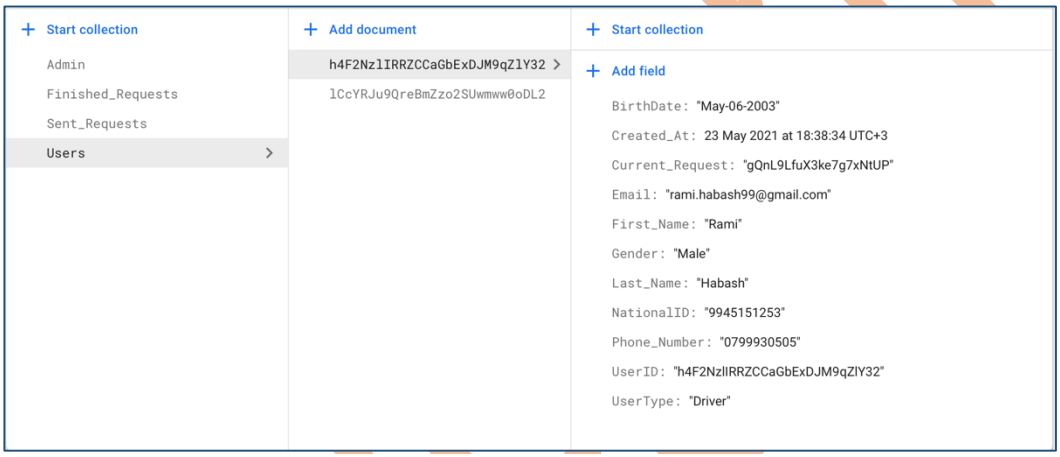


Figure 57: Driver in Users Table Firebase

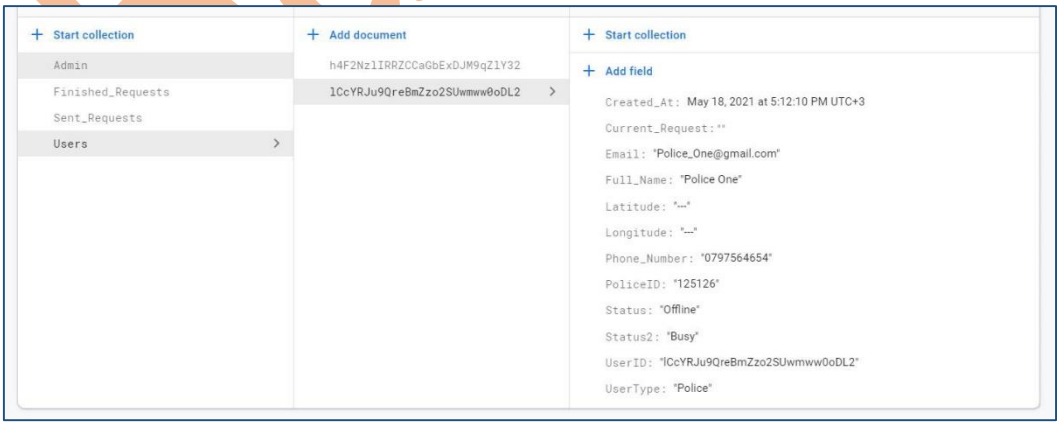


Figure 58: Police in Users Table Firebase

Appendix C

Code Documentation

For our code documentation, we used TypeDoc which converts TypeScript source code into rendered HTML documentation.
Figure 59. shows the main sample of our Code Documentation.
The full Code Documentation will be found in the folder “Baleghni Code Documentation” attached in the project.

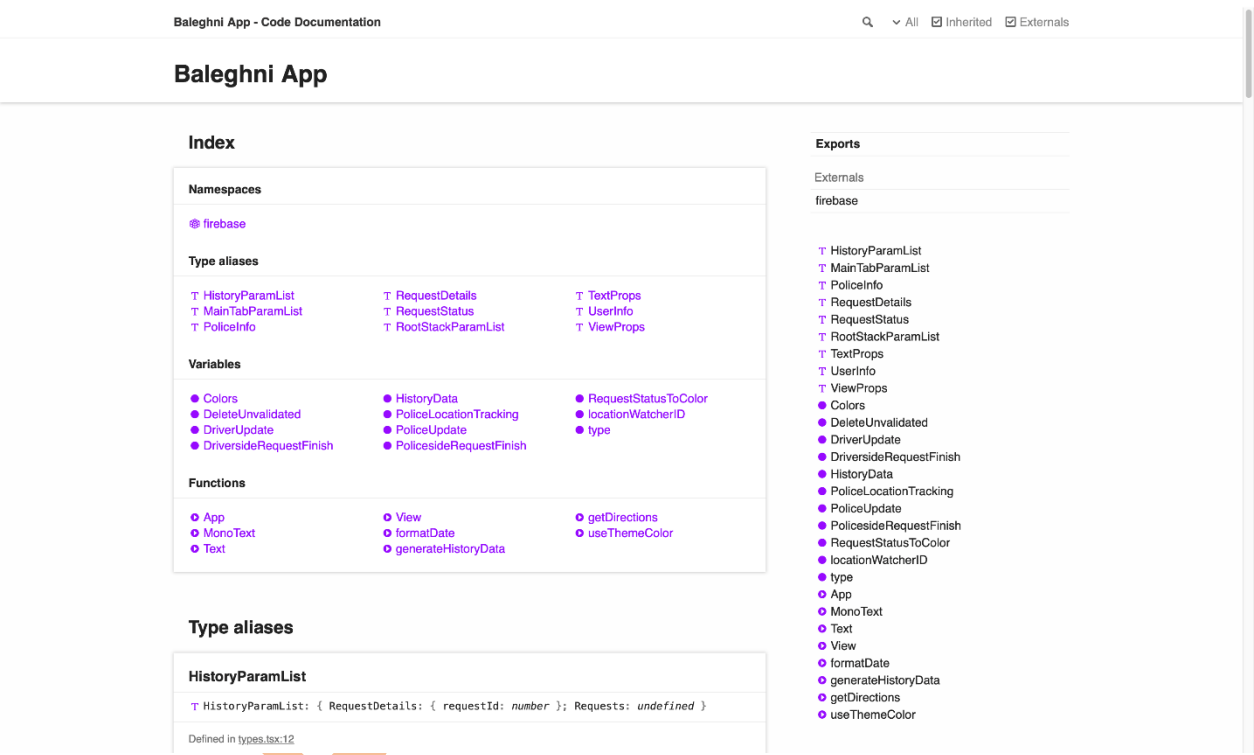


Figure 59: Code Documentation Sample

References

[1]https://apps.apple.com/app/id953140409?fbclid=IwAR1Eh_BmBZSAJiJaWVdPTG2xFY4p32vvhl4g9CmYMtIptglT-RzvVzyN7nM

[2]https://apps.apple.com/us/app/dubai-police-%D8%B4%D8%B1%D8%B7%D8%A9-%D8%AF%D8%A8%D9%8A/id384374316

[3]https://apps.apple.com/us/app/accident-reporter-track-vehicle-incidents/id1211490421

[4] All icons in the UI have been used from https://icons8.com

[5] https://app.dbdesigner.net/

Baleghni