Republic of Egypt
Ministry of Higher Education

Tanta University
Faculty of Engineering
Computer & Automatic Control
Engineering Department

IMPROVING PUBLIC TRANSPORTATION USING RESTFUL API AND MOBILE APPLICATIONS

Under The Supervision Of:

Dr. Tahani Allam

Group Members:

Alaa Mostafa Abu El-Khier
Ammar Ahmed Eltabaa
Asmaa Ashraf Foda
Asmaa Mohamed Ghodia
Hagar Abd-Elatif Hashem
Kareem Ebrahim Allam
Omar Ahmed Attallah
Raaed Elmaghory Serag
Somaya Mostafa Elbaradei

ABSTRACT

In this project we state a solution to one of the transportation problems. Klax is supposed to perform 2 main functions;

- The first function is to help passengers who are waiting on the roads to get rides, which help them not to wait for a long time and help passengers who are riding in buses not to wait a lot to start their trips.
- The second function is to help bus drivers increase their daily target of trips.

And to achieve these 2 functions we provide:

- **2 Mobile Applications:**
  - **User App**
    The It allows the passenger to sign up to select his current location, select destination and book seats.
  - **Driver App**
    It allows the driver to sign up after applying all required documents to show the road lines he works in, checks the users' requests, accepts these requests in case of having enough empty seats and updates the empty seats number.
- **Web Application:**
  - **Admin Panel**
    It monitors users, drivers and trips numbers.

Contents

List of Figures

List of Tables

CHAPTER

1

Introduction to Intelligent Transportation Systems

1.1 *Introduction*

Roads are a core piece of infrastructure that supports the movement of people and logistics, forming the foundation of social and economic development and interconnecting cities, ports, projects and airports.But while road maintenance and improvements bring about economic development, the increased volume of traffic causes problems such as traffic accidents and congestion. Intelligent Transport Systems (ITS) is a new transportation system which aims to resolve a variety of road traffic issues, such as traffic accidents and congestion, by linking people, roads, and vehicles in an information and communications network via cutting-edge technologies. It includes, for example, a road traffic information provision system in which road traffic information is collected via roadside sensors

and then provided to drivers. ITS provides people with a variety of convenient road traffic applications. In addition, the provision of new ITS applications through the use of a variety of information and communications technologies greatly contributes to the creation of new business opportunities and markets, as well as the vitalization of economic activities.

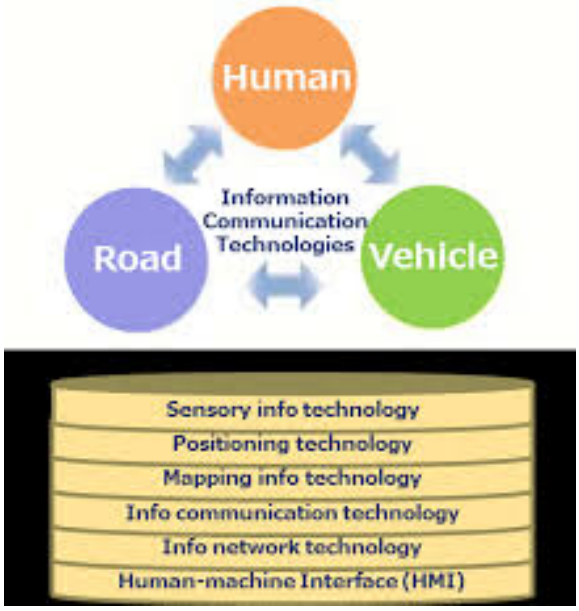


Figure 1.1. *Components of ITS*
[1]

The three most important components needed for establishing ITS are location, mapping, and communications. An example of the component technology used for location includes the Global Positioning System (GPS). As GPS receivers have become quite inexpensive, they are now extensively used on familiar information devices, such as car navigation systems and mobile phones. Mapping is a technology that plots location information on a map and can be realized with the help of a digital map.[1]

1.2 *Impact Of ITS*

Advantages of using intelligent transportation systems(ITS) systems are defined by various performance indicators. These indicators represent how ITS systems can improve the safety and mobility of the traveler, the efficiency of the transport system, performance of providers of transport services, energy-saving, and environmental protection. These measures include:

- Safety: direct safety measures may include the number of accidents and changes in their number, the number of injuries and fatalities (absolute measurements). Direct relative measures can be analyzed as micro indicators (automotive, transportation, severity) specifying the number of events about the volume of traffic (miles traveled), the number of trips or proportion of involved fatalities or seriously injured persons in the total number of traffic incidents; and macro indicators (e.g. the density of accidents in the defined area or road section). Safety analyses also apply indirect measures which include traffic parameters and their dynamic range (e.g. vehicle speed, speed variation, changes in numbers of infringements of road safety traffic rules, rescue operation time, as well as drivers' and pedestrians' behavior measures). Safety can also be examined by defining social or individual risks
  .

- Mobility: these measures may include travel time, the variability of travel times (travel time reliability), variable speed (traffic flow), time to restore normal/typical traffic conditions (from the point of view of the user and efficiency of the transport system).

- Capacity: measured by the maximum number of people, goods or vehicles passing a point in the road (junction, perimeter or the reference point of the road network per unit of time), as well as traffic conditions (e.g. number of stops, number of vehicles/persons in the queues, time lost).

- Satisfaction of a service user (traveller, supplier of goods): related to the choice of means of transport and the quality of service measured by the level of satisfaction. Typical results of satisfaction with services provided include: assessment of professionalism of the service provided, meeting the traveller's expectations, quality of use, as well as the level of service efficiency and reliability.

- Productivity: activities related to sufficient operational performance and security of the service costs .

- Energy and environment: measures include changes in the level of pollutant emissions (carbon dioxide and carbon monoxide, nitrogen oxides, hydrocarbons and volatile organic compounds) and energy consumption .

The efficiency of road traffic is closely related to the aforementioned indicators. Traffic performance is upgraded by increasing safety, improving mobility (reduction of travel time), increasing road capacity and use of public transport vehicles, reducing the cost of freight, reducing the negative impact of road traffic on the environment, as well as satisfaction of the road user (ITS services are to make, among other things, the journey more pleasurable and less tiresome).

The result of the implementation of ITS services can be a balance between all the indicators, e.g. improvement of the level of safety in such a way as not to cause

a significant increase in travel time which translates directly into user satisfaction. On the other hand, the capacity should improve along with shortened travel time (taking into account all travellers, including cyclists and pedestrians) - but not at the expense of safety. In conclusion, the implementation of a new ITS service should consider comprehensively all aspects of traffic, so that the overall balance of such implementation was favourable, which requires developing methods defining such balance. [2]

1.3 *ITS Market Size*

The global intelligent transportation system market size was valued at USD 26.58 billion in 2019 and is projected to register a compound annual growth rate (CAGR) of 5.8% from 2020 to 2027. The necessity for presenting real-time traffic information of different regions to passengers and drivers is one of the significant factors driving the demand for intelligent transportation systems across the world. The growing number of vehicles on road, aging infrastructure, and lack of traffic data management are other factors anticipated to drive the overall growth. The need to enhance traffic flow across highways and corridors in cities has led to the rising need for an alternative technology to manage traffic. Transportation authorities can address the growing traffic issue by using innovative and advanced data analytic technologies.

High traffic jams owing to the increasing number of vehicles have contributed to the need for advanced public traffic management systems. Developments in emergency services and traffic management empower authorities to speedily

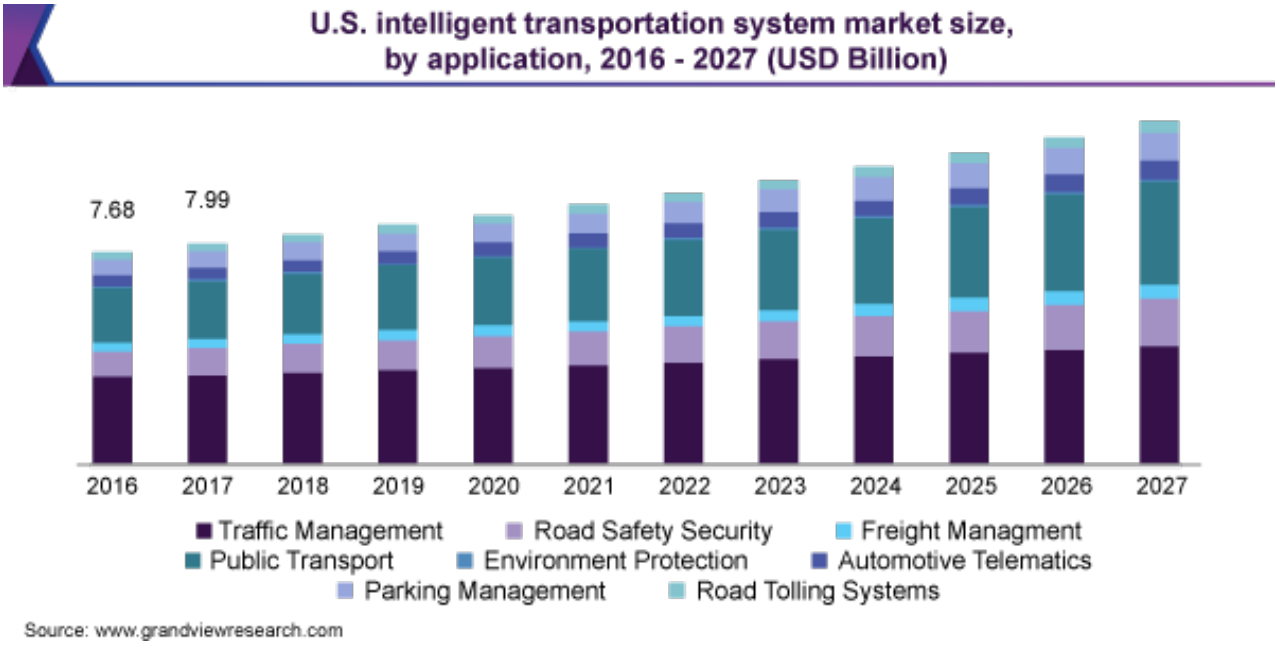


Figure 1.2. *ITS market size by applications*
[3]

respond to accidents and emergencies with more efficiency. The usage of such systems shortens the travel duration and increases efficiency in traffic management. The use of public transportation, such as buses and trains results in the reduction of carbon dioxide emissions and airborne pollutants.

Moreover, traffic congestion can be streamlined by deploying end-to-end connecting traffic management systems by using advanced techniques such as cloud computing and data analytic. Data collected from components such as sensors, video cameras, and embedded systems installed along the road can be used for analyzing and controlling real-time traffic across cities. Companies in the ITS market are conducting research and development activities to use data analytic and cloud to reduce congestion, improve traffic flow across various cities, reduce idle time, and improve fuel economy.

The logistics industry has been evolving with the advent of technology. Technologies, such as Artificial Intelligence (AI), Internet of Things (IoT),

blockchain and big data, are rapidly transforming the way transportation systems are designed, planned, built, and operated. Logistics and transport involve a significant exchange of information and data between customers and service providers, and advances in information technology have been facilitating the rapid exchange of information and data. The latest technologies have enhanced visibility and control over shipments and subsequently augmented efficiency and customer satisfaction.

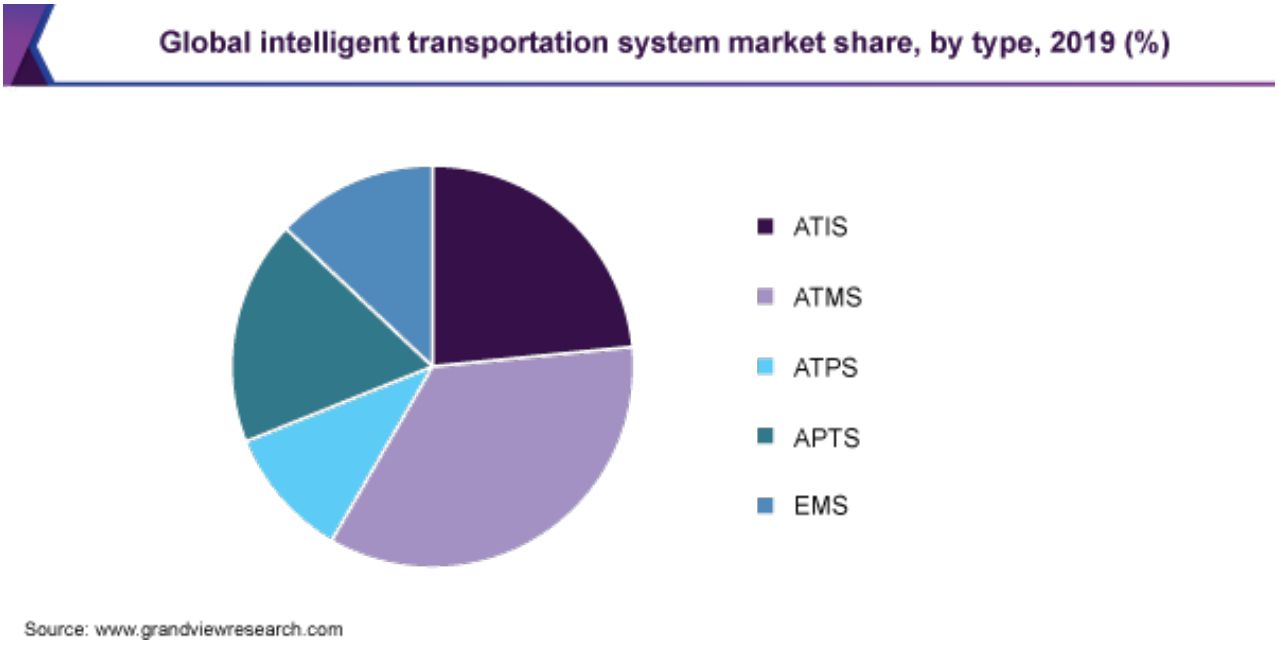


Figure 1.3. *global market statics by types*
[3]

Types of ITS application

- Emergency Management System (EMS)
- Advanced Traveler Information System (ATIS)
- Advanced Public Transportation System (APTS)
- Advanced Transportation Pricing Systems (ATPS)
- Advanced Traffic Management System (ATMS).

Advanced Traffic Management System (ATMS) type segment led the market and accounted for more than 30.0% share of the global revenue in 2019.

The advanced traffic management system primarily comprises applications for dynamic messaging signs, traffic signals, and ramp metering, among others. Bad traffic signal controls contribute considerably to congestion as well as an increase in the overall travel time to a large extent. ATMS helps in detecting dangerous weather conditions, roadway hazards, accidents, and creates a complete integrated view of overall traffic flow. They are installed with the present traffic control systems that generally help in reducing congestion and ensuring the efficient use of road space.[3]

1.4 *Advanced Public Transportation Systems*

APTS are ITS technologies applied to public transit in order to improve operational efficiency, cost savings, safety, quality of service, and other transit measures of performance. Some APTS applications offer potential for improving service by providing greater leverage to service providers for managing and controlling bus transit operations. Other APTS applications provide benefits in terms of speed, security, and convenience directly to the customer. APTS have the potential to significantly change the way transit services are provided and the way customers use them.

The first examples of APTS date back to the late 1960s and early 1970s with the introduction of Automated Vehicle Monitoring (AVM) systems. The majority of these vehicle location technologies were signpost-based systems (with

stationary signposts along bus routes). The signposts were equipped with electronic transmitters that emit unique identification codes. When a bus passed the signpost, an in-vehicle unit received the signpost's identification code and record the time and date, the difference between the current odometer reading and the last (recorded at the previous signpost), and the vehicle's identification code. The bus sent the information to the TOC via radio or other medium periodically or when prompted by the transit operations control center (TOC). Early implementations of APTS were expensive to install, operate, and maintain. Since then, new locations and communication technologies (e.g., GPS) have emerged and led to both improved performance and reduced costs.[4]

Transit Application	**APTS Technologies**
Fleet Management Systems	• Automatic Vehicle Location Systems • Transit Operations Software • Communications Systems • Traffic Signal Priority Systems

Traveler Information Systems	• Pre-Trip Transit and Multi-modal Traveler Information Systems • In-Vehicle Transit Information Systems
Electronic Payment Systems	• Smart Cards • Fare Distribution Systems
Transportation Demand Management	• Dynamic Ride-sharing • Automated Service Coordination • Transportation Management Centers

Table 1.1. *APTS technologies for transit application*

Fleet Management Systems:focus on improving the planning, scheduling, and operations of a fleet of vehicles. The related strategies aim at improved service reliability, safety, and operating efficiency (e.g., reduced non revenue time, increased productivity) and faster service disruption recovery. In general,

fleet management includes technologies that collect and make available vehicle performance data (e.g., vehicle location), and technologies that use that data for real-time control or for planning and scheduling. Important technologies used for fleet management purposes include: Communications Systems, Geographic Information Systems (GIS), Automated Vehicle Location Systems (AVL), Automatic Passenger Counters (APC), Transit Operations Software, and Traffic Signal Priority.

Traveler Information Systems refers to technologies used to provide travel information to passengers in order to assist their trip-making decision. The information provided may range from static route, schedule, and fare information to real-time vehicle location and/or estimated arrival time. AVL systems are the enabler for the provision of real-time information. Traveler information may be disseminated through various means, for example, web-based itinerary planning services, smart phones, in-vehicle units, etc. Traveler information is generally expected to improve the quality of transit service by improving the passenger experience. Traveler information may give passengers a better sense of control over their trip-making decisions and/or enable them to take action to minimize waiting times at stops, plan transfer connections, and thus reduce overall travel time. Information may be provided prior to departure (e.g., by phone, internet), at the terminal or stop, or in the transit vehicle. Information influences passenger trip-making decisions, for example, route, stop, departure time choice, etc. The attractiveness of transit alternatives is a function of waiting time, in-vehicle travel time, transfer time to the connecting trip, number of transfers, on board comfort

(crowding), etc. Information systems can provide timely information regarding many of these attributes.

Electronic Payment Systems forego cash and token payment with the aim of reducing the operating costs of fare collection systems, increasing safety and security on the vehicle, improving data collection, and increasing customer convenience . There are several electronic fare payment technologies, including magnetic stripe cards, and smart cards. Electronic fare payment technologies can have significant impacts on transit operations. The most obvious of the potential impacts on operations occur at the bus stop, where passengers board and alight from the vehicle. Depending on the type of electronic fare payment technology, considerable gains can be realized in terms of lower dwell times at stops through increased transaction speeds. Contact card technologies, where the card is physically swiped through a card reader, and contact less card technologies, where the card and card reader communicate without physical contact but rather via an electromagnetic signal, affect passenger boarding rates differently. Boarding rates increase to a greater extent with contact less card technologies.

Transportation Demand Management is the application of technology to alter the usage patterns of the transportation network, with an emphasis on encouraging users to travel by transit. There is a broad range of technologies designed to better coordinate various transit services, and provide forums for organized carpooling and car sharing . In general, approaches to managing transportation demand aim at increasing the number of high occupancy vehicles in congested transportation networks, promote travel in off-peak hours, and implementing transit

incentive programs. Examples include dynamic ride sharing and automated service coordination.[4]

1.5 *Project Scope*

Transportation is a daily need that we all require whether to go schooling, work, make a visit or even do shopping . In a country as Egypt with a broad population segment living on the public buses and micro-buses, it is a matter of millions to help them get a better transportation service that is at least more organized. At the main stations, buses -for example- take long time to get a complete number of passengers to move their way while many of them are already waiting for a ride in the road to the bus's final destination. This situation makes the bus get too much time waiting people, and on the other hand, the people are waiting for the bus miles or some away from the station. So, this project offers an advanced service for both the bus-drivers and the people waiting them. It allows the main user to register with his location and the user to book a seat or more according to their number so the bus-driver take off from the station with seats of their number unoccupied. This mobile app has a double-benefit relationship for the two kind of users due to the multi-featured software.

Our work to complete this project is divided into three milestones:

- **Develop a mobile-app (clax) for users which features**:
  - booking a trip at specific time .
  - selecting his line and station .

  - identify source destination location for user .
  - optimizing nearest station to pick-up.
  - selecting number of seats (max 5 seats).
  - tracking driver and knowing time for pick up.
  - booking a trip for other people.
  - tracking his family accounts and trips for safety .
  - paying trip cost by vise or cash .
  - editing profile page for user.
  - booking a private trip .
  - sending feedback

- **Develop a mobile-app (clax) for drivers which features**:
  - uploading information about him(photo, licence, address,..) ,his car (color ,car number,car licence,) and lines where work.
  - receiving requests from users.
  - tracking user locations .
  - following balance .
  - statics about number of trips and profit .
  - sending and receiving feedback about trips.

- **Develop a web server to perform the following tasks**:
  - Monitoring users, drivers and trip numbers.

- Geographic distribution of passengers according to their governs.
- Application monthly revenue.
- Making new offers and promo codes to users (discounts, free trips, free cash).
- Viewing passenger information.
- Getting every driver live location and make sure he is committed to his route.
- Viewing passengers complaints and send response.
- Adding lines and stations to the application whenever we decide.

CHAPTER

2

System Overview

2.1 *Introduction*

The system is composed of 2 applications, one of them is for the passenger and the other one is for the driver.The user will download the app, then sign up via Email, Name, Password and Phone Number. When the user signs up, his account will be verified via sending a code message to his phone number. And once the user verifies the sent code, he will get his Account. Then, the user has to deposit specific credit in the app's wallet so that the driver can get the receivable payment in case of the user booked a ride then cancelled it When the user wants to book his ride, he has to select the route line, then the nearest buses to him will be suggested. Then, the system automatically will choose the nearest of all buses and shows this Bus's Passengers. The user will be updated with any change in this number. The system will show all information about the bus and its driver. Also, the user will know the ride cost and how much time the bus driver takes to reach him. The user

has the choice to book any number of the bus seats. A feedback of the bus driver will be shown so the user can feel safe and comfortable. When the user confirms the ride, the bus driver will receive a request containing the user's place. Once the bus driver accepts it, the remained time for the bus arrival will be shown for both the user and the driver. The system may send the user a Promo code for free rides or any kind of offers. The user has the right to choose any of his Family or Friends to follow his rides. The system will notify the user and the driver with extra info, instructions or any updates generally.

If a bus driver wants to join Klax, he has to sign up via Email, Name, Password and Phone Number. Then check all needed conditions and tests for applying. Once he gets all conditions and tests successfully done, he gets An Account. If the driver account allows Multiple Route Lines, he has to mention them all and mention the activated one. And if it allows Private Rides, he has to mention the destinations and the cost. When the driver starts getting passengers for the bus trip, he has to manually add and update the passenger's number so that the user gets notified. The system automatically add and update the booked rides and seats. When there's is a user's request for a ride, all near drivers get notified. If the driver has enough empty seats, he accepts this request so that the user gets notified.

2.2 *Hardware Components*

2.2.1 Server

Where the process of deploying codes from source control and source artifacts to the platform is done.

- Web Server is deployed on Azure Site.
- Database is deployed on Atlas Db.

2.3 *Software Components*

2.3.1 Web Server

- The only component that has an access to Persistent Database.
- Has the access to External APIs (Paypal and Stripe).
- Has the access to Real-Time Database.
- Handles the user's requests.
- Validates all data before making it allowed in Persistent Database.

2.3.2 User Software Platform

The user system is composed of 3 essential sections as following:

1. **Home Model**

   - **Registration and login**

     The user will download the app, then sign up via Email, Name, Password and Phone Number. When the user signs up, his account will be verified via sending a code message to his phone number. And once the user verifies the sent code, he will get his Account.

   - **User's rides**

     past rides

     Where user can show details of trips and mark any trip as favorite.

     – Mark as favorite option.

     – Details shown: Trip Line, Date and Time, Trip Duration, Cost, Rating, Car ID.

     – Sort trips by: Date and Time, Trip line, Rating.

   - **Free rides**

     In Free Rides screen, a user can get a free trips using special codes provided by sponsors and partners. The screen includes:

     – Textbox for entering the redemption code.

     – A send button to send the code to the server along with some user info to be checked if it's valid.

     – A pop-up window as a response to the entered code.

     – An invitation code for each user that adds a certain cash amount to the user's wallet if a certain user used it, when he signs up.

- **Settings**

  User can adjust 3 types of settings:

  - **Edit Account:**

    Where user can show and edit personal details of account.

    * Show or edit name and picture of account user.
    * Edit password and forgotten password handling.
    * Show or edit phone and e-mail with verification handling.

  - **Edit App Settings:**

    Where user can edit customized settings.

    * Privacy Section

      Notifications pushing: Offers, My Trips, Paired Accounts Trips.

    * Safety Section

      Verify rides: using PIN.

  - **Family rides**

    Where user can show, add, or remove paired family accounts.

Figure 2.1. *Family Rides*

    * Send Pairing Requests: Search using mobile or email and create a request.
    * Received Pairing Requests: Accept or refuse.

        * Remove Paired Accounts.

- **Help**

  Tutorial on how to use application:

  - Quick Guide to KLAX.
  - Payment options.
  - Trips Issues.

2. **Pairing System**

Figure 2.2. *bus stop*

- **Get request from the user**

  When the user wants to book his ride, he has to select the route line, then the nearest buses to him will be suggested. Then, the system automatically will choose the nearest of all buses and shows this Bus's Passengers. The user will be updated with any change in this number. The system will show all information about the bus and its driver. Also, the user will know the ride cost and how much time the bus driver takes to reach him/her. The user has the choice to book any number of the bus seats. A feedback of the bus driver will be shown so the user can feel

safe and comfortable. When the user confirms the ride, the bus driver will receive a request containing the user's place. Once the bus driver accepts it, the remained time for the bus arrival will be shown for both the user and the driver.

Figure 2.3. *User's request*

- **Dealing with Database**

  The Database filters Klax's drivers by Activation, Reach time and Number of seats. Then the DB chooses the driver, sends request and waits 1m for response.

- **Acceptance scenario**

  The user and the driver are tracking each other through the distance between them and the driver contact information. Then they wait for the selected duration and check payment.

  The database updates number of seats.

- **Cancellation scenario**

  If the driver cancels the user's request, The database will repeat pairing for the same request.

If the user cancels the request after response time, The system will deduct the trip payment from the credit of the user's wallet.

3. **Payment System**

   A system that allows financial transaction among users and drivers using many different payment methods such as: MasterCard, Visa, Vodafone Cash and Fawry.

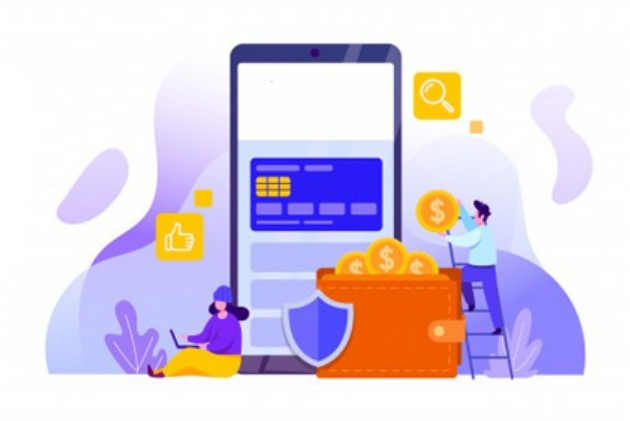


Figure 2.4. *User's Wallet*

- **User can add credits to his account using one of his personal financial credits:**

  User Login to his financial account. He can add a specific amount of credits by informing current balance and resulting's balance. Then, money is transferred from his financial account and credits are added to his account.

- **Loyal users can borrow money monotonically:**

  User sends a request to borrow. The system then checks if this user used

the application long enough using his number of trips and his rate. After making sure money is paid back after several times, user can borrow a larger amount of money

- **User can track his status such as:**
  - His current balance
  - His information about previous 10-20 rides
  - The last 10 money transfer transactions
  - The sent complains sent and received answers
- **User can send a request to loan credit from one of his contacts:**

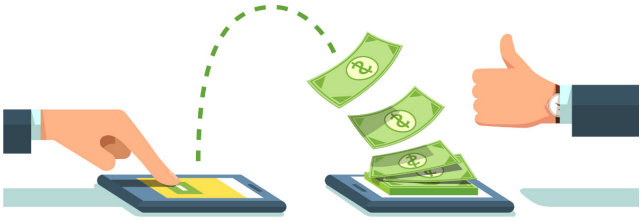


Figure 2.5. *Loaner and borrower*

Borrower asks for a transfer money request.He fills in the necessary information about the amount to be taken and the chosen contact/s. The system process the request and checks if his contact/s have this amount of money. Then, System sends a response with suitability of the chosen contact/s. Borrower is then asked to choose one or more contact to send a request to. Application then sends a request to the loaner. And when loaner sends a response, borrower balance is incremented with the requested amount.

Loaner receives a notification of a request with details about the

borrower and the amount of money. He can either accept the request or reject it. If he accepts the borrower's request, Money will be sent. Loaner will be receiving a success notification informing him about the completion of the transaction and his new balance.

- **User can send complains about a specific trip 15-20 minutes after the trip was made:**

  The user is asked to pick a trip he made. A collecting-information screen appears where user is able to explain his problem. Then his complain is sent to the system. After handling the complaint, the result is sent to the client as a notification/message to his phone.

Figure 2.6. *User's Complain*

2.3.3 Driver Software Platform

The driver system is composed of 3 essential sections as following:

1. **Home Model**

   - **Register Form**

     If a bus driver wants to join Klax, he has to sign up via Email, Name,

Password and Phone Number. Then check all needed conditions and tests for applying. Once he gets all conditions and tests successfully done, he gets An Account. If the driver account allows Multiple Route Lines, he has to mention them all and mention the activated one.

- **Private Trips**.

  a Functionality for the driver to allow passengers to contact him for private trips. This includes him station-specific lines/places that he is able to travel to. If it allows Private Rides, he has to mention the destinations and the cost.

Figure 2.7. *Private Trips*

2. **Pairing and Tracking**

   - **Availability**

     The Driver can change its state to whether be available for others see or not.

   - **Chairs Counter**.

     The Driver should be able to change its current used chairs. It does

Figure 2.8. *The driver is online*

so asking the driver to add manually offline users from the app, while updating the online users via the app itself and pushing notification to the driver considering the update.

- **Trip Ended Mechanism**.

  The system to recognize the end of a trip by providing trip feedback from the passenger, a trip is considered done.

3. **Payments and Complains Model**

- **Earning Statistics**.

  Driver can see his earnings for the past week/month.

- **Feedback System**

  Driver should be able to see continuously his feedback.

Figure 2.9. *User's feedback*

2.3.4 Admin Software Platform

It is a website that controls the whole API and the notifications sent to users. The site provides the following features:

- Statistics:
  - It monitors users, drivers and trips numbers.
  - Geographic Distribution of passengers according to their governs.
  - Application Monthly Revenue.
- Offers:

  Makes new offers and promo-codes to users (discounts, free trips, free cash).
- Passenger:

  Views passengers information.
- Tracks driver:

  Gets every driver live location and make sure he is committed to his route
- Complains:

  Views passengers complaints and send response.
- Lines:

  Adds lines and stations to the application whenever we decide.

2.3.5 Database

Database is divided into Persistent Database and Real-time Database.

- **Persistent Database (Mongo Db)**
  - Slower.
  - Used to store the passenger's data, the driver's data and the lines' data.
  - Accessed only by the server.
  - Can't be accessed by the user unless he/she sends a request to the server, then the server accepts this request.
- **Real-time Database (Firebase)**
  - Used to store the temporary data that need to be updated (The empty seats'number and The driver's location).
  - Can be accessed by the server, the passenger and the driver.

2.4 *Technologies*

2.4.1 Node.JS

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Features of Node.JS

1. **Asynchronous and Event Driven**

   All APIs of Node.js library ‘re asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
2. **Very Fast**

   Being built on Google Chrome’s V8 JavaScript Engine, Node.JS library is very fast in code execution.
3. **Single Threaded but Highly Scalable**

   Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same

program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server

4. **No Buffering**

   Node.JS applications never buffer any data. These applications simply output the data in chunks.

The Node.JS Run-time

The source code written in source file is simply JavaScript. The Node.js interpreter will be used to interpret and execute your JavaScript code. Node.js distribution comes as a binary installable for SunOS, Linux, Mac OS X, and Windows operating systems with the 32-bit (386) and 64-bit (amd64) x86 processor architectures.

Node.JS RESTful API

- **REST architecture**

  REST is web standard based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000. A REST Server simply provides access to resources and REST client accesses and modifies the resources using HTTP protocol. Here each resource is identified by URIs/ global IDs. REST uses various representation to represent a resource like text, JSON, XML but JSON is the most popular one.

- **HTTP Methods**

  - **GET** This is used to provide a read only access to a resource.

  - **PUT** This is used to create a new resource.
  - **POST** This is used to update an existing resource or create a new resource.
  - **DELETE** This is used to remove a resource.

- **RESTful Web Services**

  A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., communication between Java and Python, or Windows and Linux applications) is due to the use of open standards.

  Web services based on REST Architecture are known as RESTful web services. These web services uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, which provides resource representation such as JSON and set of HTTP Methods. [5]

2.4.2 Flutter

Flutter is an app SDK for building high-performance, high-fidelity apps for iOS, Android, web, and desktop from a single codebase. The goal is to enable developers to deliver high-performance apps that feel natural on different

platforms. We embrace differences in scrolling behaviors, typography, icons, and more. Apps are written in Dart, which looks familiar if you've used a language like Java or JavaScript. Experience with object-oriented languages is definitely helpful.

Features of Flutter

- **Be highly productive**
  - Develop for IOS and Android from a single codebase
  - Do more with less code, even on a single OS, with a modern, expressive language and a declarative approach
  - Prototype and iterate easily
    * Experiment by changing code and reloading as your app runs (with hot reload)
    * Fix crashes and continue debugging from where the app left off
- **Create beautiful, highly-customized user experiences**
  - oBenefit from a rich set of Material Design and Cupertino (iOS-flavor) widgets built using Flutter's own framework
  - oRealize custom, beautiful, brand-driven designs, without the limitations of OEM widget sets

Core Principles

Flutter includes a modern react-style framework, a 2D rendering engine, ready-made widgets, and development tools. These components work together to help you design, build, test, and debug apps. Everything is organized around a few

core principles.

Everything's a widget

Widgets are the basic building blocks of a Flutter app's user interface. Each widget is an immutable declaration of part of the user interface. Unlike other frameworks that separate views, view controllers, layouts, and other properties, Flutter has a consistent, unified object model: the widget. A widget can define:

- a structural element (like a button or menu)
- a stylistic element (like a font or color scheme)
- an aspect of layout (like padding)

Widgets form a hierarchy based on composition. Each widget nests inside, and inherits properties from, its parent. There is no separate "application" object. Instead, the root widget serves this role. You can respond to events, like user interaction, by telling the framework to replace a widget in the hierarchy with another widget. The framework then compares the new and old widgets and efficiently updates the user interface. [6]

2.4.3 MVC

The Model View Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development Framework to create scalable and extensible

projects.

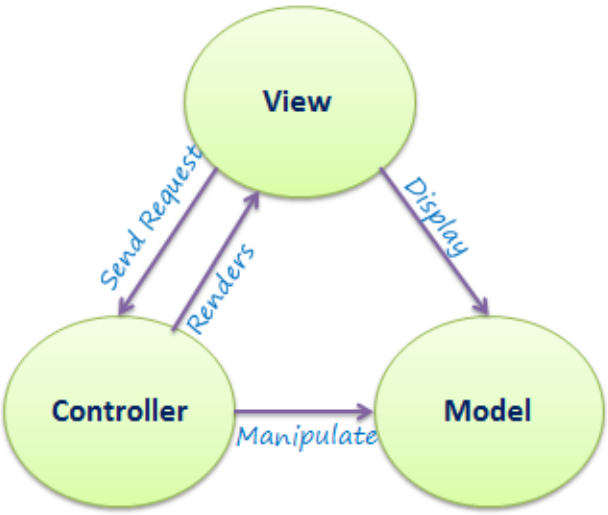


Figure 2.10. *MVC Architecture*
[7]

Model

The Model component corresponds to all the data related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model

component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

ASP.NET MVC

ASP.NET supports three major development models: Web Pages, Web Forms and MVC (Model View Controller). ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with the existing ASP.NET features, such as master pages, authentication, etc. Within .NET, this framework is defined in the System Web MVC assembly. The latest version of the MVC Framework is 5.0. We use Visual Studio to create ASP.NET MVC applications which can be added as a template in Visual Studio. [8]

MVC Flow Diagram

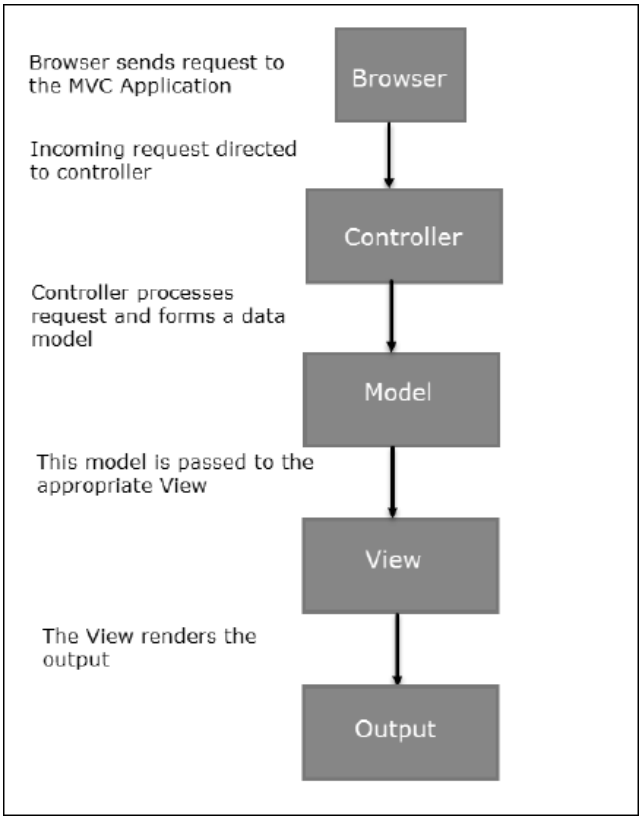


Figure 2.11. *MVC Flow Diagram*
[8]

Flow Steps

Step1

The client browser sends request to the MVC Application.

Step2

Global.ascx receives this request and performs routing based on the URL of the incoming request using the RouteTable, RouteData, UrlRoutingModule and MvcRouteHandler objects.

Step3

This routing operation calls the appropriate controller and executes it using the IControllerFactory object and MvcHandler object's Execute method.

Step4

The Controller processes the data using Model and invokes the appropriate method

using ControllerActionInvoker object

Step5

The processed Model is then passed to the View, which in turn renders the final output.

2.4.4 PayPal

PayPal provides an easy and quick way to send and request money online. You can transfer money (abroad) to family, friends, online shops, and auction sites like eBay.

Online payments

All you need to send money to family or friends is the email address of the recipient. By registering your credit card or bank account with your PayPal account you can send payments using the option. [9]

Send and Request Money

The money will be credited to the recipient's account and can then be transferred to a bank account, or used to make a payment. If you shop online and see the PayPal logo on the merchant's website, it means you can pay using PayPal. Just select PayPal at the checkout. You will be asked to log in to your account and confirm the payment. Your financial information will never be visible to sellers or online shops.

To receive money online

Anyone can send money to you using your email address. Your email address is linked to your personal PayPal account. You will receive an email notification whenever you receive a payment and the payment will be shown on your account. Online shops that offer PayPal as a payment option can post the PayPal logo on their

website. There are various ways to offer PayPal as a payment option, like Pay Now and Shopping Cart. If you receive a payment through PayPal, PayPal 'll send you a notification by email. The money will be credited to your PayPal balance. You can transfer the money to your bank account or use it to make a payment yourself.

Fees

Opening a PayPal account is free. Fees will be charged depending on the payment you make:

- **Personal payments**

  Payments to friends and family are free provided you use your PayPal balance or bank account to send these payments. If you use your credit card, the recipient will be charged the associated fees. However, you as sender can state that you will pay these fees.

- **Commercial payments**

  If you buy an item, the recipient will be charged the associated fees.

Benefits of Using PayPal

1. **Speed**

   PayPal transfers between sellers are instant, and transfers from PayPal accounts to bank accounts can take as little as 24 hours.
2. **Affordability**

   The fee to use PayPal is 30 cents per transaction, plus 3 per cent of the total amount of the transaction.

2.4.5 Stripe

Stripe builds the most powerful and flexible tools for internet commerce. Whether you're creating a subscription service, an on-demand marketplace, an e-commerce store, or a crowdfunding platform, Stripe's meticulously designed APIs and unmatched functionality help you create the best possible product for your users. Millions of the world's most innovative technology companies are scaling faster and more efficiently by building their businesses on Stripe. Platforms use Stripe Connect to accept money and pay out to third parties. Connect provides a complete set of building blocks to support virtually any business model, including on-demand businesses, e-commerce, crowdfunding, and travel and events. [10]

Features

1. **Integrate quickly**

   Building the payments infrastructure for your platform used to be a big undertaking no longer. Take advantage of pre-made UI components to launch faster and simplify operations. Sign up new users on your platform and get them paid quickly.
2. **Customize**

   Connect is API-first and lets you design the best experience for your platform. You can customize onboarding, set payout timing, allow complex money movement, and get integrated financial reporting. You own the experience from end to end.

Routing Payments

Connect will automatically track balances, batch earnings into payouts, time transfers with local cutoffs, and retry failed transfers. You can also incorporate advanced flows like Account Debits, one-to-many payments, and others. Connect's payout engine lets you specify payout timing for your users, and includes Instant Payouts, which allows your users to receive funds within minutes. Connect lets you get recipients paid faster and removes errors and reconciliation work.

2.4.6 Angular 9

Angular is a way to build applications for any deployment target. For web, mobile web, native mobile and native desktop.

It builds features quickly with simple, declarative templates. Extend the template language with your own components and use a wide array of existing components. Get immediate Angular-specific help and feedback with nearly every IDE and editor. All this comes together so you can focus on building amazing apps rather than trying to make the code work.

Figure 2.12. *Angular*

Features [11]

- **Cross Platform**
  - **Progressive Web Apps**

    Use modern web platform capabilities to deliver app-like experiences. High performance, offline, and zero-step installation.
  - **Native**

    Build native mobile apps with strategies from Cordova, Ionic, or NativeScript.
  - **Desktop**

    Create desktop-installed apps across Mac, Windows, and Linux using the same Angular methods you've learned for the web plus the ability to access native OS APIs.
- **Speed And Performance**
  - **Code Generation**

    Angular turns your templates into code that's highly optimized for today's JavaScript virtual machines, giving you all the benefits of handwritten code with the productivity of a framework.
  - **Universal**

    Serve the first view of your application on Node.js®, .NET, PHP, and other servers for near-instant rendering in just HTML and CSS. Also paves the way for sites that optimize for SEO.
  - **Code Splitting**

    Angular apps load quickly with the new Component Router, which

delivers automatic code-splitting so users only load code required to render the view they request.

- **Productivity**
  - **Templates**

    Quickly create UI views with simple and powerful template syntax.
  - **Angular CLI**

    Command line tools: start building fast, add components and tests, then instantly deploy.
  - **IDEs**

    Get intelligent code completion, instant errors, and other feedback in popular editors and IDEs.
- **Full Development Story**
  - **Animation**

    Create high-performance, complex choreographies and animation timelines with very little code through Angular's intuitive API.
  - **Accessibility**

    Create accessible applications with ARIA-enabled components, developer guides, and built-in a11y test infrastructure.
  - **Testing**

    With Karma for unit tests, you can know if you've broken things every time you save. And Protractor makes your scenario tests run faster and in a stable manner.

Angular 9 Ivy

Angular 9's compiling and rendering engine is known as Ivy. The older versions of Angular made use of View Engine. The bundle size produced by the View Engine is very large but with Ivy, this bundle has considerably reduced thereby helping Angular overcome its bundle issues.

Benefits

- Smaller bundle size.
- Very helpful when it comes to debugging:

  With Angular 9, you will not have to debug through the framework, but rather, you can do it on the component itself which helps you debug your code instantly.

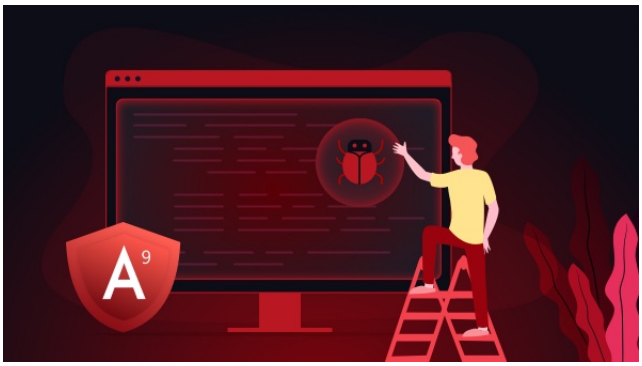


Figure 2.13. *Debugging*

- Faster testing.
- Improved CSS class and style binding.
- Improved type checking.
- Improved build errors.

- Improved build times, enabling AOT on by default.

CHAPTER

3

Mobile Application

3.1 *Intro*

In this chapter, we discuss the application in terms of front-end, Design and scenarios that include the app. As mentioned before, we use the Google Mobile UI (Flutter), It is famous for its unique design everything needed for developing mobile apps. Flutter also has widgets for Material Design and Cupertino that allow developers to easily render the UI on both IOS and Android platform. Before developing a new mobile app, you need to design it first. It's critical to plan every step, and at some point, you might want to retreat and examine what you're building.

3.2 *Mobile App*

3.2.1 App Design

There are two phases of any mobile app design.

1. Mobile app design strategy
2. App design process

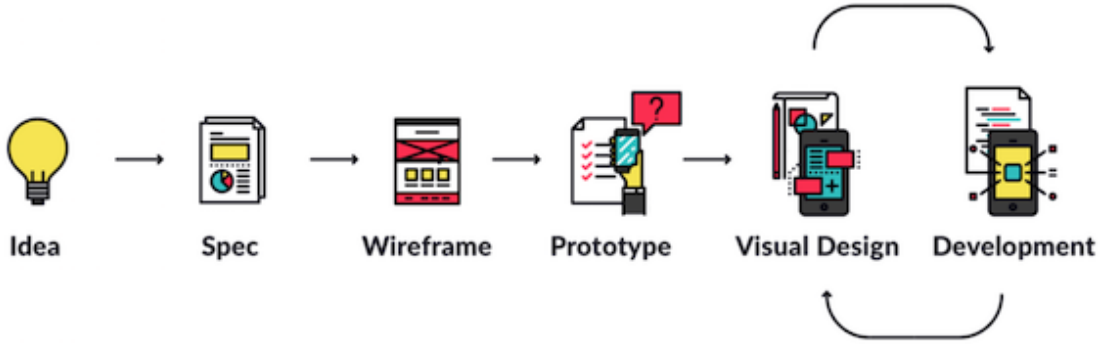
Mobile App Design Strategy It starts with a strategy. It defines the future and the path to reach your destination. Developing a mobile strategy links back to the company strategy and has four stages:

1. Understand the business strategy
2. Business mobile app strategy
3. App strategy
4. Product management strategy

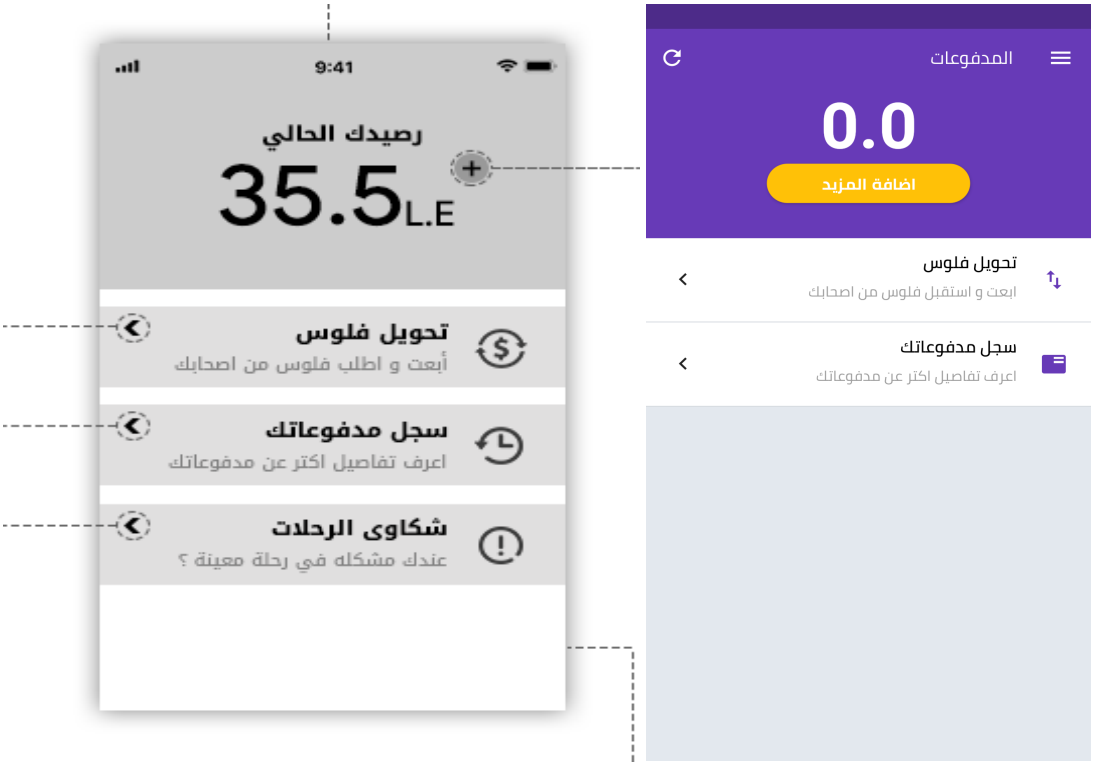
App design process

The basic app design process consists of following steps:

1. idea
2. spec
3. wireframe
4. Prototype
5. UI design
6. App development



(a) *The design process]The design process*



(b) *wireframe* **(c)** *prototype*

Figure 3.1. *wireframe and prototype*

3.3 *Mobile Front-end*

3.3.1 Introduction

Over the past ten years, the number of smartphones used has exceeded 2.5 billion. Every year, consumers spend 380 billion dollar on new devices. Each of them contains applications that simplify life, help calculate taxis or food, call them, and order them. As for this application, it is intended for all age groups, it is easy to use and used for both Android and IOS, and anyone can download it.

The front end, also known as the client-side through which all our customers can use our apps and enjoy their services.

Not only are the beautiful screens, but the application connects to the server to be able to send and receive data to provide the desired service. Fetching data from the internet is necessary for most apps. Luckily, Dart and Flutter provide tools, such as the HTTP package, for this type of work.

Mobile App Screens:

login Page: to enter the email and password also checks that the user must be authorized.it also contain sing up page if the user does not have account(this page in each app).

if the user forgot the password, he can recovery by phone number or email.

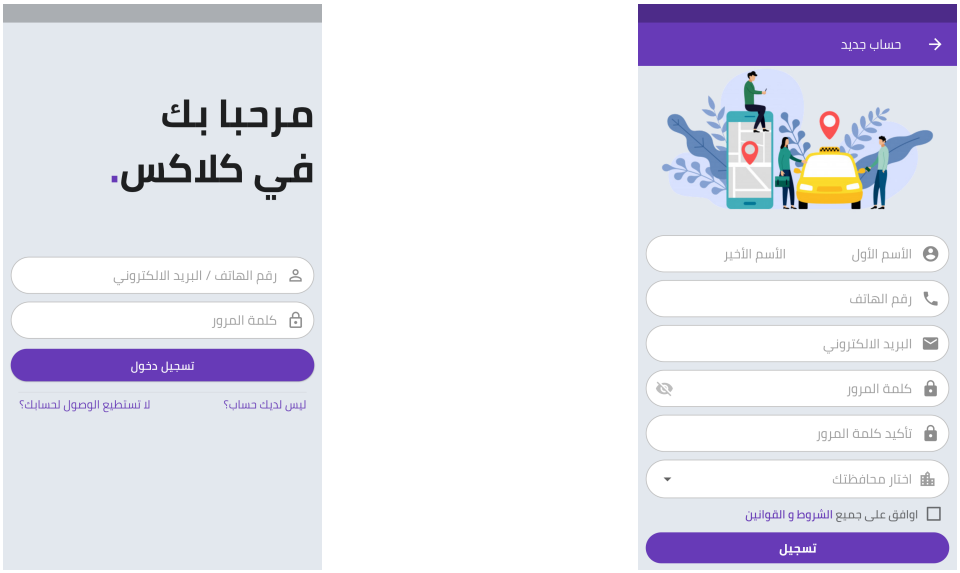


Figure 3.2. *Login and Registration Page*

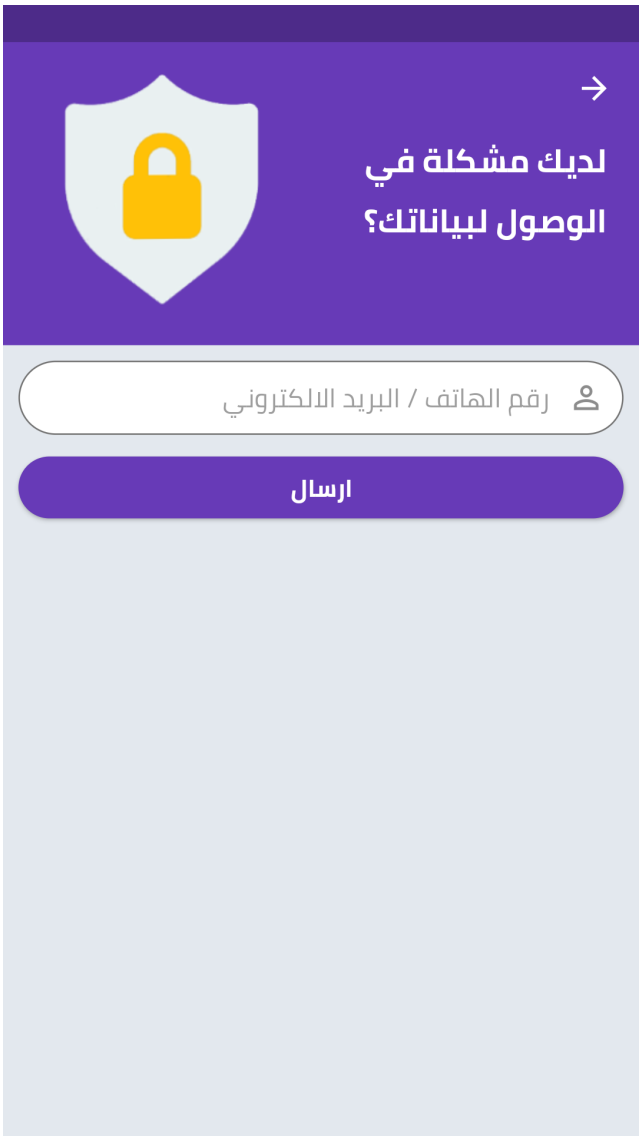
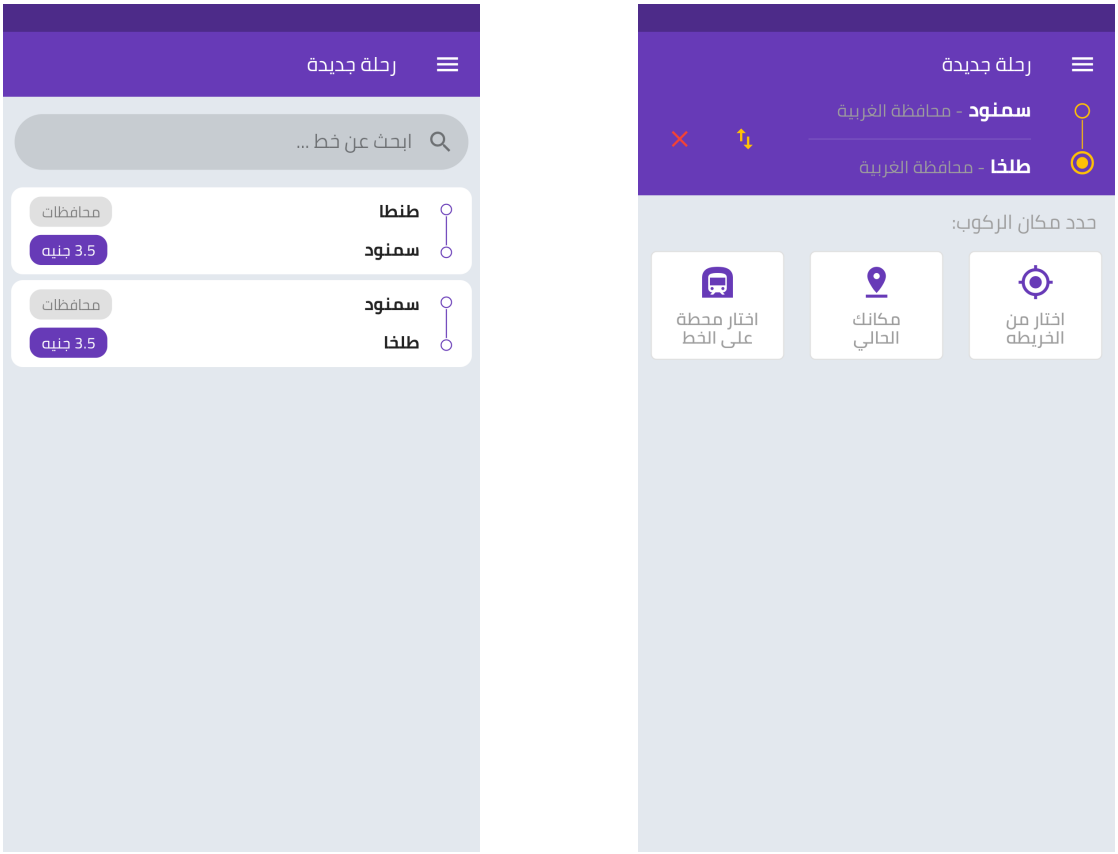


Figure 3.3. *Reset password page*

home Page:the user's page contains the available lines, the number of seats the user can use to book a new ride, select location, and payment way. On the other hand, the driver's homepage contains a choice of the number of seats available to him that are ready to receive user's new ride notifications can be received

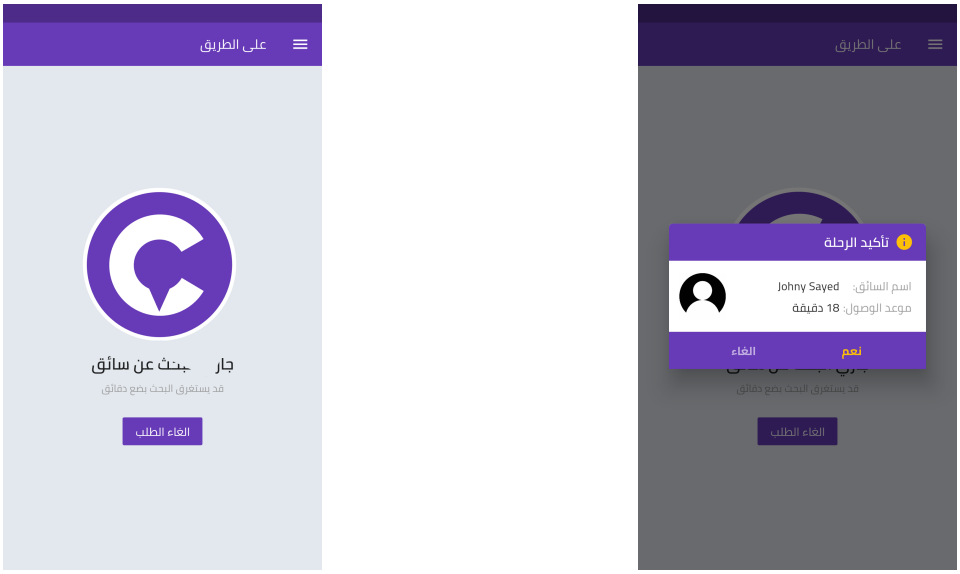


(a) *Choose the available line* **(b)** *Confirm the location*

Figure 3.4. *User Home page*

Pairing Page:

The processes to find the closest driver with requested seats until the driver accepts the user requests which showing until 10 seconds for each driver and the driver information appear to the user who can accept this driver or not available driver now.



(a) *Searching for drivers* **(b)** *The user finds driver*

Figure 3.5. *Pairing page*

tracking page :where the driver and the user are connected via google map until the driver arrived at the desired location.

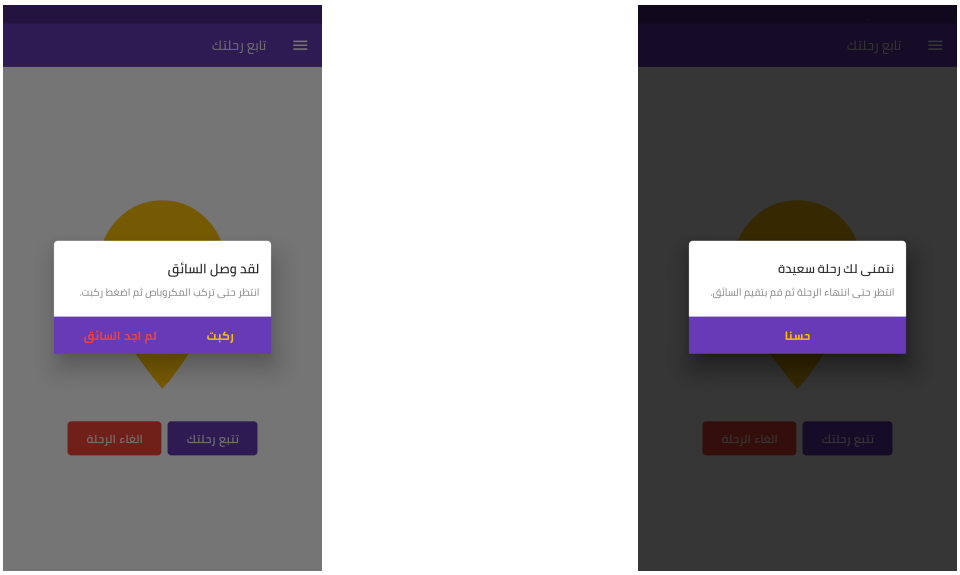


Figure 3.6. *Tracking page*

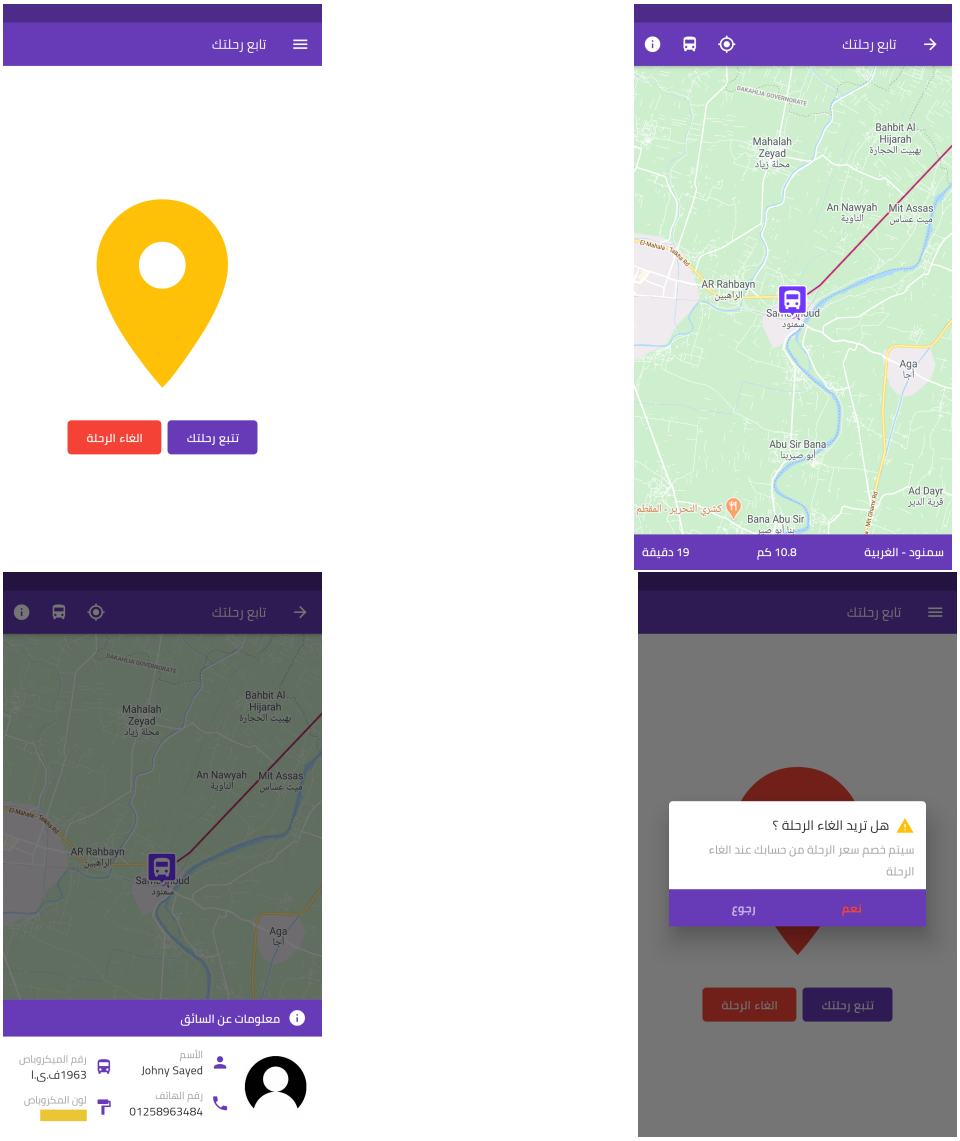


Figure 3.7. *Tracking page*

Rating Page: for both the driver and the user can evaluate their ride and this appears after the end of each ride

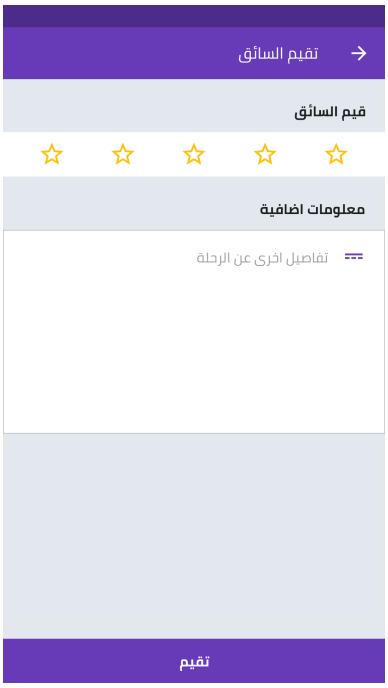


Figure 3.8. *Rating Page*

Profile Page:

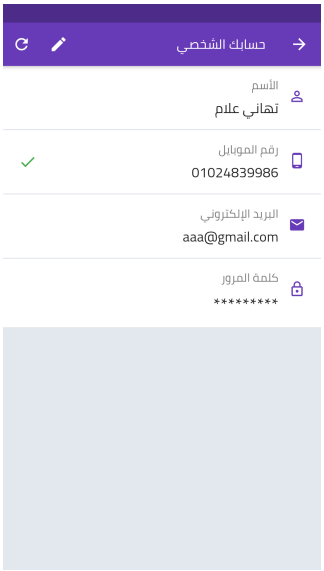


Figure 3.9. *Profile page*

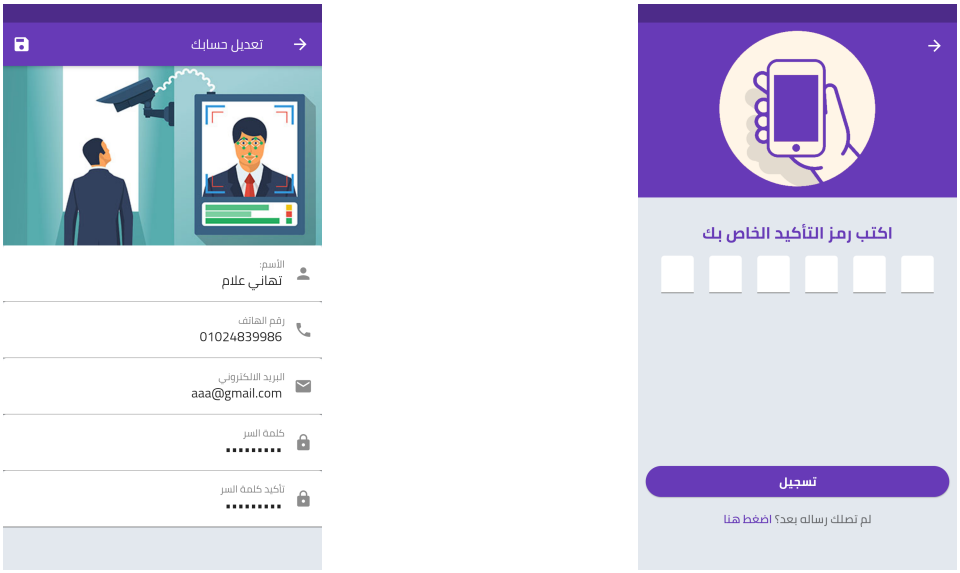


Figure 3.10. *Editing profile page*

Family services Page:

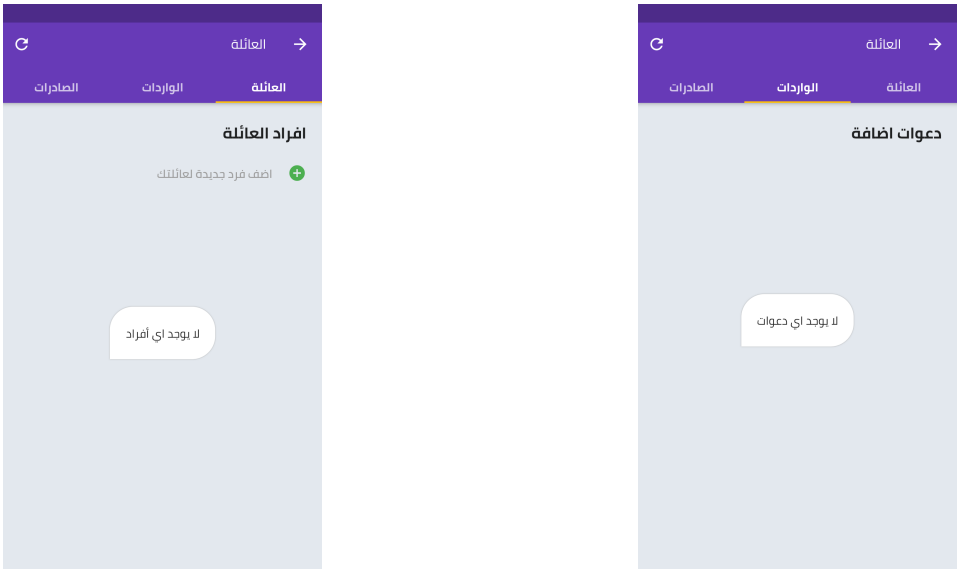
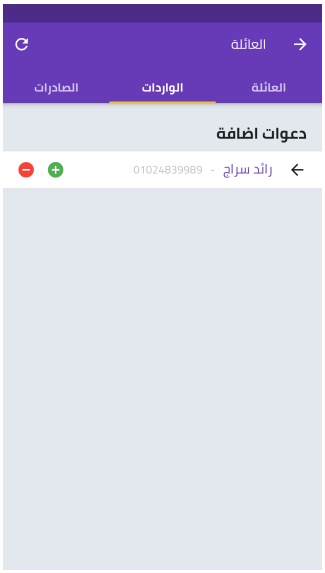
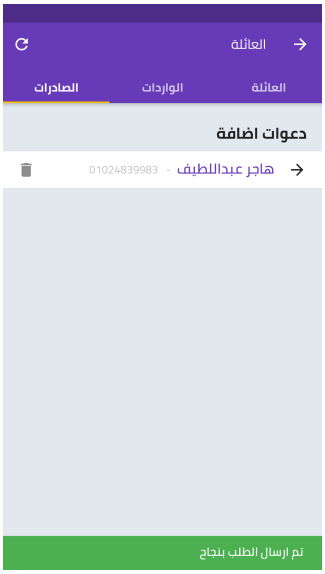
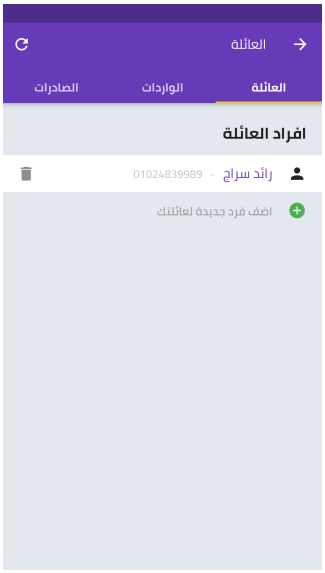


Figure 3.11. *Family services Page*







- The application provides for users Family services which can make an invitation for someone who uses the app

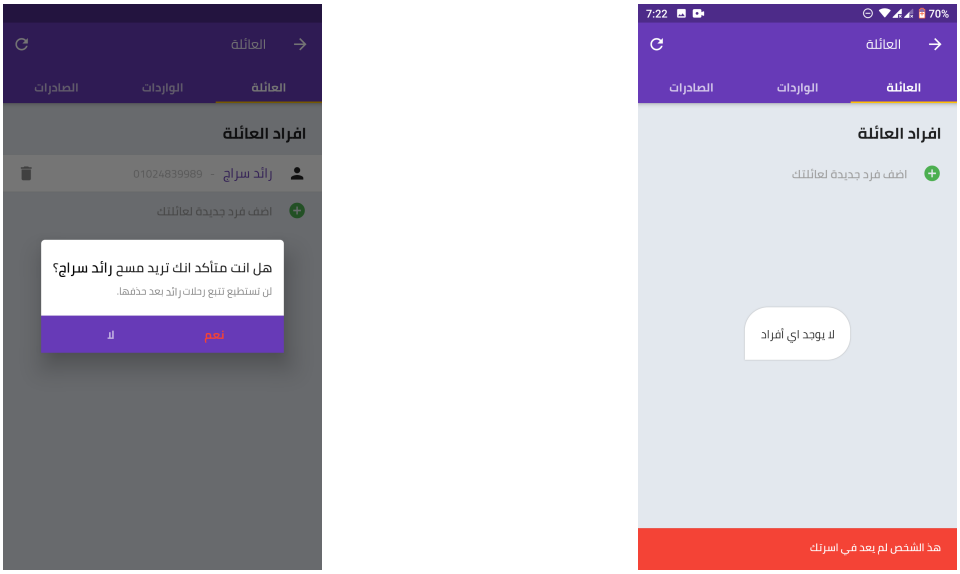


Figure 3.14. *Family services Page*

Payment Page:

It allows the user to use his own balance for the payment process, the ability to transfer money and borrow some money too.

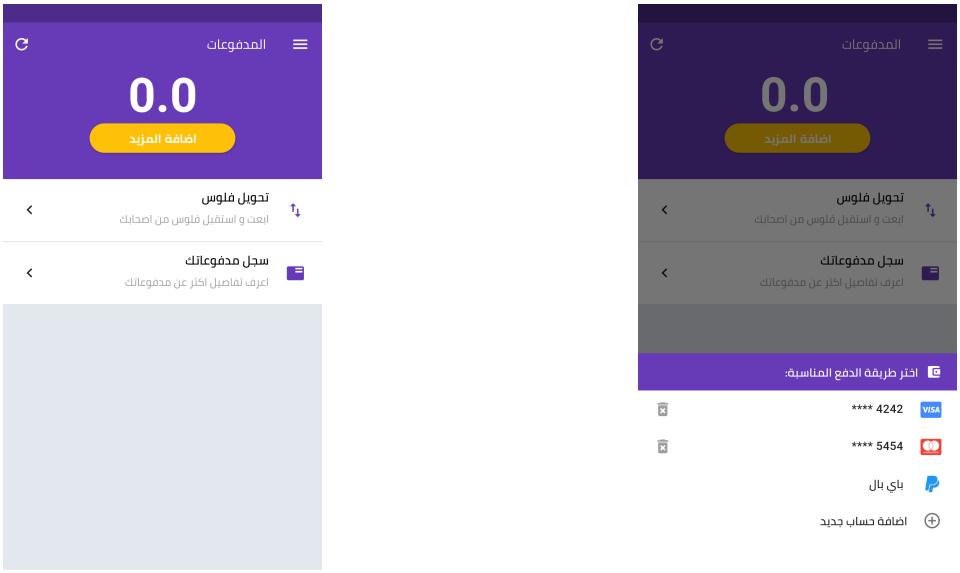


Figure 3.15. *Add credit Page*

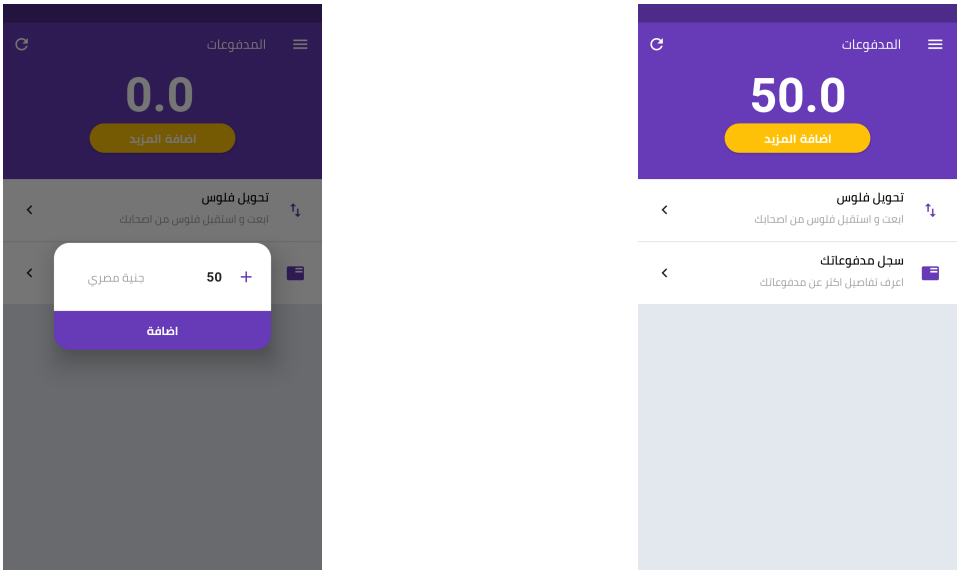


Figure 3.16. *Payment services history Page*

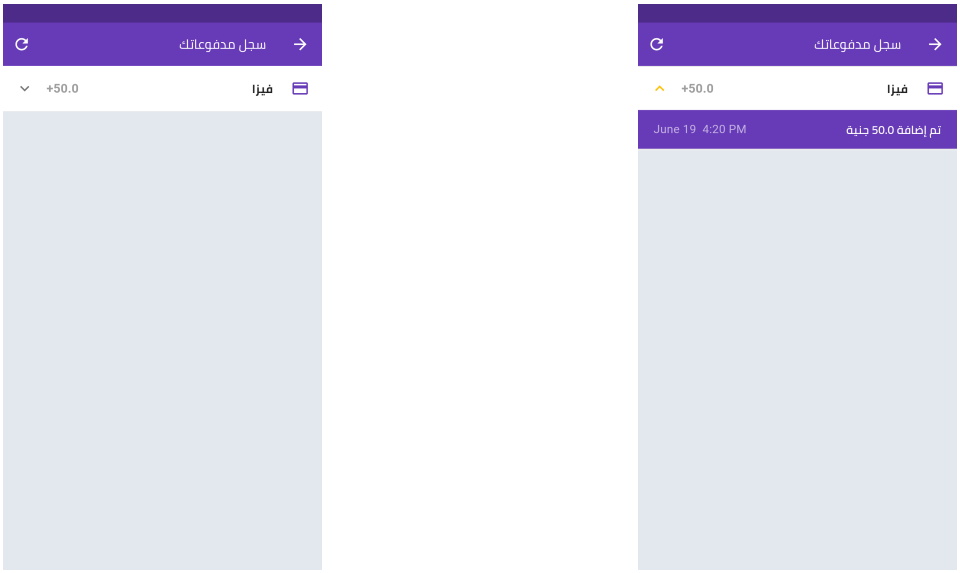


Figure 3.17. *Payment services Page*

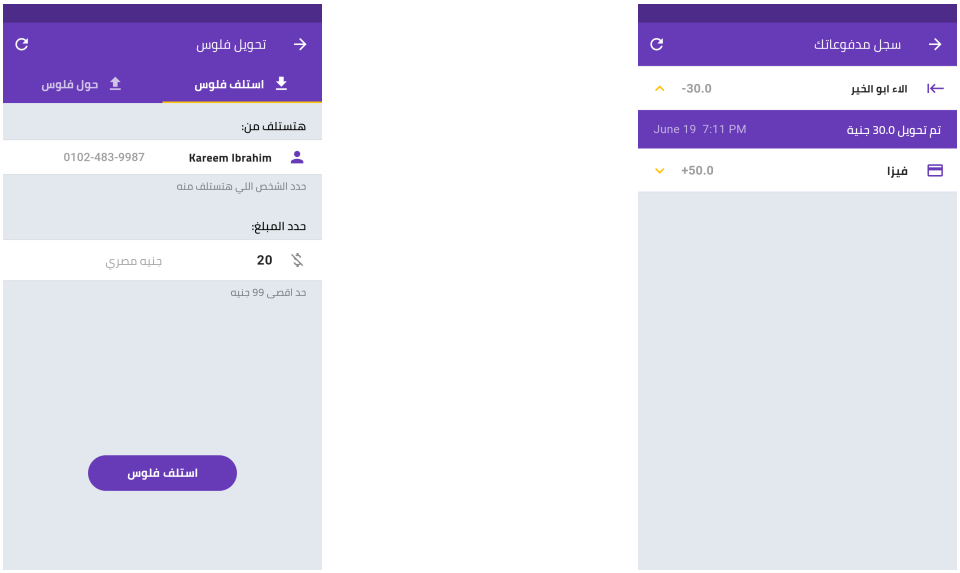


Figure 3.18. *Transfer and borrow money Page*

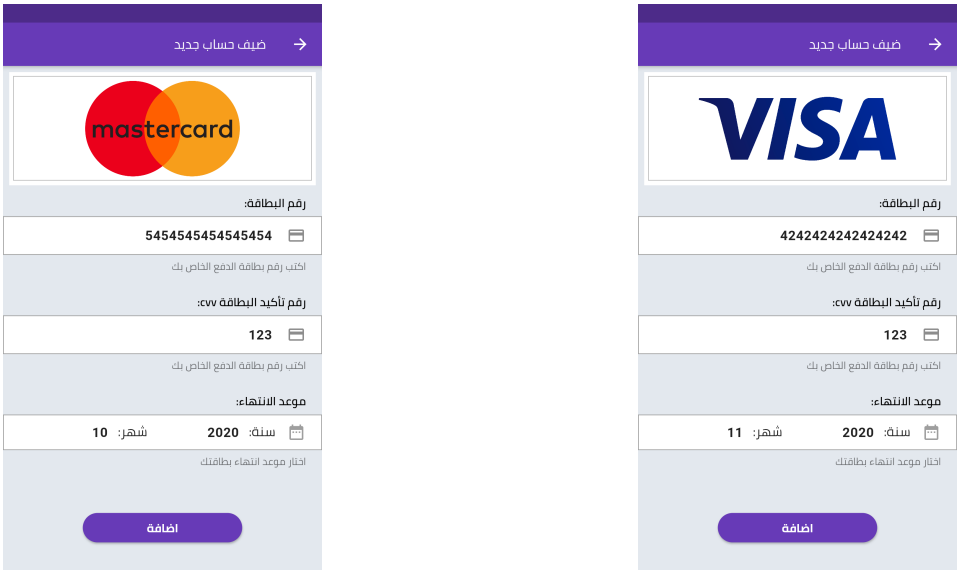


Figure 3.19. *Payment Methods page*

CHAPTER

4

Back-end Development

4.1 *Introduction to Back-end*

Front end development involves what a user sees on the screen when they open a specific URL owned by you. Even in a completely static environment (with only HTML/CSS), when someone opens a website, some server on the planet needs to respond to you with those HTML and CSS files.

That server is just a computer, just like the one you use yourself to browse the internet. But it has been tuned for performance, and doesn't have unnecessary components like a mouse or keyboard attached. And it sits with tons of other computers probably in a data warehouse.

Programming those computers in some special way is called back end development.

4.2 *API*

4.2.1 API Server

4.2.1.1 History of REST API

An API, or ”application programming interface” is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. The developer creates the API on the server and allows the client to talk to it.

Prior to REST, developers used SOAP to integrate APIs. To make a call, developers handwrote an XML document with a Remote Procedure Call (RPC) call in the body. They then specified the endpoint and POST their SOAP envelope to the endpoint. In 2000, Roy Fielding and a group of developers decided to create a standard so that any server could talk to any other server. He defined REST and the architectural constraints explained above in his 2000 PhD dissertation at UC Irvine. These universal rules make it easier for developers to integrate software.

Salesforce was the first company to sell an API as part of its “Internet as a Service” package in 2000. However, few developers were actually able to use the complicated XML API. EBay built a REST API, which expanded its market to any site that could access its API. This caught the attention of another ecommerce giant, and Amazon announced its API in 2002.

Flickr launched its own RESTful API in August 2004, enabling bloggers to easily embed images on their sites and social media feeds. Facebook and Twitter both released their APIs in 2006, buckling under the pressure of developers who scraped the sites and created "Frankenstein" APIs. When Amazon Web Services (AWS) helped launch the cloud in 2006, developers were able to access data space in minutes using AWS's REST API, and the request for public APIs quickly escalated. Since then, developers have embraced RESTful APIs, using them to add functionality to their websites and applications. Today, REST APIs are considered the "backbone of the Internet."[4]

4.2.1.2 Work of REST API

REST is an architectural style that uses simple HTTP calls for inter-machine communication instead of more complex options like CORBA, COM+, RPC, or even SOAP. Using REST means your calls will be message-based and reliant on the HTTP standard to describe these messages.

Using the HTTP protocol means REST is a simple request/response mechanism. Each request returns a subsequent response. You can see what a simplified request and response look like below:

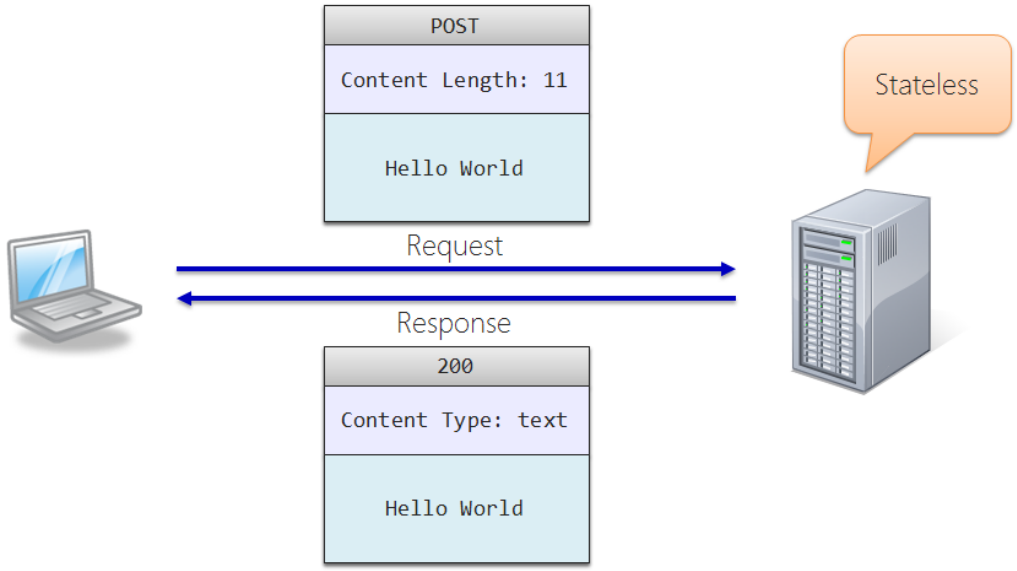


Figure 4.1. *Request/Response Mechanism*

Requests are made up of a verb (POST, in this example), headers that describe the message, and a body (Hello World, in this example). The request is a message that describes what you want the server to accomplish. Likewise, the response consists of three pieces: a status code (200), headers describing the response and the body itself.

HTTP Verbs describe the type of operation:

- GET: Retrieve a resource.
- POST: Create a resource.
- PUT: Update a resource.
- DELETE: Delete a resource.

On the Web, the most common verb is GET. This is because the main purpose of a Web page's function is to request different resources that make up a page.

In REST-based APIs, we leverage these verbs to describe the types of operations we want.

4.2.1.3 Reasons of Using REST API

You could imagine that this API is accessible from a client-side Web project, an iOS app, an IoT device and even a Windows Phone. This allows you to build the infrastructure for your organization with fewer worries about the longer-term marrying to a particular client-side stack. The server lives longer than the client. It always does.

Another key idea in this architectural philosophy is that the server supports caching and is stateless. These ideas are very important to how REST works. Caching is important, as if multiple requests for the same resource are requested, caching of the result of the request means that the scalability of the server should increase.

Likewise, statelessness is about making sure that calls to the API aren't tied to a particular server, which increases the likelihood of building scalable server infrastructures. Because the server is stateless, every call to it must include all the data required to execute the request. The HTTP protocol is connectionless (there's no guarantee that your request is going to the same server, or how long it will be between those calls to the server).[12]

4.2.2 Authentication and Authorization

In Our Project, We needed to achieve Security so that we use authentication and authorization.

Authentication and authorization are both common terms in the world of identity and access management (IAM). While they might sound similar, both are distinct security processes, and understanding the difference between the two is key to successfully implementing an IAM solution. Both authentication and authorization are required to deal with sensitive data assets. Without any of them, you are keeping data vulnerable to data breaches and unauthorized access.

Authentication is the act of validating that users are who they claim to be. Passwords are the most common authentication factor—if a user enters the correct password, the system assumes the identity is valid and grants access.
Other technologies such as One-Time Pins, authentication apps, and even biometrics can also be used to authenticate identity. In some instances, systems require the successful verification of more than one factor before granting access. This multi-factor authentication (MFA) requirement is often deployed to increase security beyond what passwords alone can provide.

Every organization is trying to use the best authentication process to keep their data secure. There are so many authentication processes that can be used to validate user identity.**Given below are some of them:**

- Single Sign-On (SSO) allows users to get access to various applications through a single set of login credentials. SSO uses a federation when the user logs in into a spread across the different domains.

- Multi-Factor Authentication (MFA) uses different means of authentication. During log in with user name and password the user is asked to provide a one-time access code that the website sends to the user's cell phone. It provides a higher level of assurance during the authentication step to improve security.

- Consumer Identity and Access Management (CIAM) provides various features like customer registration, self-services account management, consent and preference management, and other authentication features.[13]

Authorization in system security is the process of giving the user permission to access a specific resource or function. This term is often used interchangeably with access control or client privilege. Giving someone permission to download a particular file on a server or providing individual users with administrative access to an application are good examples. In secure environments, authorization must always follow authentication—users should first prove that their identities are genuine before an organization's administrators grant them access to the requested resources.[14]

4.2.3 Requests Validation

In Our Application, We need to verify whether values of data come from the given request are acceptable values, So we use the Validation concept to achieve that. Validation can mean a lot of things, but in API land it generally means figuring out if the data being sent to the API is any good or not.

For the standard application we can put data validation logic in three places:

- GUI – it is entry point for users input. Data is validated on the client side, for example using Javascript for web applications.
- Application logic/services layer – data is validated in specific application service or command handler on the server side.
- Database – this is exit point of request processing and last moment to validate the data.

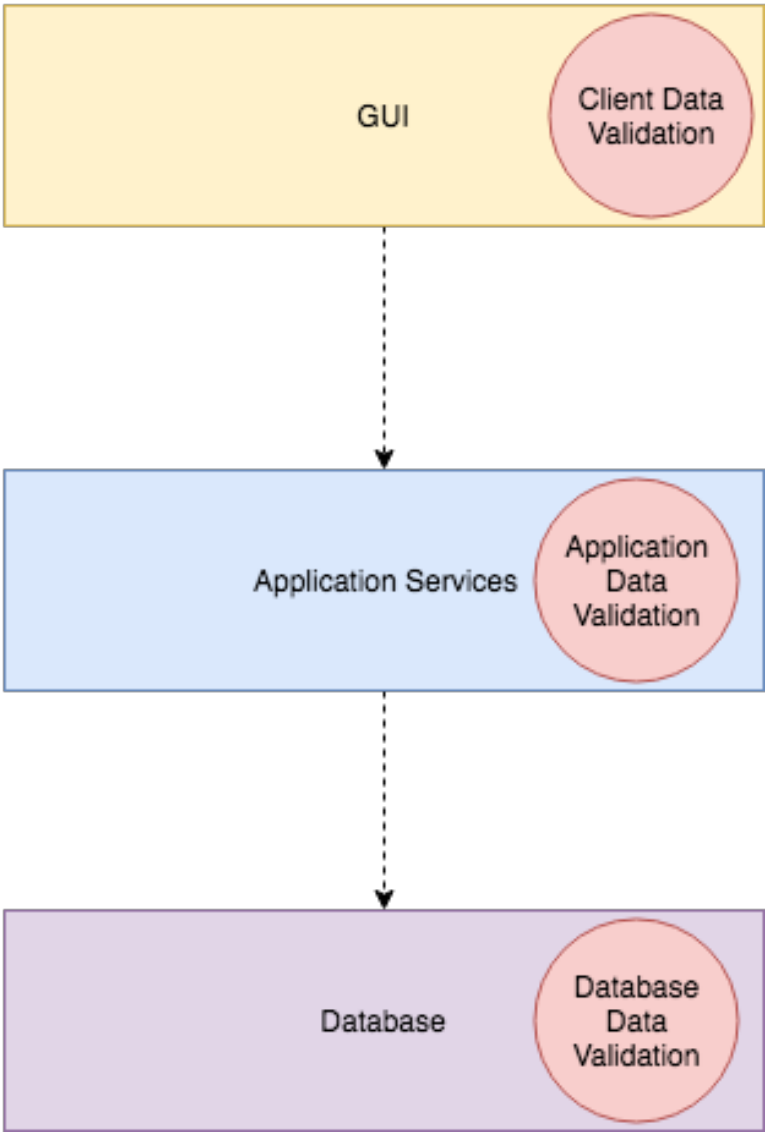


Figure 4.2. *Data validation localization*

Server-side validation has always been required and for an API is the most important of the two. An API that relies entirely on the client is going to end up with problems. Data coming from the client can never be trusted because it's impossible for the server to know what happened on the client. Even if you're developing a private API for only two known clients, there are always chances that validation in those clients breaks down; or someone will hit those APIs with curl or Postman and send some invalid stuff. Even if the database catches invalid data, the errors won't be useful.

we used Hapi Joi to achieve validation. Hapi Joi is an object schema description language and validator for JavaScript objects.
Here are some very useful validation constraints that can be attached to the base schema:

- allow(): Whitelists value or set of values that will always be valid before applying other rules. It can take values as its arguments or an array of values.
- valid(): Whitelists value or set of values as the only valid value(s) before applying other rules. It can take values as its arguments or an array of values.
- invalid(): Blacklists value or set of values that will never be valid. It does the direct opposite of allow(). It can take values as its arguments or an array of values.
- required(): Prevents the schema from allowing undefined as value.
- optional(): he schema can allow undefined as value. The schema however, will not allow a value of null. This is the default behaviour.

- raw(): This outputs the original untouched value instead of the casted value.
- strict(): This enables strict mode - which prevents type casting for the current key and any child keys. In non-strict mode, the values are casted to match the specified validation schema where possible. This is the default behaviour.
- default(): This sets a default value if the original value is undefined. In its simplest form, it takes as its first argument the value to use as default.[15]

4.2.4 Database Access

API handles requests responses between a template loaded in a web browser / mobile app and a data store. The data store can be a SQL-based database (ex: SQL Server/MySQL), or a No SQL database (ex: MongoDB), or a flat-filesystem (text/json files).
When a HTTP-based request is made from the API end-point (http/https URL), the software uses the data store to process the request and then returns a HTTP-based response to the web template which is then displayed in the web browser or the mobile app.

For Details, When a client-side requests some data to an API, passing the filter conditions, and parameters. API goes to the database (read hits the database) with the access credentials known to it (or the shared credentials passed as parameters), fetches the records from the relevant tables in the database, wraps them and tags them, and returns to the client-side as a returning parameter or sends to an address as passed in the parameters.

Being a server-side code the API uses the data store connection option / library available to the programming language in which it is written.

4.2.5 APIs Integrations

The term API integration refers to how two or more applications can be connected via their APIs to perform some joint function or to have the results which we need..using the API layer of two or more applications to make them talk to each other.

We need to integrate with other APIs to use their services and functions in our project to be more performance and efficiency.

For Example, we need to define locations through GPS so we integrated with Google Maps API to do this process.

Another Example, we need to manage debit and credit cards to make online Payment available in our application so we use Stripe to achieve this.

Below are the APIs which we used:

-Google Maps APIs:

–Distance API: to measure distances.

–Directions API: define directions of movement.

–Places API: get locations names and details.

-Firebase Cloud Messaging(FCM): to push notification.

-Stripe: to Manage debit and credit cards.

-Paypal: manage paypal banking accounts.

-SendGrid: send mails.

-Nexmo: send sms.

4.2.6 Handling and Logging Error

Our application don't run in an ideal world. Unexpected errors can happen as a result of bugs in our code or issues in the running environment.
For example, our MongoDB server may shut down, or a remote HTTP service we call may go down.
As a good developer, you should count for these unexpected errors, log them, and return a proper error to the client in a message.
- Use the Express error middleware to catch any unhandled exceptions in the "request processing pipeline".
- To pass control to the error middleware, wrap your route handler code in a try/catch block and call next().
- Adding a try/catch block to every route handler is repetitive and time consuming So Using express-async-errors module.
This module will monkey-patch your route handlers at runtime. It'll wrap your code within a try/catch block and pass unhandled errors to your error middleware.
- To log errors use winston.
- Winston can log errors in multiple transports. A transport is where your log is stored.
The error middleware in Express only catches exceptions in the request processing

pipeline. Any errors happening during the application startup (eg connecting to MongoDB) will be invisible to Express.

- Use process.on('uncaughtException') to catch unhandled exceptions, and process.on('unhandledRejection') to catch rejected promises.[16]

4.3 *Database Architecture*

4.3.1 Database Model

This section illustrates the architecture of the database using a NoSQL database. Figure 4.3 is a database ER Model which describes the tables created in the database. We used MongoDB is a cross-platform document-oriented database program.

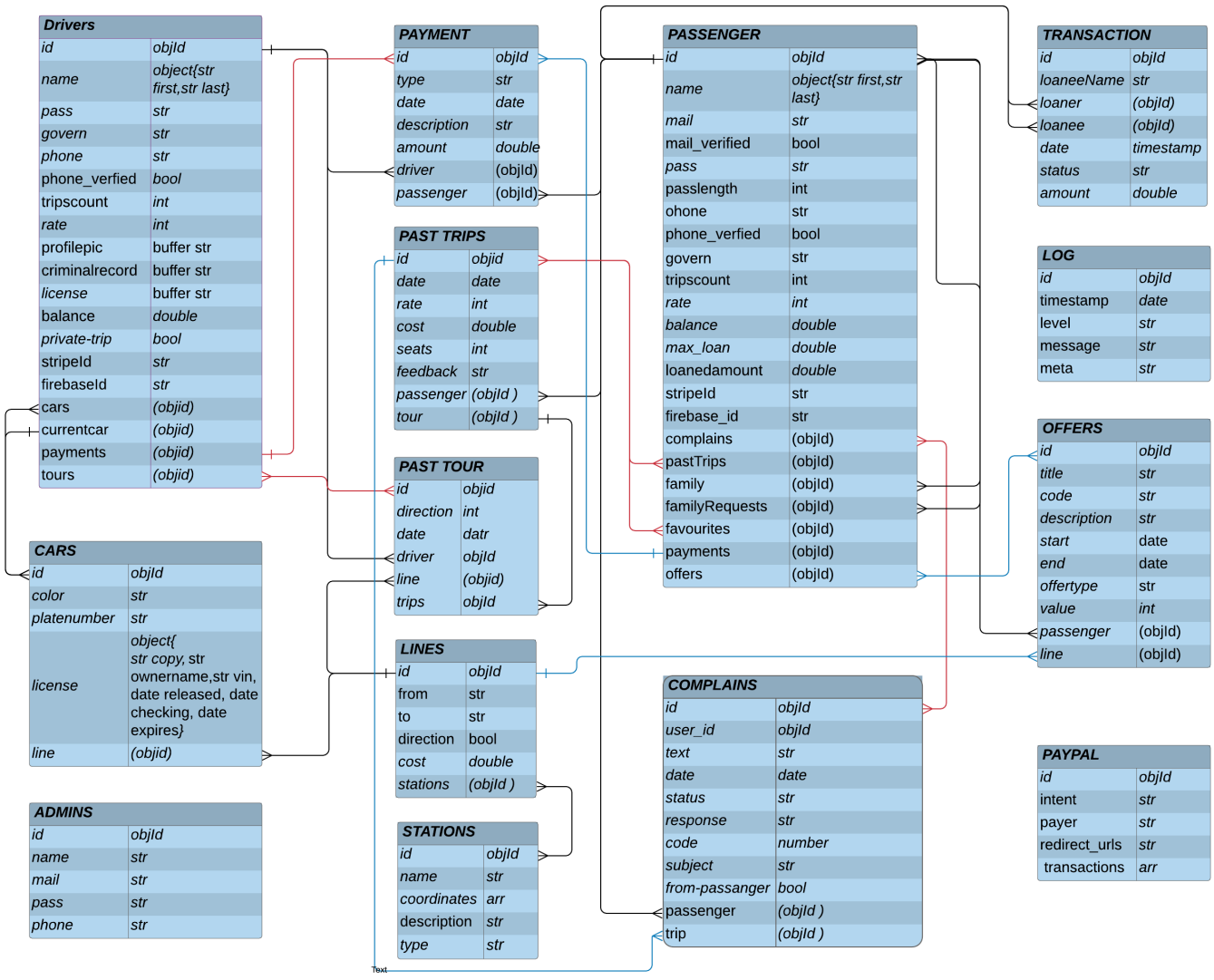


Figure 4.3. *Database ER Model*

4.3.2 MongoDB

In this section, we will show why we use MongoDB Database? When We need to Database to store data in our application, we have two options to choose ”SQL Database”, ”NoSQL Database (MongoDB)”.

The main differences between MongoDB and MySQL:
MySQL is a relational database management system (RDBMS) from the Oracle Corporation. Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access. When MySQL developers need to access data in an application, they merge data from multiple tables together in a process called a join. In MySQL, you predefine your database schema and set up rules to govern the relationships between fields in your tables.

MongoDB is a NoSQL database that stores data as JSON-like documents. Documents store related information together and use the MongoDB query language (MQL) for access. Fields can vary from document to document - there is no need to declare the structure of documents to the system, as documents are self-describing. Optionally, schema validation can be used to enforce data governance controls over each collection.

This is a comparison between them: [17]

SQL Database	**NoSQL Database (MongoDB)**
Relational database	Non-relational database

Supports SQL query language	Supports JSON query language
Table based	Collection based and key-value pair
Row based	Document based
Column based	Field based
Support foreign key	No support for foreign key
Support for triggers	No Support for triggers
Contains schema which is predefined	Contains dynamic schema
Not fit for hierarchical data storage	Best fit for hierarchical data storage
Vertically scalable - increasing RAM	Horizontally scalable - add more servers
Emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability)	Emphasizes on CAP theorem (Consistency, Availability and Partition tolerance)

Table 4.1. *Comparison between SQL and NoSQL database*

Organizations of all sizes are adopting MongoDB, especially as a cloud database, because it enables them to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale.
Thousands of companies like Bosch, Barclays, and Morgan Stanley run their businesses on MongoDB, and use it to handle their most demanding apps in areas like IoT, Gaming, Logistics, Banking, e-Commerce, and Content Management.

Development is simplified as MongoDB documents map naturally to modern, object-oriented programming languages. Using MongoDB removes the complex

object-relational mapping (ORM) layer that translates objects in code to relational tables. MongoDB's flexible data model also means that your database schema can evolve with business requirements. MySQL's rigid relational structure adds overhead to applications and slows developers down as they must adapt objects in code to a relational structure.

MongoDB can also be scaled within and across multiple distributed data centers, providing new levels of availability and scalability previously unachievable with relational databases like MySQL. As your deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing your application. In contrast, achieving scale with MySQL often requires significant custom engineering work.

MongoDB is a great choice if you need to:
Represent data with natural clusters and variability over time or in its structure Support rapid iterative development. Enable collaboration of a large number of teams Scale to high levels of read and write traffic. Scale your data repository to a massive size. Evolve the type of deployment as the business changes. Store, manage, and search data with text, geospatial, or time series dimensions.[18]

4.3.3 Firebase Realtime Database

We faced a problem with our need to continuously update data as we needed process data in real-time.

As the data of drivers and trips in our application need to be updated in real-time like available drives, the number of seats, and locations of drivers.

So we use firebase real-time database to solve this problem.

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.[19]

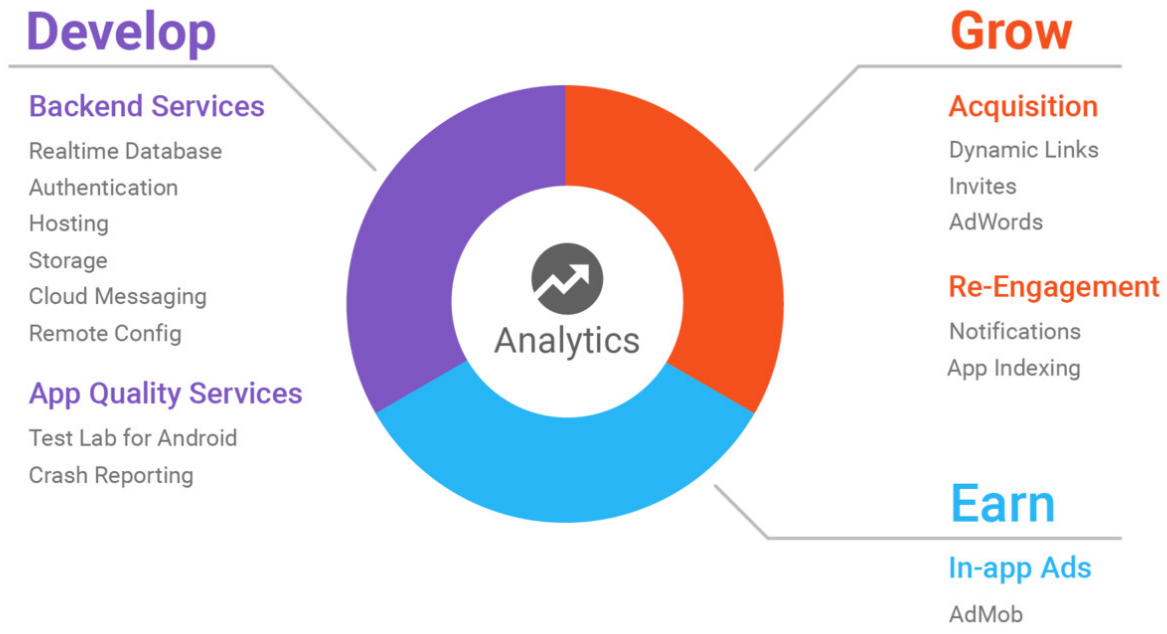


Figure 4.4. *Firebase Features*

The main goal of the system is to provide a secure, reliable and fast way to synchronize data with a minimal amount of coding effort on the part of the developer. The database system is also designed to scale to support millions of users.

The database is referred to as being “realtime” because the speed with which the data is synchronized across clients is probably as close to realtime as is currently achievable (taking into consideration the physical limitation of transmitting data

over the internet and wireless connections). The elapsed time while a data change on one client propagates to another is visually imperceptible.

The Realtime Database also provides data persistence by storing data locally, thereby enabling the data to remain accessible even when a device is offline. When connectivity is re-established, the local data is automatically synchronized and merged with the remote data.[20]

Figure 4.5. *Firebase Process*

4.4 *Map Matching Algorithm*

4.4.1 Introduction

Map-matching algorithms integrate positioning data with spatial road network data (roadway cent relines) to identify the correct link on which a vehicle is travelling and to determine the location of a vehicle on a link. A map-matching

algorithm could be used as a key component to improve the performance of systems that support the navigation function of intelligent transport systems (ITS). The required horizontal positioning accuracy of such ITS applications is in the range of 1 m to 40 m (95%) with relatively stringent requirements placed on integrity (quality), continuity and system availability. A number of map-matching algorithms have been developed by researchers around the world using different techniques such as topological analysis of spatial road network data, probabilistic theory, Kalman filter, fuzzy logic, and belief theory. The performances of these algorithms have improved over the years due to the application of advanced techniques in the map matching processes and improvements in the quality of both positioning and spatial road network data. However, these algorithms are not always capable of supporting ITS applications with high required navigation performance, especially in difficult and complex environments such as dense urban areas. This suggests that research should be directed at identifying any constraints and limitations of existing map matching algorithms as a prerequisite for the formulation of algorithm improvements.

Map-matching algorithms use inputs generated from positioning technologies (such as GPS) and supplement this with data from a high resolution spatial road network map to provide an enhanced positioning output. The general purpose of a map-matching algorithm is to identify the correct road segment on which the vehicle is travelling and to determine the vehicle location on that segment . Map-matching not only enables the physical location of the vehicle to be identified but also improves the positioning accuracy if good spatial road network data are available. This means that the determination of a vehicle location on a particular

road identified by a map-matching algorithm depends to a large extent on the quality of the spatial road map used with the algorithm. A poor quality road map could lead to a large error in map-matched solutions. A map-matching algorithm can be developed generically for all applications or for a specific application.[21]

We use topological analysis in our project to define locations of driver (using GPS) .Topology refers to the relationship between entities (points, lines, and polygons). The relationship can be defined as adjacency (in the case of polygons), connectivity (in the case of lines), or containment (in the case of points in polygons). Therefore, a map-matching algorithm which makes use of the geometry of the links as well as the connectivity and contiguity of the links is known as a topological map-matching algorithm.We integrate our application with google-map and using distance-matrix service API and using merge sort algorithm to define best driver for our users who request a trip .

4.4.2 Google MAP API

API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. You can also use API calls to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area. These objects provide additional information for map locations, and allow user interaction with the map. The API allows you to add these graphics to a map:

- Icons anchored to specific positions on the map (Markers).
- Sets of line segments (Polylines).

- Enclosed segments (Polygons).
- Bitmap graphics anchored to specific positions on the map (Ground Overlays).
- Sets of images which are displayed on top of the base map tiles (Tile Overlays).

[22]

We use GeoJSON to identify lines and station locations in project .is an extension of the JSON data format and represents geographical data. Using this utility, you can store geographical features in GeoJSON format and render them as a layer on top of the map. To add and remove your GeoJSON data to and from the map, call addLayerToMap() and removeLayerFromMap() respectively. Similarly you can add and remove individual features by calling addFeature() and removeFeature() and passing in a GeoJsonFeature object. If you want to access the features, you can call getFeatures() to get an iterable of all GeoJsonFeature objects that have been added to the layer.

The map object

The Maps SDK for Android allows you to display a Google map in your Android application. These maps have the same appearance as the maps you see in the Google Maps for Mobile (GMM) app, and the API exposes many of the same features. Two notable differences between the GMM application and the maps displayed by the Maps SDK for Android are:Map tiles displayed by the API don't contain any personalized content, such as personalized smart icons. Not all icons

on the map are clickable. For example, transit station icons can't be clicked. However, markers that you add to the map are clickable, and the API has a listener callback interface for various marker interactions. In addition to mapping functionality, the API also supports a full range of interactions that are consistent with the Android UI model. For example, you can set up interactions with a map by defining listeners that respond to user gestures.

The key class when working with a map object is the GoogleMap class. GoogleMap models the map object within your application. Within your UI, a map will be represented by either a MapFragment or MapView object.

GoogleMap handles the following operations automatically:

- Connecting to the Google Maps service.
- Downloading map tiles.
- Displaying tiles on the device screen.
- Displaying various controls such as pan and zoom.
- Responding to pan and zoom gestures by moving the map and zooming in or out.

In addition to these automatic operations, you can control the behavior of maps with objects and methods of the API. For example, GoogleMap has callback methods that respond to keystrokes and touch gestures on the map. You can also set marker icons on your map and add overlays to it, using objects you provide to GoogleMap.

- **MapFragment** MapFragment, a subclass of the Android Fragment class, allows you to place a map in an Android fragment. MapFragment objects act as containers for the map, and provide access to the GoogleMap object. Unlike a View, a Fragment represents a behavior or a portion of user interface in an activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. Refer to the Android documentation on Fragments to learn more.

- **MapView**

  MapView, a subclass of the Android View class, allows you to place a map in an Android View. A View represents a rectangular region of the screen, and is a fundamental building block for Android applications and widgets. Much like a MapFragment, the MapView acts as a container for the map, exposing core map functionality through the GoogleMap object.

  When using the API in fully interactive mode, users of the MapView class must forward the following activity lifecycle methods to the corresponding methods in the MapView class: onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(), onSaveInstanceState(), and onLowMemory(). The ApiDemos repository on GitHub includes a sample that demonstrates how to forward the activity lifecycle methods. When using the API in lite mode, forwarding lifecycle events is optional.[22]

Indoor Maps

At high zoom levels, the map shows floor plans for indoor spaces such as airports, shopping malls, large retail stores, and transit stations. These floor plans, called

indoor maps, are displayed for the 'normal' and 'satellite' map types . They are automatically enabled when the user zooms in, and they fade away when the map is zoomed out. Here is a summary of the indoor maps functionality in the API:

- You can disable indoor maps by calling GoogleMap.setIndoorEnabled(false). By default, indoor maps are enabled. Indoor maps are displayed on one map at a time. By default this is the first map added to your app. If you'd like to display indoor maps on a different map, disable them on the first map then call setIndoorEnabled(true) on the second map.

- To disable the default level picker (floor picker), call GoogleMap.getUiSettings().setIndoorLevelPickerEnabled(false). For more details, see Interacting with the Map.

- An interface on GoogleMap, OnIndoorStateChangeListener, allows you to set a listener to be called when either a new building comes into focus, or a new level is activated in a building. For more details, see Interacting with the Map.

- GoogleMap.getFocusedBuilding() gives you the building that is currently in focus. You can then find the currently active level by calling IndoorBuilding.getActiveLevelIndex(). Refer to the reference documentation to see all the information available in the IndoorBuilding and IndoorLevel objects.

  Styling of the base map does not affect indoor maps.

4.4.3 Distance Matrix API

Google's Distance Matrix service computes travel distance and journey duration between multiple origins and destinations using a given mode of travel.

This service does not return detailed route information. Route information, including poly lines and textual directions, can be obtained by passing the desired single origin and destination to the Directions Service.

Distance Matrix Requests Accessing the Distance Matrix service is asynchronous, since the Google Maps API needs to make a call to an external server. For that reason, you need to pass a callback method to execute upon completion of the request, to process the results. You access the Distance Matrix service within your code via the google.maps.DistanceMatrixService constructor object. The DistanceMatrixService.getDistanceMatrix() method initiates a request to the Distance Matrix service, passing it a DistanceMatrixRequest object literal containing the origins, destinations, and travel mode, as well as a callback method to execute upon receipt of the response. The DistanceMatrixRequest contains the following fields:

- origins (required) — An array containing one or more address strings, google.maps.LatLng objects, or google.maps.Place objects from which to calculate distance and time.
- destinations (required) — An array containing one or more address strings, google.maps.LatLng objects, or google.maps.Place objects to which to calculate distance and time.

```
var origin1 = new google.maps.LatLng(55.930385, -3.118425);
var origin2 = 'Greenwich, England';
var destinationA = 'Stockholm, Sweden';
var destinationB = new google.maps.LatLng(50.087692, 14.421150);

var service = new google.maps.DistanceMatrixService();
service.getDistanceMatrix(
  {
    origins: [origin1, origin2],
    destinations: [destinationA, destinationB],
    travelMode: 'DRIVING',
    transitOptions: TransitOptions,
    drivingOptions: DrivingOptions,
    unitSystem: UnitSystem,
    avoidHighways: Boolean,
    avoidTolls: Boolean,
  }, callback);

function callback(response, status) {
  // See Parsing the Results for
  // the basics of a callback function.
}
```

Figure 4.6. *Example of distance matrix request*

- travelMode (optional) — The mode of transport to use when calculating directions. See the section on travel modes.
- transitOptions (optional) — Options that apply only to requests where travelMode is TRANSIT. Valid values are described in the section on transit options.
- drivingOptions (optional) specifies values that apply only to requests where travelMode is DRIVING. Valid values are described in the section on Driving Options.
- unitSystem (optional) — The unit system to use when displaying distance. Accepted values are: google.maps.UnitSystem.METRIC (default) google.maps.UnitSystem.IMPERIAL
- avoidHighways (optional) — If true, the routes between origins and destinations will be calculated to avoid highways where possible.

- avoidTolls (optional) — If true, the directions between points will be calculated using non-toll routes, wherever possible.[23]

4.4.4 Merge Sort

We can obtain the list addresses of available drivers and there are available seats on them cars by using distance matrix API . We use merge sort to obtain nearest driver from user's location and match between them .

Merge sort works as follows:

1. Divide the unsorted list into n sublists, each containing one element (a list of one element is considered sorted).
2. Repeatedly merge sublists to produce new sorted sublists until there is only one sublist remaining. This will be the sorted list.

Example of Merge Sort in detail[24] :

1. We take an unsorted array as the following

Figure 4.7. *unsorted array*

2. We know that merge sort first divides the whole array iterative into equal halves unless the atomic values are achieved. We see here that an array of 8 items is divided into two arrays of size 4.
3. This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.
4. We further divide these arrays and we achieve atomic value which can no more be divided.

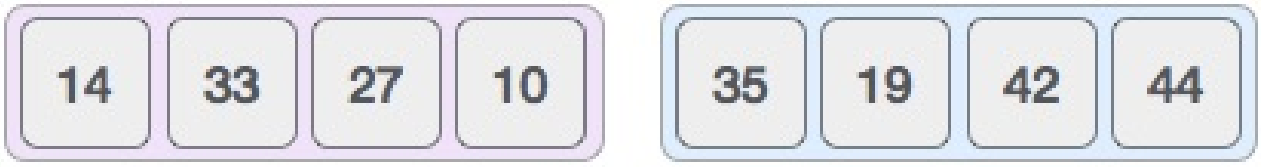


Figure 4.8. *Divide-1*

Figure 4.9. *Divide-2*

5. Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

   We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.
6. In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

7. After the final merging, the list should look like this

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being (n log n), it is one of the most respected algorithms.[24]

Figure 4.10. *Divide-3*

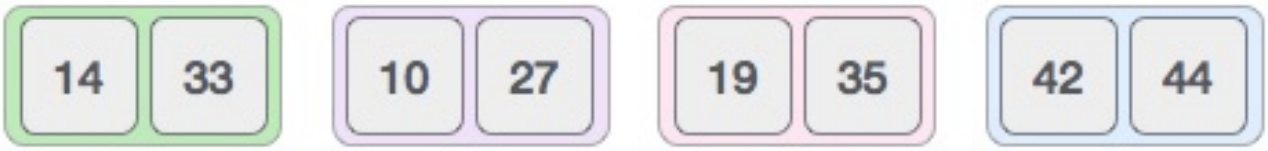


Figure 4.11. *Combine-1*

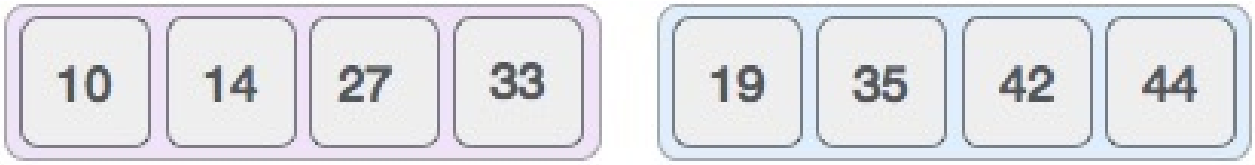


Figure 4.12. *Combine-2*

Figure 4.13. *Sorted array by Merge Sort*

CHAPTER

5

Admin Panel

5.1 *Features of Admin Panel*

Admin Panel monitors the application and provide some statistics to admins that give them an idea about the users' behaviors, revenue, number of errors and updates, trips count and the number of users joined in each month. Also the admin panel provides some features that help to control and view some of the application data. It provides the following features:

- **Statistics**:
  - Users' behavior: Shows the monthly performance of the number of trips made, the number of registered passengers and drivers.
  - Govern distribution: Points to the number of passengers in each govern. This helps to clarify which governs uses the application. This will be much useful when it comes to marketing and regional concentration of application advertisement. Governs with low numbers

of users registered will need more effort to make them get to know the application.

  - Monthly Revenue: Shows how much the application earned in each month throughout the year.
  - Info Cards: Views a quick non-detailed information about the application's total users (drivers and passengers), total revenue, number of errors in server's run time, number of updates made in application's back-end side.

- **Offers**: Provides the capability of pushing new offers to passengers with the help of Google Firebase. Simply we add the offer to the main application remote database and pushes notification to the user's using Google's Firebase services.

  These offers can be a discount on any trip the passenger make, a number of totally free trips or a free cash amount added to his wallet. These offers are most likely provided by partners and sponsors to help the application fan base grows. Also it shows the past offers published before and if they are still active or not.

- **Passengers**: Shows the List of the registered passengers and information about each registered account (name, E-mail, phone, govern, rate, wallet balance, number of trips made and family member) and also gives admins the ability to change some of these data. Driver Tracking: Admins can live

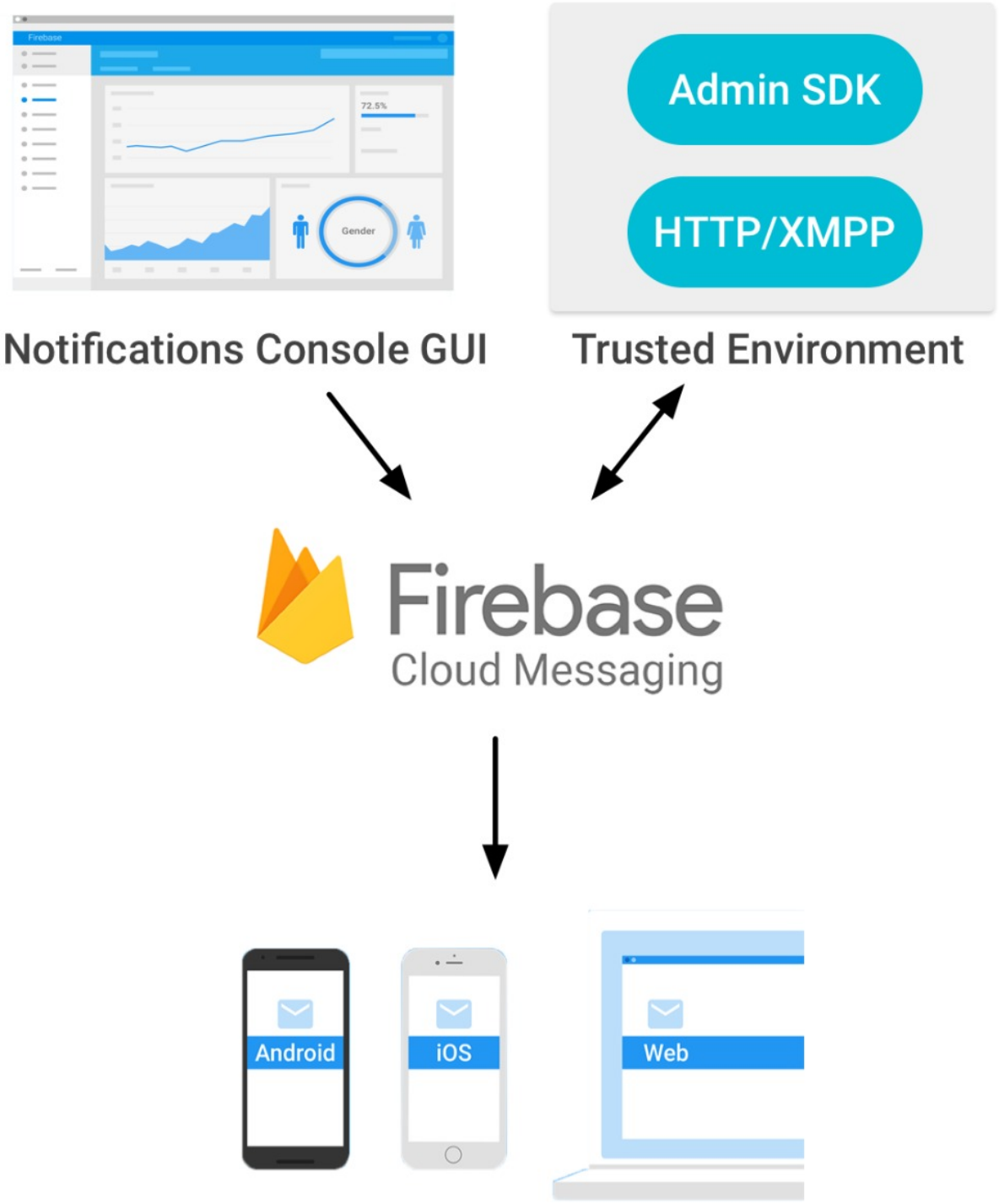


Figure 5.1. *Shows the requests flow between admin panel, firebase and client application.*

track every active driver using Google maps, to make sure that every driver sticks to his designated route.

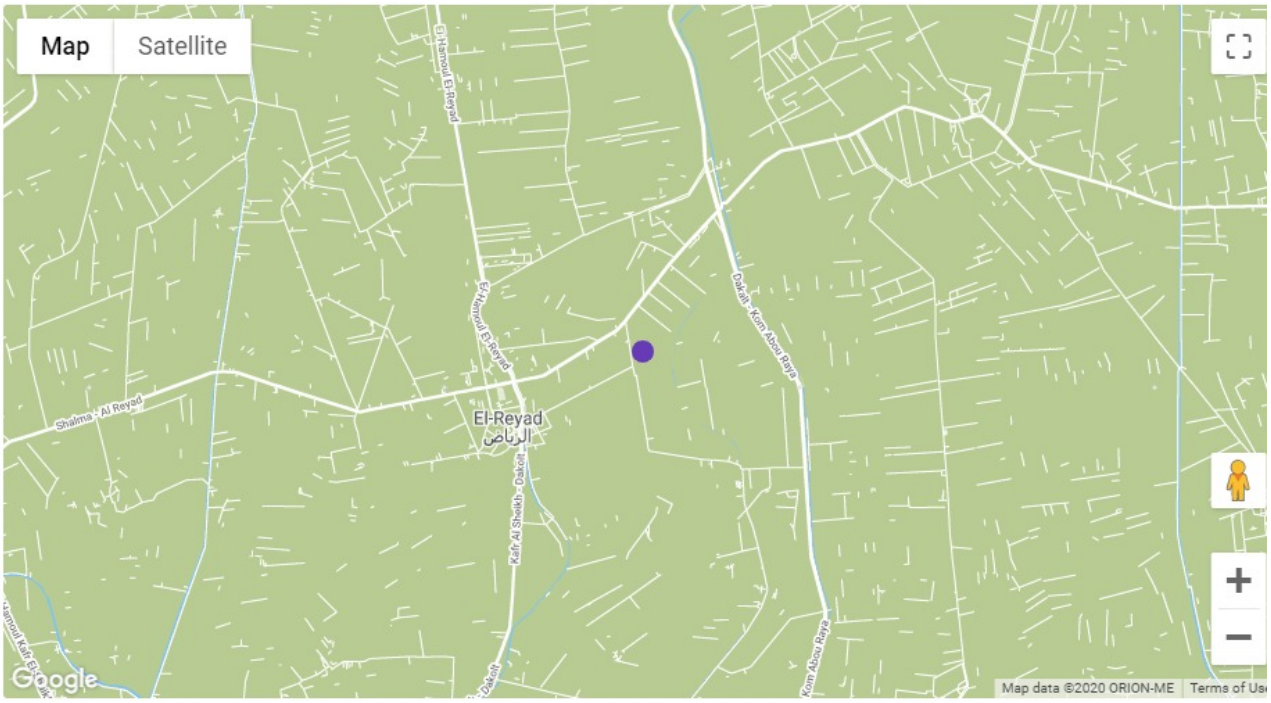


Figure 5.2. *Live location of the selected driver in admin panel.*

- **Complaints**: Admins can see all users' complaints and give proper responses to these complaints.
- **Lines**: Views all applications lines and gives the ability to delete or add lines. When adding lines the admin can choose lines' stations from the map directly and give names to these stations.

- **Driver Confirmation**: admin can see all drivers sent applications and accept or reject any of which, if they don't meet the requirements of registration.

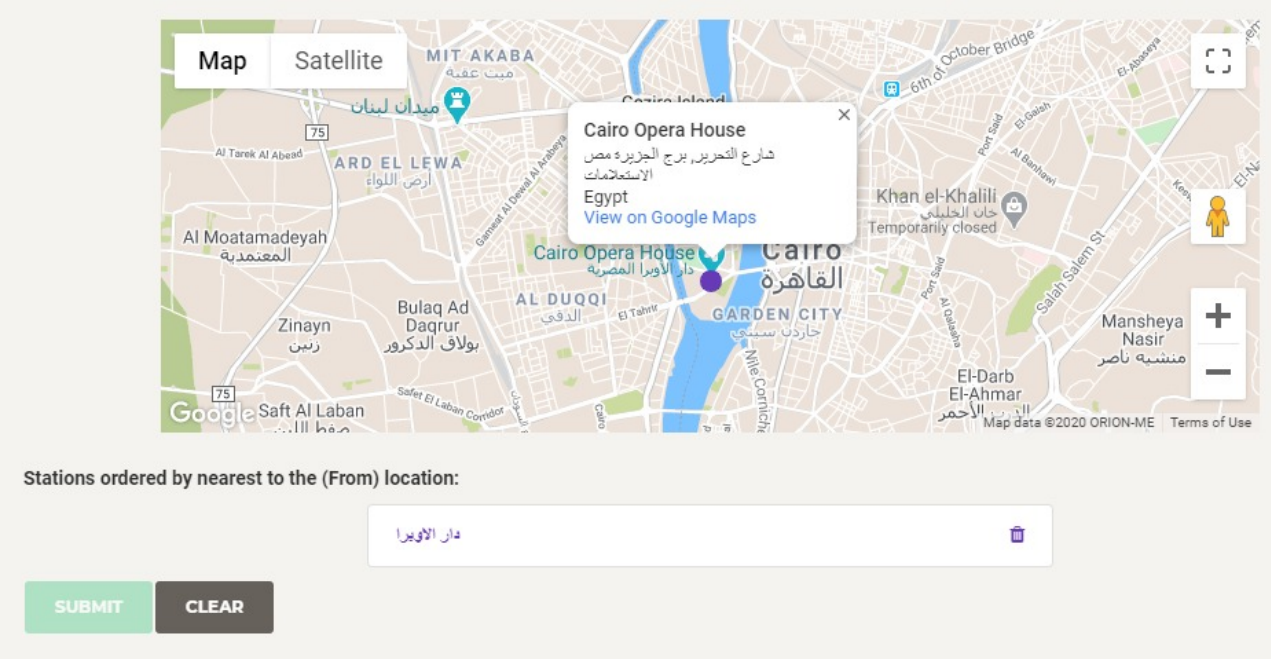


Figure 5.3. *Selecting station location from the map by clicking its location and naming it.*

CHAPTER

6

Conclusion and Future Work

6.1 *Conclusion*

Transportation is an important part of our daily lives. Thus, we did our best trying to solve one of its common problems that we almost face everyday. Some multi-national and national companies and some national start-ups have already founded actual solutions to this problem specifically, but they usually worked on the private transportation means only. Thus, we tried hard to work on the governmental transportation means to make this solution available to as many citizens as possible. Usually, the services of private companies and start-ups don't cover all governorates or all cities, while our service can serve all governorates and all cities. We put into consideration the passengers on highways and the passengers in remote areas.

6.2 *Future Work*

We still have a lot of stuff to work on and update to proceed our planed ideas in this project. In this section we introduce some of these ideas.

- Setting Multiple Admins Privileges
- Admin actions
- Enhance Punishment System
- Passenger Tour Broadcasting
- Family Members Tracking Mechanism
- Other Payment Options
  - Fawry.
  - Vodafone Cash.
- Web Application for Users
- Data Scaling
- Better Route Selection using PreviousDriver's Trip Information
- Automated Testing

CHAPTER

A

Appendix: Project Code

All project work is found in the following gitHub link:

https://github.com/raaedserag/Clax

References

[1] *ITS Introduction*. URL: https://www.acecc-world.org/ITS%20Introduction%20Guide_170306.pdf.

[2] *Smart Solutions in Today's Transport*. URL: https://link.springer.com/chapter/10.1007/978-3-319-66251-0_12.

[3] *Intelligent Transportation System Market Size, Share Trends By Type , By Application, By Region, And Segment Forecasts, 2020 - 2027*. URL: https://www.grandviewresearch.com/industry-analysis/intelligent-transportation-systems-industry.

[4] *Advanced Public Transport Systems, Simulation-Based Evaluation*. URL: https://link.springer.com/referenceworkentry/10.1007\%2F978-1-4614-5844-9_297.

[5] *Node.js*. URL: https://www.tutorialspoint.com/nodejs/index.htm.

[6] *Flutter.dev*. URL: https://flutter.dev/docs/resources/technical-overview.

[7] *ASP.NET MVC*. URL: https://www.tutorialsteacher.com/mvc/asp.net-mvc-tutorials.

[8] *MVC Framework*. URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_architecture.htm.

[9] *PayPal*. URL: https://www.paypal.com/be/smarthelp/article/what-is-paypal-and-how-does-it-work-faq1655.

[10] *Stripe*. URL: https://stripe.com.

[11] *Angular.io*. URL: https://angular.io.

[12] *RESTful API (REST API)*. URL: https://searchapparchitecture.techtarget.com/definition/RESTful-API.

[13] *REST matters (and you need more of it)*. URL: https://www.pluralsight.com/blog/tutorials/representational-state-transfer-tips.

[14] *authentication processes*. URL: http://www.infoguardsecurity.com/why-you-need-both-authorization-and-authentication/.

[15] *Hapi Joi*. URL: https://hapi.dev/module/joi/.

[16] *Mosh Node js couese*. URL: https://www.udemy.com/course/nodejs-master-class/?utm_source=adwords&utm_medium=udemyads&utm_campaign=LongTail_la.EN_cc.ROW&utm_content=deal4584&utm_term=_._ag_77879424134_._ad_437497333830_._kw__._de_c_._dm__._pl__._ti_dsa-1007766171312_._li_1005390_._pd__._&matchtype=b&gclid=CjwKCAjwrcH3BRApEiwAxjdPTSIWlrAjeaJtoFw1vxG6XGiw7dufyitUwUzoiOSCxkNxdpPHhoCCP4QAvD_BwE.

[17] *MongoDB vs SQL Databases*. URL: https://www.studytonight.com/mongodb/mongodb-vs-rdbms.

[18] *MongoDB Documentation*. URL: https://www.mongodb.com/.

[19] *Firebase Documentation*. URL: https://firebase.google.com/.

[20] *About Firebase*. URL: https://www.techotopia.com/index.php/Firebase_Realtime_Database.

[22] *google-map sdk android*. URL: https://developers.google.com/maps/documentation/android-sdk/map.

[23] *Distance Matrix API*. URL: developers.google.com/maps/documentation/javascript/distancematrix.

[24] *Example of Merge Sort algorithm*. URL: https://www.tutorialspoint.com/data_structures_algorithms/merge_sort_algorithm.htm.

[25] *authentication and authorization*. URL: https://www.okta.com/identity-101/authentication-vs-authorization/.

[26] *design steps*. URL: https://www.smashingmagazine.com/2016/11/what-everyone-should-know-about-the-process-behind-app-design/.

[27] *Merge Sort*. URL: https://en.wikipedia.org/wiki/Merge_sort.