CEN 308 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

VoziMe!

Prepared by:
Azra Kadric
Faris Mulaosmanović

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

24.06.2022.

TABLE OF CONTENTS

Table of Contents

1. Introduction

Ever found yourself in a hurry with no vehicle to drop you off to your desired destination? Most likely you have. Ever forgot the Taxi number? Ever got frustrated because of no answer? Ever googled for a different Taxi company? And all that in extreme hurry? Yes, it can be really frustrating, and that's why with VoziMe! we will make sure you will have all realtime available taxi drivers on the palm of your hand..

1.1. About the Project

VoziMe! Is an idea of an application whose primary audience would be young people who use smartphones in their everyday lives, as well as all those who can navigate through a simple application. We came up with a system similar to the popular Uber, but we still have to conform to Bosnia and Herzegovina's law. With modern technologies, it should be easy to convert a ready made web application into a very functional mobile one. Since we explored this subject in detail last year, it will be useful to try and develop it in code. We will be implementing just the part for users seeking a driver, not the driver user interface.

1.2. Project Functionalities

1. *Register as driver/passenger*
2. *Login*
3. *User profiles; Edit profile, update information, logout, delete profie*
4. *List of nearby available drivers based on the city and their current availability status*
5. *Request a ride, confirm driver arrived*
6. *Pay for ride (showcase)*
7. *Review your driver*
8. *View drivers profile*
9. *View your ride history*
10. *Invite friends*

2. Project Structure

2.1. Technologies

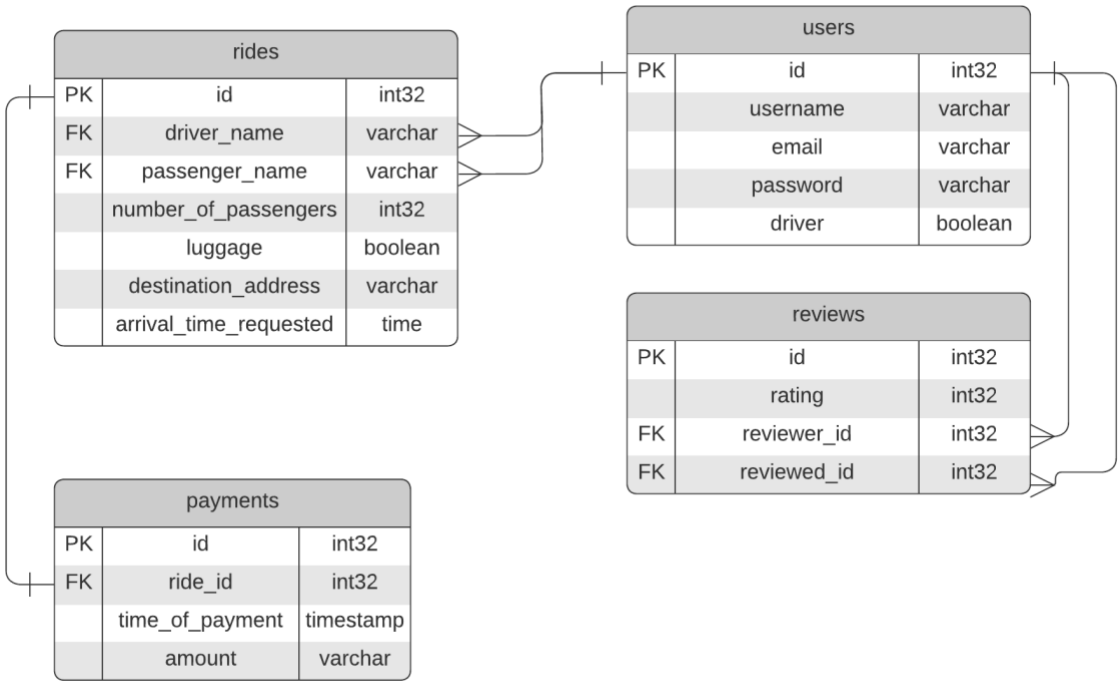
- ***- WAMPServer***
- ***- MySQL database with the help of SQLYog***
- ***- PHP for backend CRUD operations; PHP PDO usage***
- ***- FlightPHP for route handling, REST API***
- ***- HTML, CSS with Bootstrap template and JS***
- ***- JQuery for single page application framework***
- ***- AJAX requests***
- ***- Documentation using OpenAPI***
- ***JSON Web Tokens for authentication and authorization***

For PHP code, we were adhering to the PSR-12 standard of coding, which has been active since 2019. It requires that the code already follows PSR-1, so that was however the main focus in our project. For manually written CSS styles, we will use Airbnb CSS / Sass Styleguide.
For JavaScript and JQuery, as well as CSS and HTML, style guides found on jQuery's Style Guides were used.

2.2. Database Entities

- **Users**
- **Rides**
- **Reviews**
- **Payments**

These four entites are all interconnected accordingly, **users** entity provides certain data to **rides** and **reviews** data entities, **rides** entity consist all data necessary for certain drive and drop action to function mutually among driver-passenger relationship. Lastly **payments** entity is connected to **rides** entity as it enables monetization of drive and drop functionality of the application, more precisely the main feature of the app itself.



2.3. Architectural Pattern

As of architectural patterns are concerned we see that the compostion of the code of VoziMe! App is structured in a **Layered pattern.** This pattern can be used to structure programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer. Layered pattern is composed out of 4 sub-layers and so:

- ***Presentation layer (also known as UI layer)***

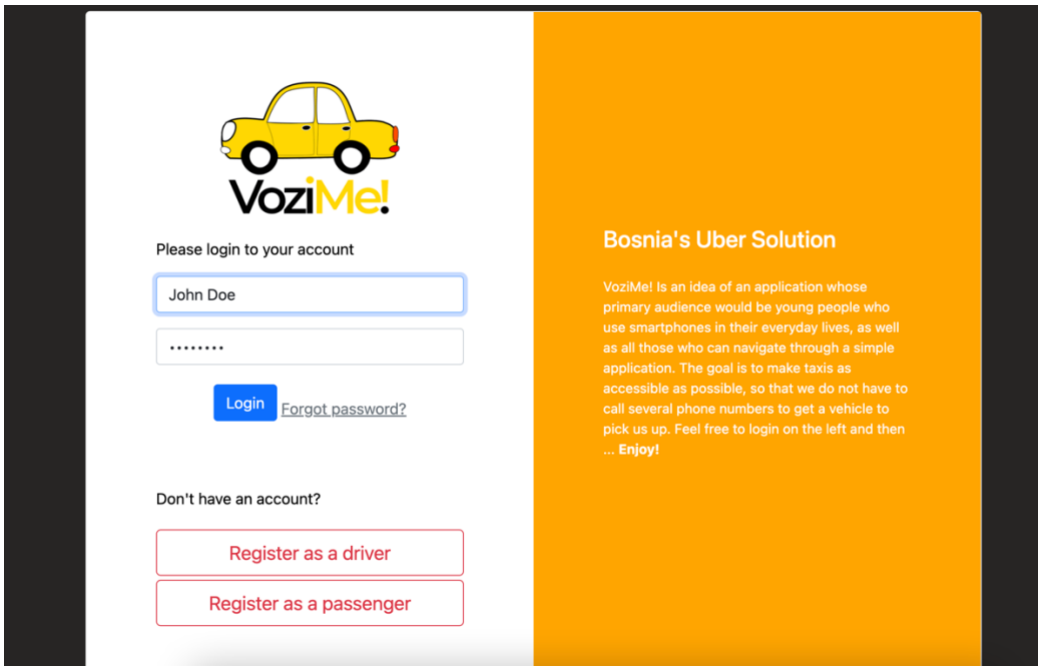


Figure 1. *We can see login screen for all registered users.*

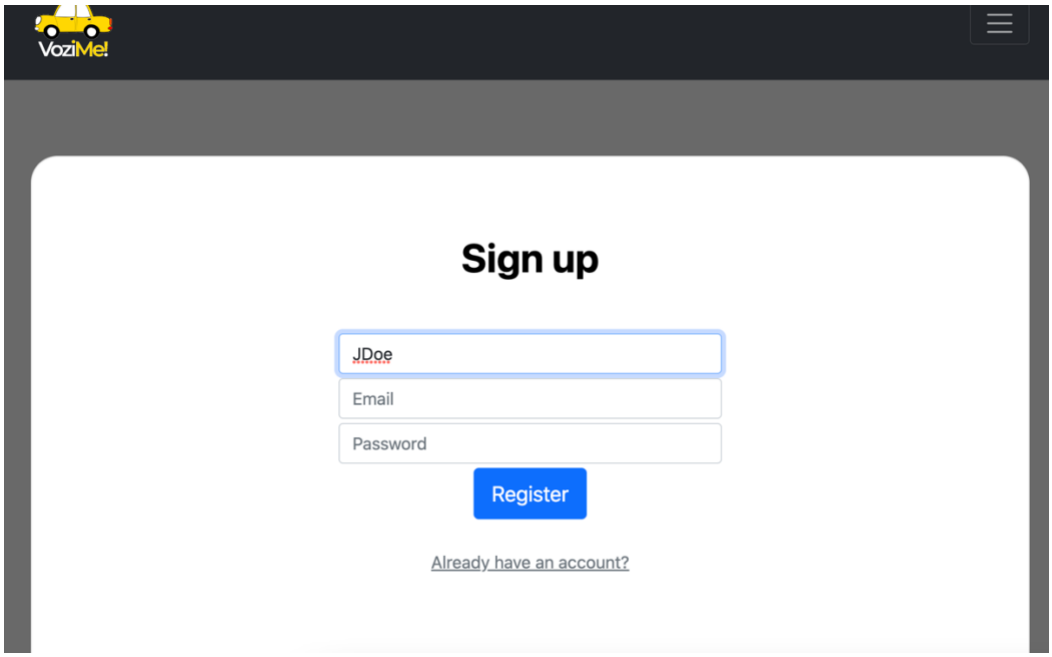


Figure 1.2 *We can see Registration form for all „Passenger“ Users*

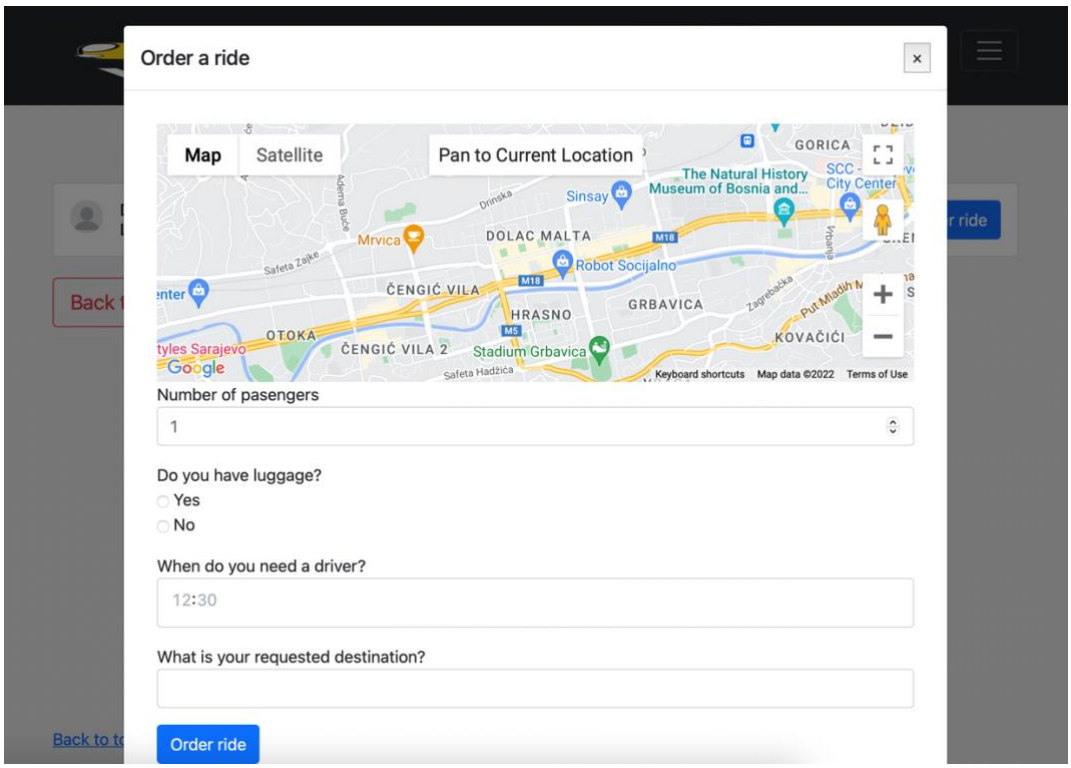


Figure 2. *Geolocation is presented for the passenger user and to pinpoint to current location, as well as fill up form for desirable fare.*

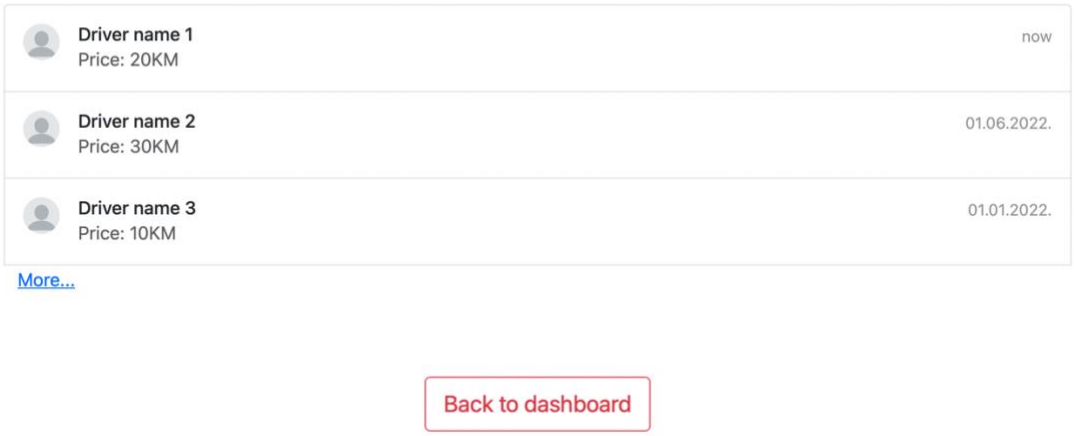


Figure 3. *In this pane the user „passenger“ is presented with all ride history along with its relevant data*

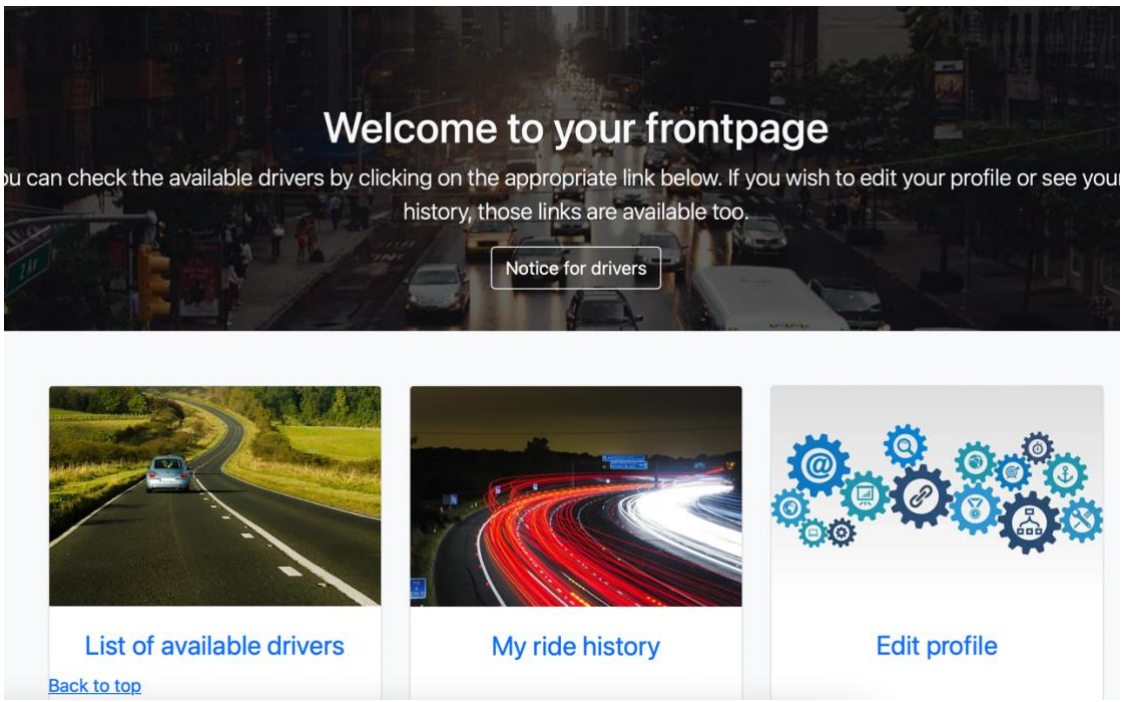


Figure 4. *Front page of the app, with quick access to all crucial apps functionalities*

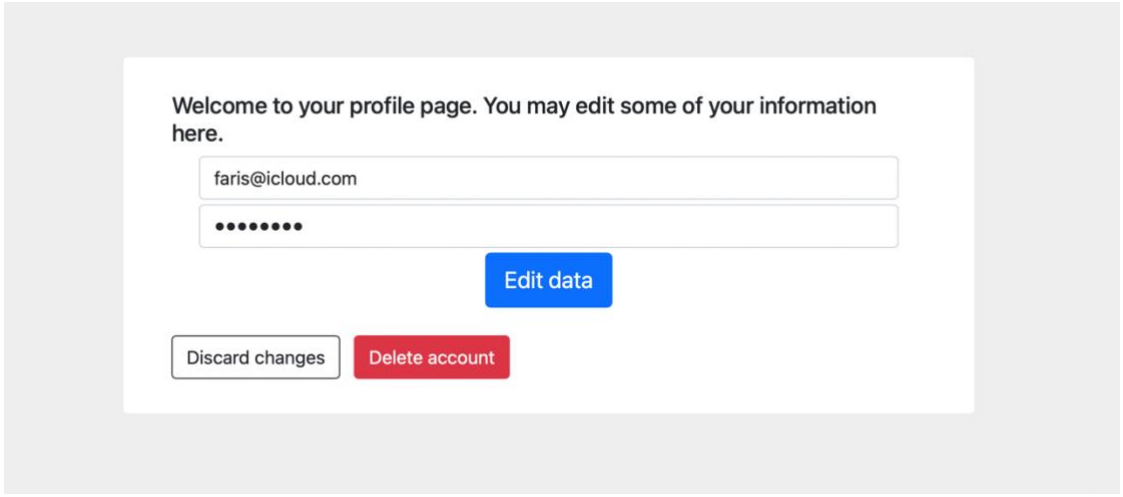


Figure 5. *Edit profile pane in which user can alter/update his personal information*

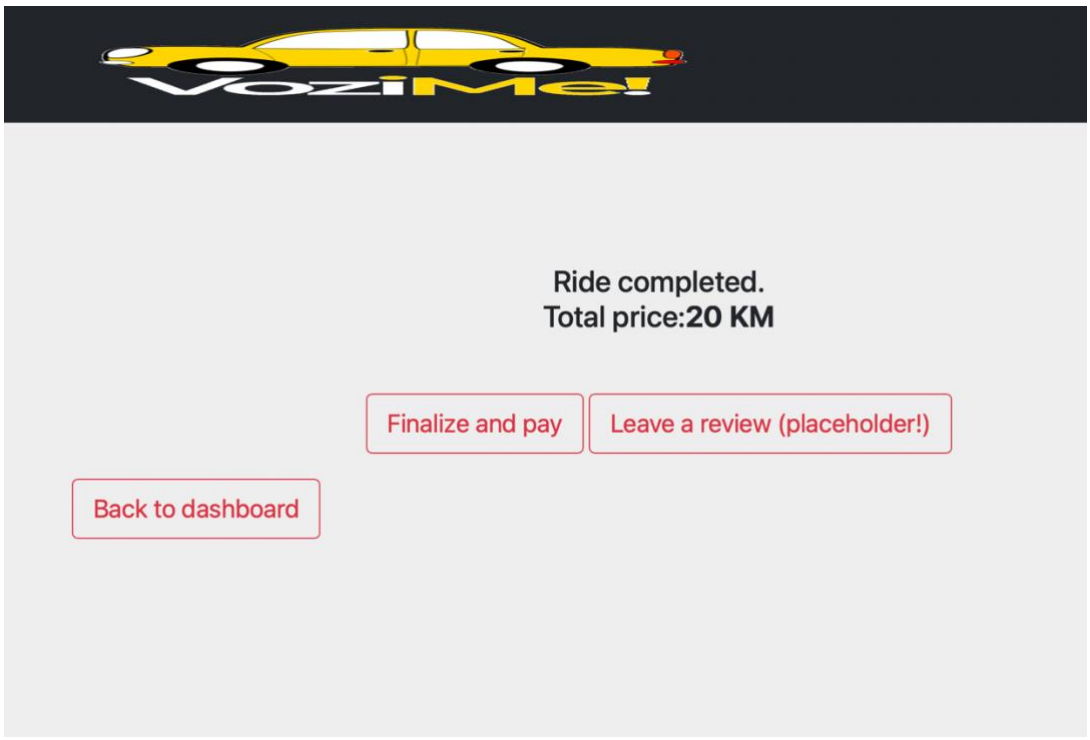


Figure 6*. Pane that is presented to a user upon fares completion, with option to finalize or leave a review and finalize*

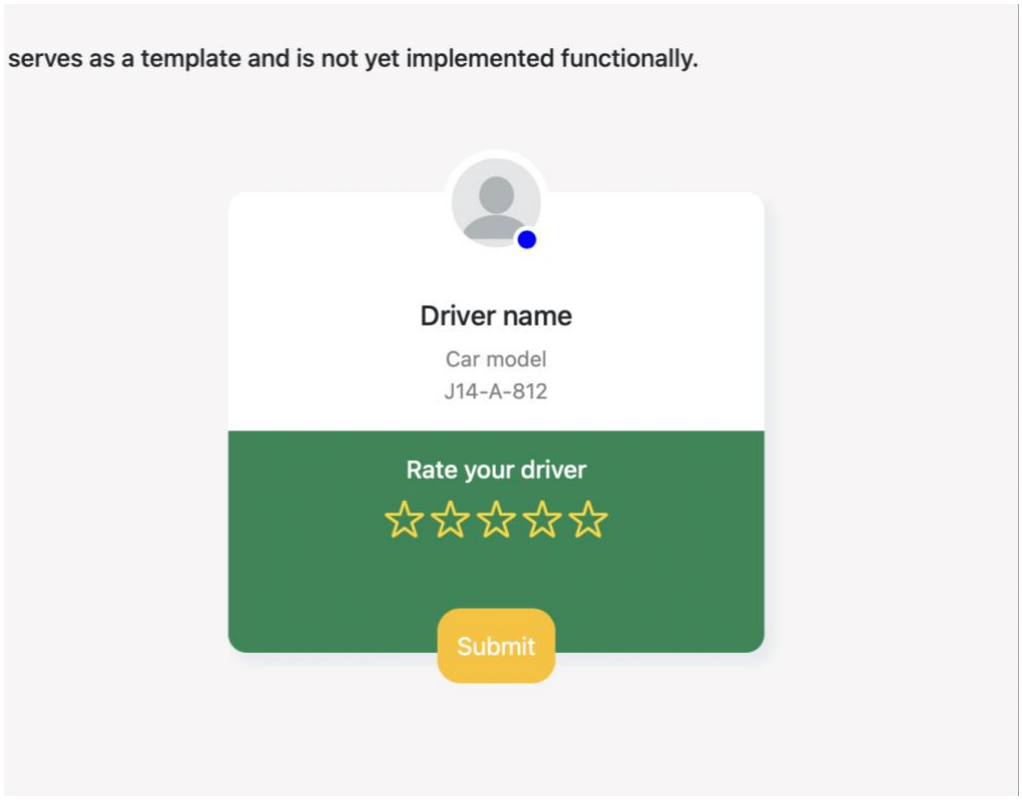


Figure 6.1 *Review pane that presents if passenger want to leave a feedback of his driver.*

- ***Application layer (also known as service layer)***

  VoziMe! Has couple of services implemented regarding the following:

a. User registration service

b. Ride service

c. Payment service

d. Driver registration service

e. Base service

All of named service are interconnected among each other, User/Driver registration is differentiated as it can be seen, reason for such differentiation is because these two types of users have different options available throughout the application; Ride service used functionality as a bridge for interconnection between two separate types of users; Payment service is self explanatory; Base service is used to store all data relevant for application proper functionality such as (Driver/Passenger dana, fare data, payment transactions,etc…)

2.4. Design Patterns

As of design patterns are concerned, in our application we can clearly identify the following design patterns:

- ***Structural Design Pattern***

Structural design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality. Moreover we can derive that **Private Class Data** has been implemented, this type of class dana restricts access to certain functionalities if user configuration is not supposed to have some of the apps functionalities, precisely in VoziMe!. Passenger account can not take other passengers for ride, can not be displayed as available driver and monetize hypothetical fares.

- ***Behavioral Design Pattern***

Behavioral design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects. We can identify the **Chain of responsibility** implementation, which is a way of passing a requests between a chain of object. We can see this for example in communication between passenger-driver, when a passenger initiates a request for a ride and selects a preferable driver, driver is notified of a such an action and is prone to react accordingly, which is accept the ride and continue with further actions upon completion or reject a ride.

3. Conclusion

Although we wanted to implement Selenium testing for this project, we decided against it, and to preferably focus on adding as much functionality as possible to the project. Since it is a small-scope web application for now, manual testing by the both of us and a helper person ensured that everything works properly. We tested the project manually after each and every change, from its earliest version to this final one. This applies to unit tests as well. Since VoziMe! is a small-scope student project, it was easy to track down all functions without having to write tests additionally. If we had kept further developing it, we agree that implementing both unit tests and Selenium tests would be a good option. It would make the process of testing code after changes a lot faster.

We believe that VoziMe! Has a lot of potential to serve its purpose as initially intended, having in mind that many industries are switching from analog to digital type of functionalities, main reason being ease of use and better time efficiency. We are witnesses of this as a fact in foreign countries all over Europe, and it is just a matter of time before application as this or as similar to this breaks the ground. User-friendly, ease of use, shorter course of interaction from call-wait-elaborate-order-wait-arrive-pay; to order-arrive-pay. Also very important aspect that VoziMe! App can benefit from is that Sarajevo is a anually visited from six to eigth hundred thousend tourists plus all other cities in Bosnia and Herzegovina. Most of these tourists are used to using these type of services, which can help it spread even faster and hopefully in near future become its own sort of transport standard for general population.