FMWAY-IŞIK

System Design

V1.0

12.04.2020

Umutcan Neşet Ölçer
Ersin Çebi
Emre Can Rua
Çağatay Demircan

Prepared for
SOFT3102 Software Project Development

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

<FMWay-Işık>

1. Introduction

Previously in RAD document, project's application domain and system analysis model were presented.

SDD document reports the transformation of the analysis model to system design model. SDD documents contains solution domain which is proposed and specified, design goals, subsystem decomposition, strategies and the definitions of subsystems and interfaces. Mainly, SDD portrays a virtual system that includes all of the specifications and requirements in RAD, and will create a service in boundaries between subsystems and interfaces.

As a software architecture system, we will use MVVM(Model-View-ViewModel). We believe that this system will fit our development the most. Some of the design goals that we will rely on are usability, customer experience, visual appeal, customer needs, performance, fit for purpose and so on. We will mention all of the design goals on "Design Goals" section.

1.1. Purpose of the System

FMWay Işık project is a transportation application exclusively for Işık University students. The main goal of the project is to provide a mobile application for the students who tries to reach to the campus or tries to travel to the city center from the campus.

The system aims to provide an efficient platform for users, which are admins, drivers and passengers. For passengers, the main functionality that aimed is to search convenient trips to the places that want to travel and join these trips. For drivers, the main functionality that aimed is to create the trips that they will do on upcoming time, and publish it on the application so the passengers can attend and split the road cost. For the admins, the main functionality that aimed is to editing the backup information ,user profiles, settings quickly, correctly and with ease. In short, for the admin, the aim is to control all the system to erase the mistakes such as fake trips, and control user profiles such as deciding on a user to be a driver.

1.2. Design Goals

FMWay-Işık android app is a mobile application which users can create or attend using their account balance. Passengers can use the system for searching the appropriate trips and join them. Drivers can use the system for creating a trip to others to join them. Admins can use the system for editing some backup information such editing trip information or approving passenger to be the driver. Therefore, this system should provide certain constraints, functional and non-functional requirements. For example, the passengers who use FMWay-Işık application should be able to see their active trips which they joined to, and the admins should be able to delete a trip.

Some of the important requirements and constraints are:

• Multiple Users

The system should support tasks that are performed by multiple users at a time, supplying each with the necessary information at the appropriate time.

<FMWay-Işık>

• Usability

Since the end-user will be using the system while performing work, it is essential for the system to be intuitive and easy to use. For example the user should be able to understand the process.

• Scalability

The system must be scalable in terms that it can support many users communicating or retrieving information at the same time.

• Reusability of Code

To minimize implementation time and improve efficiency, each part of the system has been designed as a component.

• Understandability

User should be able to understand the components of the system and can use them without any instructions.

• Reliability

System should be reliable which means it should take the stress of the systems and not make any failure.(A measure of success with which the observed behavior of a system confirms to the specification of its behavior)

• Location-Transparency

Server might itself be distributed, but provides a single "logical" service to the user

• High Performance

Client optimized for interactive display-intensive tasks; Server optimized for mobile CPU-intensive operations

• Flexibility

User interface of client supports a variety of end devices (From Android 6.0 to latest version).

• Customer Experience

Aim is to create a comforting feeling to user while using the application.

•Visual Appeal

Aesthetic design which can be told as modern,high quality, stylish, is aimed.

•Customer Needs

On the mark functions which are needed from the user, are targeted to implement.

•Performance

Well-working, errorless system is aimed.

•Fit for Purpose

Containing useful functions without any unnecessary additions to quality is aimed.

•Privacy

Goal is to avoid collecting and processing customer data. The saved customer information must be protected well.[2]

Functional Requirements

Functional requirements for the passenger:

- Sign up
- Login
- Edit profile

<FMWay-Işık>

- Logout
- Search Trip
- Apply for to be a driver
- Confirm trip
- Cancel the participation
- Finish the trip
- Payment
- Driver feedback

Functional requirements for the driver(Except the passenger features):

- Add trip
- Edit trip
- Delete trip
- Passenger feedback

Functional requirements for the admin:

- Edit user profiles
- Ban(Delete) user profiles
- Approve drive role for users
- Provide support for users
- Add admin profile

Constraints:
-The implementation language is Java.
-The project is mobile-based.(Android)

1.3. Definitions, Acronyms, and Abbreviations

SDD - System Design Document
RAD - Requirement Analysis Document
MVVM- Model View ViewModel
MVP- Model View Presenter
Admin: Actor of FMWay Işık System.
Database: The collection of large data set/stack.
Feedback: The answer of the system for the operation.
GitHub: A web-based hosting service, mostly used for computer code.
Passenger: Actor of FMWay-Işık System.
Driver: Actor of FMWay-Işık System.
Trips: The specific travels that drivers create and passengers attend to.
Parse Server: Open source Backend-as-a-Service(BaaS) framework initially developed by Facebook.
Java 8: A programming language.

<FMWay-Işık>

Template: A guide of making patterns.
Android Studio: An Integrated Development Environment (IDE) developed by IntelliJ.

1.4. References

For examining and understanding the transportation structure, we examined the applications Bla Bla Car, Uber, Carpolo, Carfriend and Tourbar. From these applications, especially Bla Bla Car and Carpolo are most similar and appropriate ones to our application. We took their appearances and the functions as examples to our project.

2. Current Software Architecture

There is no specific application which gives a service to the people either live in Şile or the students of Işık University. When we examine much more bigger systems such as Bla Bla Car, we can see that they use MVP(Model View Presenter) architecture.[3]

Model–view–presenter (MVP) is a derivation of the model–view–controller (MVC) architectural pattern which mostly used for building user interfaces. In MVP, the presenter assumes the functionality of the "middle-man". In MVP, all presentation logic is pushed to the presenter. MVP advocates separating business and persistence logic out of the Activity and Fragment.

Model

In an application with a good layered architecture, this model would only be the gateway to the domain layer or business logic. See it as the provider of the data we want to display in the view. Model's responsibilities include using APIs, caching data, managing databases and so on.

View

The View, usually implemented by an Activity, will contain a reference to the presenter. The only thing that the view will do is to call a method from the Presenter every time there is an interface action.

Presenter

The Presenter is responsible to act as the middle man between View and Model. It retrieves data from the Model and returns it formatted to the View. But unlike the typical MVC, it also decides what happens when you interact with the View. [4]

3. Proposed Software Architecture

The current systems that we examined and the mention on the previous headlines, are the successful ones. They have huge pool of users and provide a service on the big part of the world. However, even some of the applications are in use in Turkey, they don't have any service or big pool of users which covers or from Istanbul-Anatolian side region. For Işık University students, this system is a need, in order to provide this, we decided to create this very specialized system just for the people-students who study-work in Işık University and travel to Istanbul or travel from Istanbul to Şile.

As a software architectural system, we decided to use MVVM(Model-View-ViewModel).

<FMWay-Işık>

Model–view–viewmodel (MVVM) is a software architectural pattern that facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model). The view model of MVVM is a value converter, meaning the view model is responsible for exposing (converting) the data objects from the model in such a way that objects are easily managed and presented. In this respect, the view model is more model than view, and handles most if not all of the view's display logic. The view model may implement a mediator pattern, organizing access to the back-end logic around the set of use cases supported by the view.[5]

Model

Model refers either to a domain model, which represents real state content (an object-oriented approach), or to the data access layer, which represents content (a data-centric approach).

View

As in the model-view-controller (MVC) and model-view-presenter (MVP) patterns, the view is the structure, layout, and appearance of what a user sees on the screen. It displays a representation of the model and receives the user's interaction with the view (clicks, keyboard, gestures, etc.), and it forwards the handling of these to the view model via the data binding (properties, event callbacks, etc.) that is defined to link the view and view model.

View model

The view model is an abstraction of the view exposing public properties and commands. Instead of the controller of the MVC pattern, or the presenter of the MVP pattern, MVVM has a binder, which automates communication between the view and its bound properties in the view model. The view model has been described as a state of the data in the model.

The main difference between the view model and the Presenter in the MVP pattern, is that the presenter has a reference to a view whereas the view model does not. Instead, a view directly binds to properties on the view model to send and receive updates. To function efficiently, this requires a binding technology or generating boilerplate code to do the binding.[6]

3.1. Overview

In our design, the system has 17 subsystems.
These are: User Interface, Admin Interface, Driver Interface, Passenger Interface are the subsystems for the ‘Presentation Layer’. Login Subsystem, Join Subsystem, Trip Information Subsystem, Edit Trip Subsystem, Add Admin Subsystem, Payment Subsystem, To Be a Driver Subsystem, Chat Subsystem, Rating Subsystem, Manage Account Subsystem, Map Subsystem, Balance Subsystem are the subsystems for ‘Business Layer’. Data Access Subsystem is the subsystem for ‘Data Layer’.

User Interface: Provides services for users (driver, admin and passenger) for common interfaces, it has Login Page, Password Change Page, Profile Page and etc.

Driver Interface: Provides services for driver for driver interfaces, it contains trip information, get payment, see map, see trip destination and so on. Therefore, Driver Interface;

provides services to display forms which are trip details such as trip time and trip destination, functions.

Admin Interface: Provides functionalities that are only usable for admins. Under the interface there are subsystems for admin.

Passenger Interface: Provides services for passengers for passengers' interfaces, it contains trip information, payment, see map, see trip destination, chat system, balance and so on. Therefore, Passenger Interface; provides services to display forms which are trip details functions.

Login Subsystem: Provides services for users (driver, admin and passenger) to login.

Join Subsystem: Provides services for passenger to register for trips.

Trip Information Subsystem: Provides services for users to see trip details.

Edit Trip Subsystem: Provides services for users (admin and passenger) to update their trip information and to change trip details.

Add Admin Subsystem: Provides services for users (driver and passenger) to give administrator authority.

Payment Subsystem: Provides services for users (driver and passenger) to pay the money and to transfer travel fee

To Be a Driver Subsystem: Provides services for users (driver and passenger) to switch from passenger to driver status

---

Chat Subsystem: Provides services for users (driver and passenger) to send messages. They can explain their problems.

Rating Subsystem: Provides services for users (driver and passenger) to scores other users.

Manage Account Subsystem: Provides services for admin to change all user information, to update trip information, to check the balance of drivers and passengers etc.

Map Subsystem: Provides services for users (driver and passenger) to see the travel map and follow the itinerary.

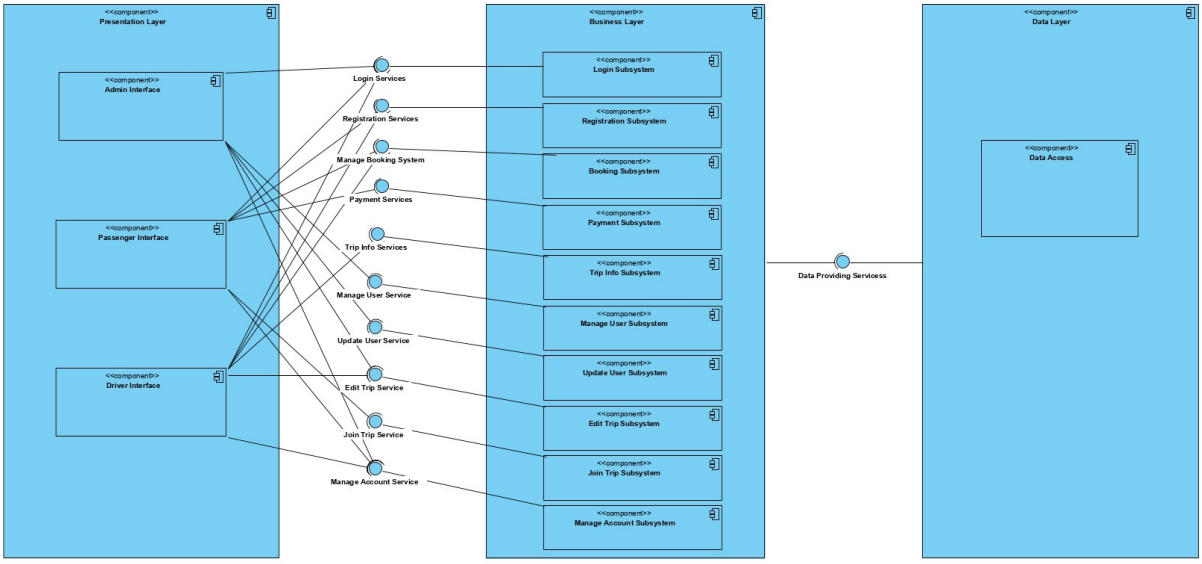
Balance Subsystem: Provides services for users (driver and passenger) to see the current balance.

Data Access Subsystem: Contains all our persistent objects.

3.2. System Decomposition

In FMWay Işık application, our subsystems are Admin Interface, Passenger Interface, Driver Interface, Login Subsystem, Login Subsystem, Registration Subsystem, Booking Subsystem, Payment Subsystem, Trip Info Subsystem, Manage User Subsystem, manage account Subsystem, Update User Subsystem, Edit Trip Subsystem, Join Trip Subsystem. Admin interface contains login service, Manage User Service, Update User Service, Manage Aaccount Service, Edit Trip Service. Passenger interface contains Login Services, registration service Manage Booking System, Payment Services, Join Trip Service. Driver interface contains login service, Login Services ,Registration Services, Manage Booking System, Trip Info Services, Edit Trip Service. Login subsystem provides service for all users to log in to the system. Join trip subsystem provides service to the passenger who have already registered to the system to trip which they searched. Registration subsystem provides service to guest to register to the system. Edit trip subsystem provides service to driver and admin to add, update or delete the trips on the system. Manage booking subsystem provides service to admin and driver to delete or edit the booking which the driver has made on the system. Trip info subsystem provides service to passenger to see the trip details which they search. Payment subsystem provides service to passenger to make payment on the system. Manage account subsystem provides service to users to edit or freeze their account on the system.

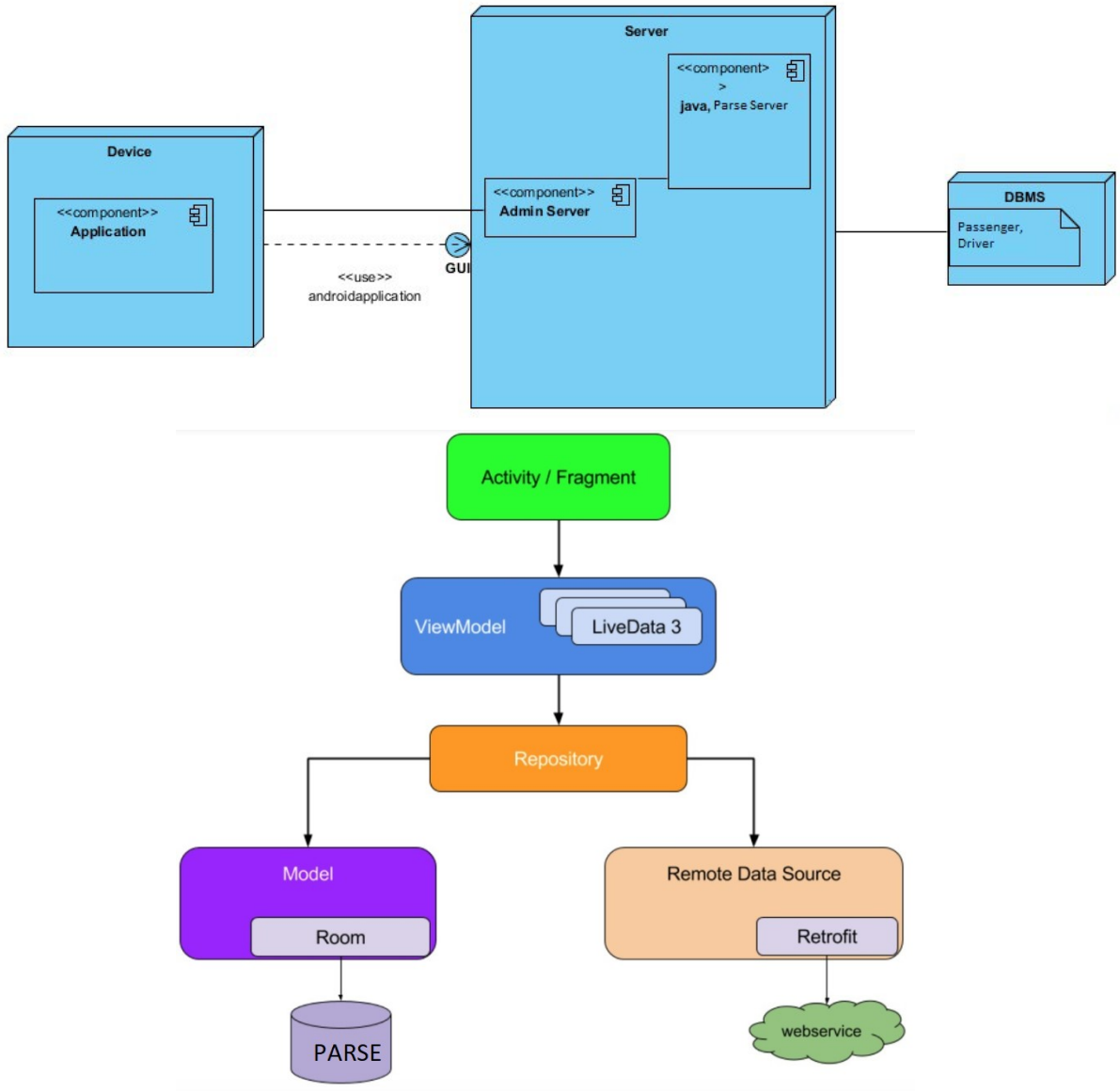
Update user info provides service to admin to update users account information or authorities on the system. Data Access Subsystem; contains all our persistent objects, this part could be called Model of MVVM.



3.3. Hardware Software Mapping

The FMWay Işık app has three layers for reusability and readability, which are Presentation, Business and data Access layers. Data Access layer will use Parse database management system for server connection, which is a version of mongodb, like firebase. Presentation Layer will be the personal smart phone. Since we are writing codes in Java and

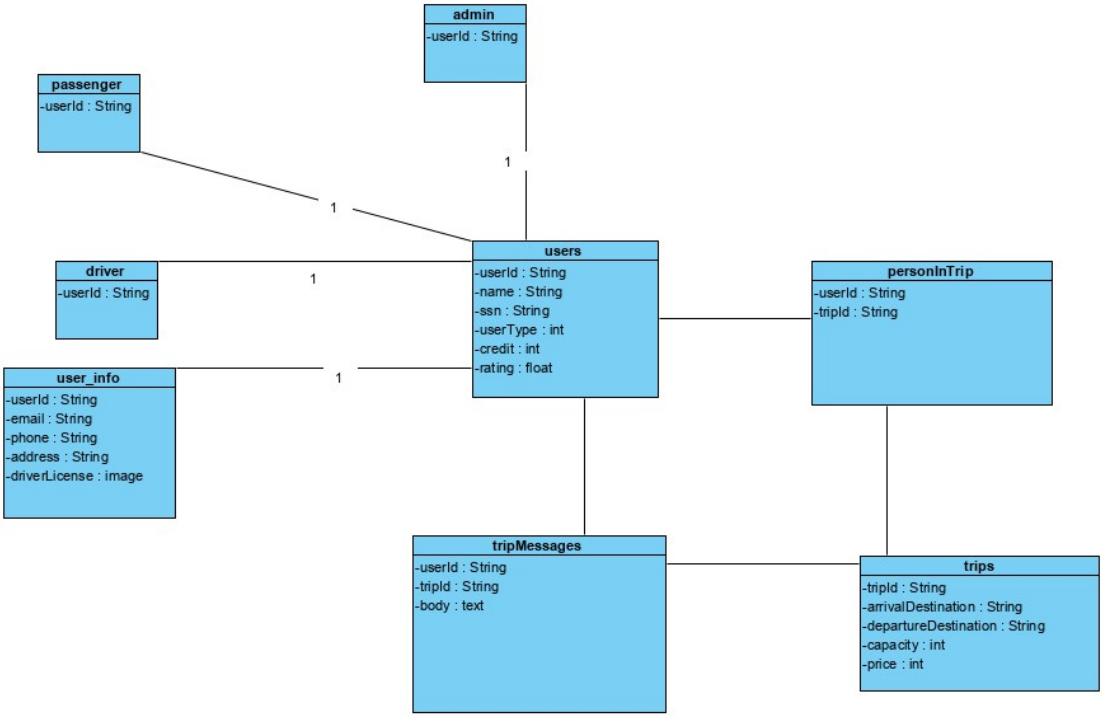
our machines are compatible with Android, we are using Android Studio on Presentation Layer.



3.4. Persistent Data Management

There are some constant data stored by the system, which is vital for the system to be of any use, so that the data can outlive a single execution of the system. For this reason, in the FMWay Işık app, we store this data in a database. The persistent data recorded are; Users of the system (Passengers, Drivers and Admin), and their information. We can store a list of Passengers enroll into a trip in their database of theirs and their personal information. Different tables from which information must be connected and extracted so they can be easily manipulated by operators such as project and join to give information in the form in which it is desired. Data independence is achieved more easily with normalization structure used in a relational database than in the more complicated tree or network structure. Looking at all these advantages that relational database provides us, we have decided that the relational database is the closest data management substructure for our system. Parse, which is a relational database management system (RDBMS) will be used to manage and keep the data.

<FMWay-Işık>



3.5. Access Control and Security

	Guest	Passenger	Driver	Admin	Trip
Guest	Register()				
Passenger		Login() ManageAccount() PayBalance() ApplyDriver()	RateDriver()	GetSupport()	ListTrip() JoinTrip() PayTrip()
Driver		RatePassenger()	Login() ManageAccount()	GetSupport()	ListTrip() JoinTrip() PayTrip() CreateTrip() EditTrip() DeleteTrip()
Admin		BanPassenger() ApplyDriver()	BanDriver()	GiveSupport()	EditTrip() DeleteTrip()

For this system, trip operations is important since we organize them clearly. The system should prove every user easy process for joining, creating, editing and deleting trips. The system checks and limits user operations for their roles. For example Passenger cannot create trip but join it. Drivers can create, join, edit and delete trip. So Drivers can reach both passenger and driver's operations. Admin can edit since security reasons. We added rating system to detect good and bad users to give this information to users. Every user can check rating on user's profiles.

<FMWay-Işık>

3.6. Global Software Control

The following section will describe the software control implementation. Centralized design will be good for our global software control.

Centralized Design; One control object or subsystem ("spider") controls everything.

All systems and operations can work concurrently.

3.7. Boundary Conditions

FMWay Işık is initialized by the system admin invoking the 'Initialize System' use case. Once the initialization of the system is completed, the place and time, date are added into system by the system admin. Furthermore, the system admin initialize the server by invoking the 'Start Server' use case. Then, when the system is brought from non-initialized state to steady-state, the system is opened for the system users (Passenger, Driver and Admin) to login. While the users is online and perform their tasks (join trip etc.) there might be some errors occurring during the transaction processes of the tasks.

The errors would be tried to update the personal information with empty information. This exception are get caught by the system. For example, to login with invalid credentials error, the use case named 'Invalid data' is invoked to handle the exception. When the 'Invalid data' use case is invoked, the system has already checked the user's username and password that if the username and password matches with the username and password in the database of the system. The system realizes that they do not match with each other, then in the process of the 'Invalid data' use case, the user is getting with a message that "Username or Password is not correct". The exception occurs if the user tries to update his/her personal information but necessary things is empty. Then FMWay Işık performs a checking to make sure that the necessary information fields are given and the information fields are proper to be updated. However, the system realizes that the necessary fields were not added, so the use case named 'Things To Fill In' is invoked to handle the exception. When the 'Things To Fill In' use case is invoked, a proper message is display to the logged user to fill the necessary things the update task.

The use cases mentioned above, namely 'Start Server', 'Initialize System' can be seen below:

Use case name:	Start Server
Participant actors:	Admin
Flow of events:	Admin executes the command that is used for the start server.
Entry Condition:	Admin logs into the server machine that the FMWay Işık services built on.

<FMWay-Işık>

Use case name: Initialize System
Participant actors: Admin
Flow of events: Admin executes the commands to add event, event date information and destination information
Entry Condition: Admin logged in into the database server
Exit Condition: Initialization of the system is completed and the event , event date information and destination information are added into the database server.

4. Subsystem Services

FmwayIşık system has 14 subsystems. Those subsytems are Passenger Interface, Admin Interface, Driver Interface, Registration Subsytem, Manage Account Subsytem, Trip Info Subsystem. Payment Subsytem. Pay Trip Subsytem. List Trip Subsytem, Join Trip Subsytem, Edit Trip Substem, Delete Trip Subsytem. Approve Driver Subsystem.

Presentation Layer:

- **Admin Interface :** Provides fuctionalities that are only usable for admins. Under the interface there are subsystems for admin.
- **Passenger Interface:** Provides subsytems that are only compatible with passenger user type.
- **Driver Interface:** Provides subsystem that are only compatible with driver user type.

Application Layer:

- **Registration Subsystem:** is a service provided to the user to register to the system. In this way, the user's information will be known by the system and the users will be able to benefit from different privileges.
- **Trip Info Subsystem:** Provides services for user to update trip information. In addition, the subsystem also provides the service about of trips information for drivers and passenger.
- **Manage Account Subsystem:** Provides a service for the user to edit his account. The user can edit or delete the account, but is in admin control.
- **Pay Trip Subsytem:** Provides a service for the user to pay to price of trip that decreases from balance.
- **List Trip Subsytem:** Provides a service for the user to list all trips that can joinable.
- **Join Trip Subsytem:** Provides a service for the user to join selected trip as passenger.

- **Edit Trip Subsytem:** Provides a service for the user to edit trip that created by user or admin.

- **Delete Trip Subsytem:** Provides a service for the user to delete trip that created by user or admin.

- **Approve Driver Subsystem:** Provides a service for the admin to approve passenger users that wants to be driver.

- **Payment Subsystem:** Provides a service for the user to add money to balance.

- **Update user info Subsystem:** Provides services for admin to change user profile from admin account.

5. References

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Lecture presentations of the course (the presentations were provided by the Instructor who is Emine Ekin)
2. 1. Bruegge B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed
.