Bilkent University

Department of Computer Engineering

Senior Design Project

Carpus

Low-Level Design Report

Deniz Çalkan, İbrahim Furkan Aygar, Mehmet Yiğit Harlak, Murat Sevinç, Veli Can Mert

Supervisor: Özgür Ulusoy
Jury Members: Shervin Arashloo, Hamdi Dibeklioglu

High-Level Design Report
February 28, 2022

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

1.1 Object design trade-offs

Rapid Development over Platform Independence: As a result of the time scope of the project, our project group has decided to develop the application only for Android devices. Therefore, the application will be available only for students using a valid version of the Android and will not be available for other students which means Carpus is a platform dependent application.

Usability over Functionality: Carpus is planned to be a simple and practical application as it will be used for a short period of time per day for each user. In order to design the application as simple as possible, it is planned to add functionalities which are necessary and not to have a complex design to maximize the user experience.

Security over Performance: Carpus needs to have access to student ID numbers and plate numbers of its users. Encryption of this information is essential to maintain the privacy of users. However, encryption methods and algorithms would decrease the performance of the application.

Data Reliability over Maintenance Time: Despite data reliability increases maintenance time, it is a must for Carpus. Therefore, our project group has chosen data reliability over maintenance time.

1.2 Interface documentation guidelines

In the following sections, conventions below will be used:

- Class names are given as the subsection titles and attributes and methods of the respective class are listed under these subsections.
- All class names are in a standard format which means meaningful and singular.
- Attributes belonging to the class are given under the “Attributes” part after class name and marked with ‘+’ and ‘-‘ signs. ‘+’ sign indicates the attribute is public and ‘-‘ sign indicates the attribute is private.
- Methods belonging to the class are given under the “Methods” part after class name and marked with ‘+’ and ‘-‘ signs. ‘+’ sign indicates the method is public and ‘-‘ sign indicates the method is private.

1.3 Engineering standards (e.g., UML and IEEE)

In this report IEEE referencing style is used for citations [1] and Unified Modeling Language (UML) guidelines [2] are strictly followed for the visualization of the design of our system.

1.4 Definitions, acronyms, and abbreviations

App: Abbreviated version for the word “application”. In the context of this report, an app is a smartphone software that is designed for users.

Carpus: The name of the application. It is a concatenated version of words “Car” and “Campus”.

Carpool: An arrangement between people to make a journey in a single vehicle.

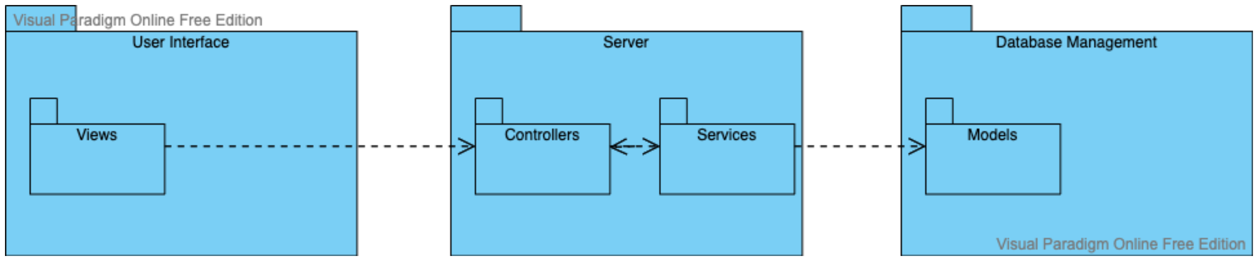
Route: The chosen way taken from a starting point to a destination.

Trip: A journey to a place, which is from a starting point to the campus or vice versa in our case.

Passenger: User of the Carpus who gets in any other user's car to travel.

Driver: User of the Carpus who drives his own car to carpool.

2. Packages


Visual Paradigm Online Free Edition
User Interface
Views
Server
Controllers
Services
Database Management
Models
Visual Paradigm Online Free Edition

User Interface:

Views: This package contains views classes which build the user interface part of the application. These views are specially created views like table component,selection data component. Also, buttons, text areas and other common user interface components are included. This package can have open-source user interface components.

Server:

Controllers: This controller package contains classes that manage HTTP requests which come from the user interface and connect these requests to application services in order to do back-end operations.

Services: This service package contains classes that are about application logic utilization.This classes manage and connect user-interface requests to application logic and application database. Many back-end actions can be defined in this package.

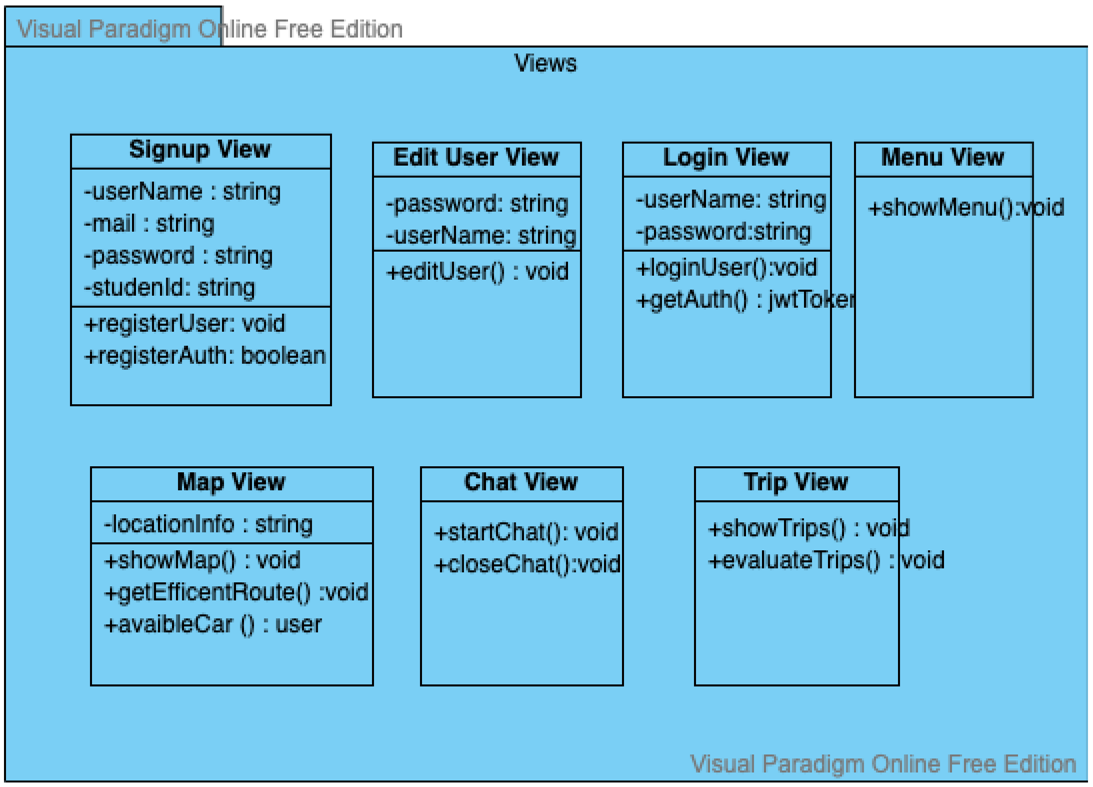
Database Management:

Models: Models package has classes such as User, Passenger, Driver. Basically there are data entities needed.In this package, to make easy connection of services and database, Hibernate and JPA artifacts can be used.

3. Class Interfaces

Some of the classes of Carpus are explained below in detail. Each class' attributes and methods other than getters and setters are listed.

3.1 Views



3.1.1 Signup View

Attributes	
-string userName	Unique identifier of users
-string password	Password of users
-string mail	University mail of users Required for registration
-string studentId	Identification number of the student

Methods	
+applyForTrip(Time arrivalTime, Date date)	Allows users to join an existing trip
+registerUser()	Send request to backend to register the user
+registerAuth()	Get auth token or control about registration of the user

3.1.2 Edit User View

Attributes	
-string userName	Users can send a new username to the backend to change their usernames.
-string password	Users can send a new password to the backend to change their passwords.

Methods	
+editUser()	Editing user information according to inputs.

3.1.3 Login View

Attributes	
-string userName	Unique identifier of users
-string password	Password of users

Methods	
+loginUser	To login the system method with user information.
+getAuth()	Get auth about the login process.

3.1.4 Menu View

Methods	
+showMenu()	Showing menu items

3.1.5 Map View

Attributes	
-string locationInfo	Information about the user location

Methods	
+showMap()	Get general map view
+getEfficentRoute()	Get efficient route information to user
+availableCar()	Show available cars to users.

3.1.6 Chat View

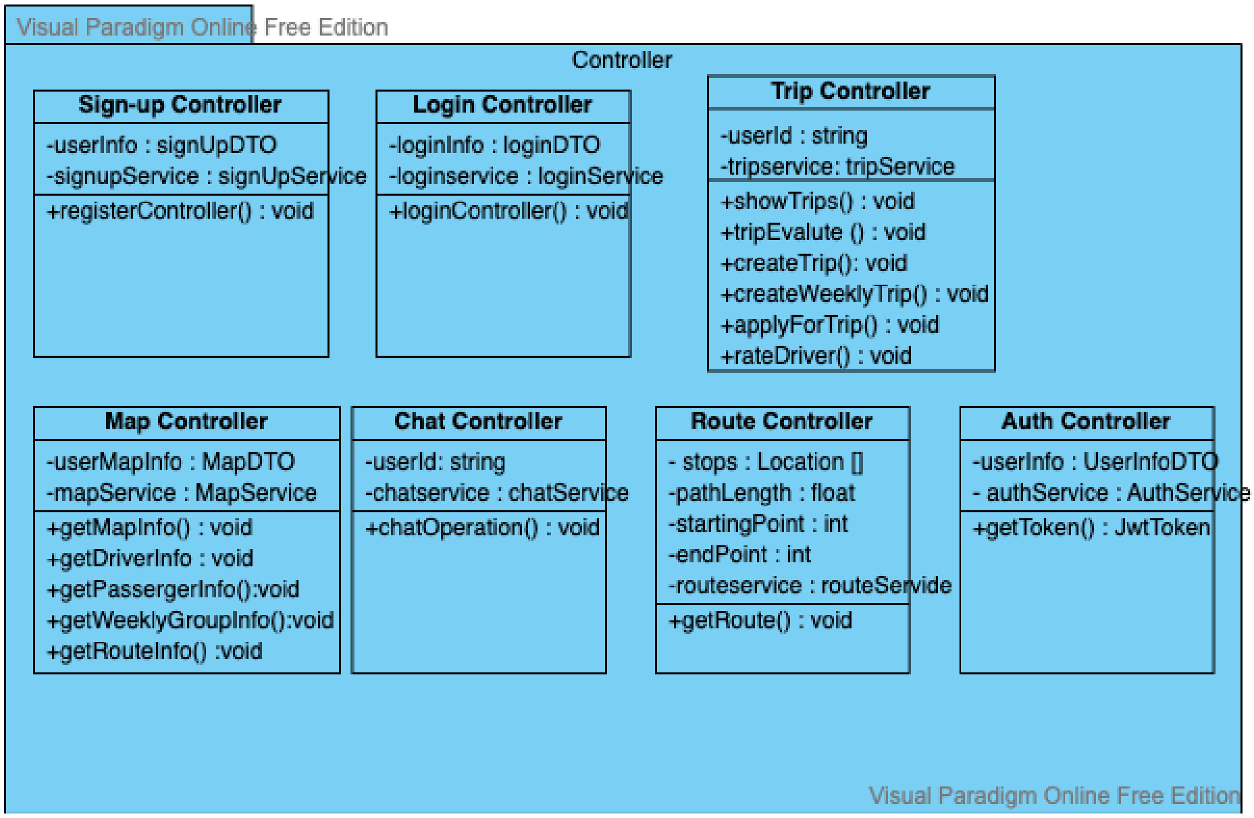
Methods	
+startChat()	Start chat view and operation.
+closeChat()	Close chat view and operation.

3.1.7 Trip View

Methods	
+showTrips()	Show all trips to users.
+evaluateTrips()	Evaluates users' trips.

3.2 Server

3.2.1 Controller



3.2.1.1 Sign-up Controller

Attributes	
-signUpDTO userInfo	Instance variable that shows if user signed up to the database
-signUpService signupService	Service object

Methods	
+registerController()	User information is sent by registerController

	service

3.2.1.2 Login Controller

Attributes	
-loginDTO loginInfo	Instance variable that shows if user signed up to the database
-loginService loginservice	Service object that calls login service

Methods	
+loginController()	It is used for handling http requests of login service

3.2.1.3 Trip Controller

Attributes	
-string userId	Instance that holds unique id of users
-tripService loginservice	Service object that calls trip service

Methods	
+showTrips()	This method send GET request to the server to obtain all trips took by user
+tripEvaluate()	This method send PUT request to the server by the evaluation which comes from end-user
+createTrip()	When a user creates a trip ,the controller will send a POST request to the server to hold information about that.
+createWeeklyTrip()	Creating a weekly car pooling group with users who use the same route.
+applyForTrip(Time arrivalTime, Date date)	POST request for allowing users to join an existing trip

+rateDriver()	Send POST request to rate driver

3.2.1.4 Map Controller

Attribute	
-userMapInfo: MapDTO	User map information object
-mapservice : mapService	Service object calls map service

Methods	
+getMapInfo()	GET request for getting general map information
+getDriverInfo()	GET request for getting driver map information
+getPassengerInfo()	GET request for getting passenger map information
+getWeeklyTripInfo()	GET request for getting weekly car pooling map information
+getRouteInfo()	GET request for getting current route map information

3.2.1.5 Chat Controller

Attribute	
-chatservice: chatService	Service objet calls chat service
-userId: string	User unique id in the system database.

Methods	
+chatOperation()	POST request creating and closing chat

	operations.

3.2.1.6 Route Controller

Attributes	
-Location[] stops	All stops on the current route
-float pathLength	Length of the current route
-Location startingPoint	Starting point (driver’s location)
-Location endPoint	Ending point (university)

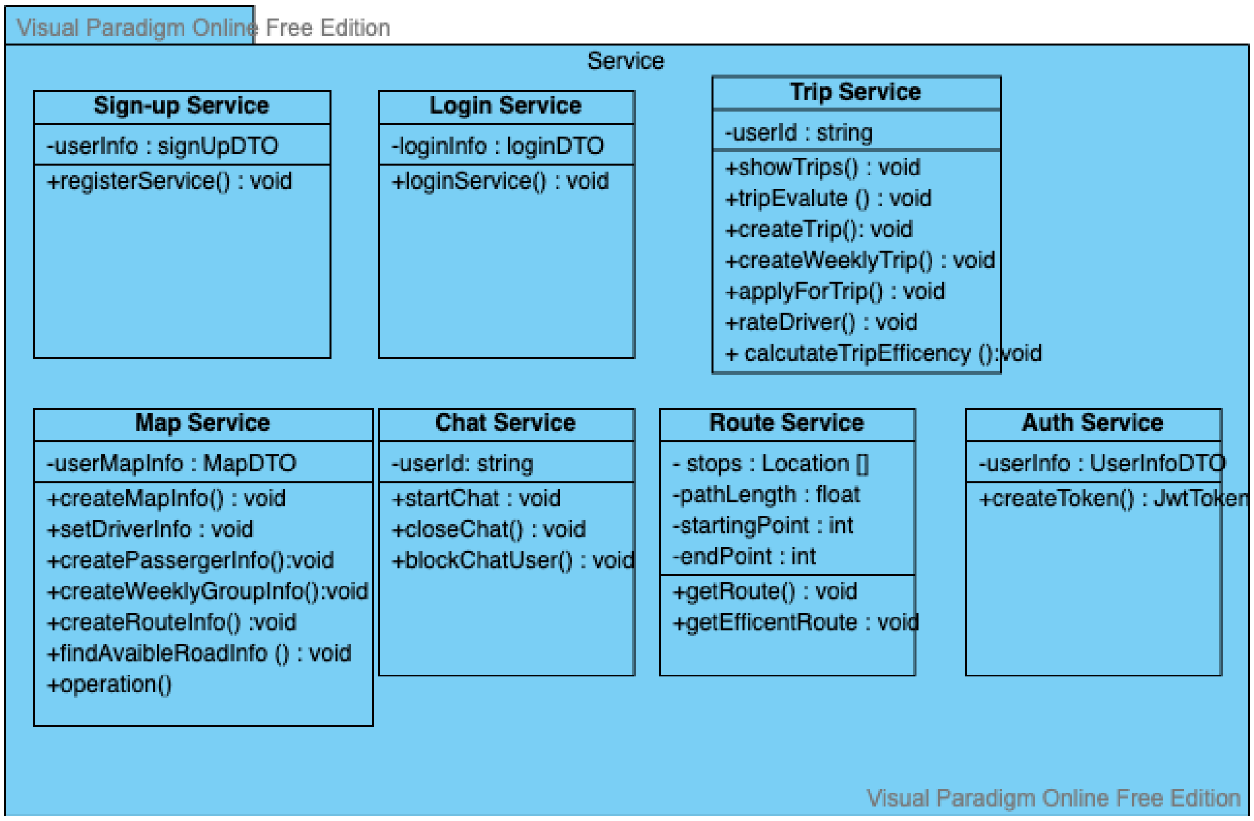
Methods	
+getRoute()	GET request for getting route information about the trip

3.2.1.7 Auth Controller

Attributes	
-userInfo : UserInfoDTO	Ending point (university)
-authservice :authService	Service object call auth service

Methods	
+getToken : JwtToken	GET request for creating jwt token for application session token

3.2.2 Service



3.2.2.1 Sign-up Service

Attribute	
-userInfo: signUpDTO	User information object for the service method

Methods	
+registerService()	Register the user to the system.

3.2.2.2 Login Service

Attribute	
-loginInfo: loginDTO	Login information object for the service method

Methods	
+loginService()	Login the user to the system.

3.2.2.3 Trip Service

Attribute	
-userId: string	User unique id in the system database.

Methods	
+showTrips()	Showing trips and getting all trips datas.
+tripEvaluate()	Evaluating the trip function.
+createTrip()	Creating a new trip function for users.
+createWeeklyTrip()	Creating a weekly car pooling group for those who use the same route and they have a car.
+applyForTrip()	Applying for a suitable trip for passengers.
+rateDriver()	Passengers can rate the driver of the trip.
+calculateTripEfficency()	Calculating trip efficiency to store trip data in order to find the best efficient trip.

3.2.2.4 Map Service

Attribute	
-userMapInfo: MapDTO	User map information data object.

Methods	
+createMapInfo()	Creating general map information for map view.
+setDriverInfo()	Setting driver information function.
+createPassengerInfo()	Creating passenger information who attend the

	trip.
+createWeekGroupInfo()	Creating weekly car pooling group map info for use map view.
+createRouteInfo()	Creating route info for map view.
+findAvailableRoadInfo()	Finding available and efficient roads information for the trip.
+operation()	General operating function about map when user wants to change anything about map.

3.2.2.5 Chat Service

Attribute	
-userId: string	User unique id in the system database.

Methods	
+startChat()	Starting chat conversation function for users
+closeChat()	Closing chat conversation function for users who attend the chat.
+blockChatUser()	Blocking unwanted users in chat functionality.

3.2.2.6 Route Service

Attributes	
-Location[] stops	All stops on the current route
-float pathLength	Length of the current route
-Location startingPoint	Starting point (driver’s location)
-Location endPoint	Ending point (university)

Methods	
+getRoute()	Getting current trip route information function

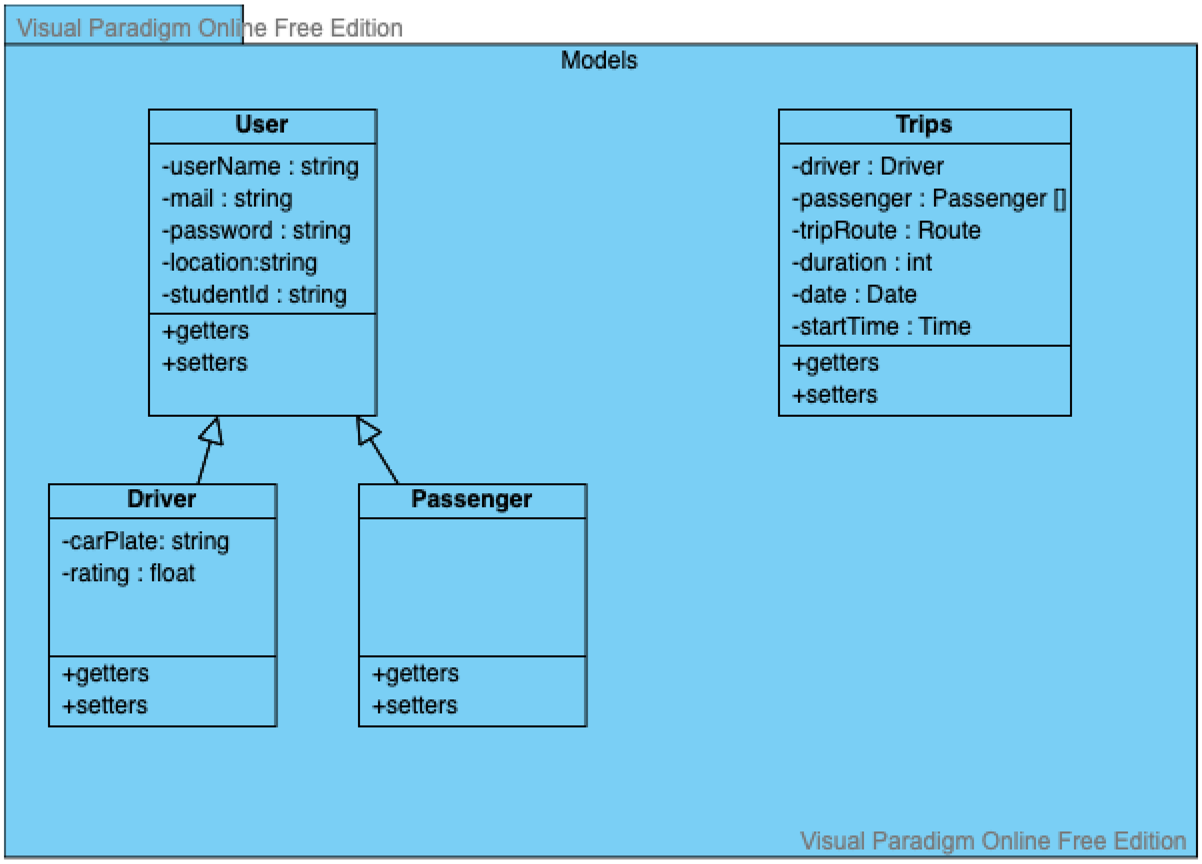
+getEfficentRoute()	Getting and calculating efficiency of the trip route function.

3.2.2.7 Auth Service

Attributes	
-userInfo : UserInfoDTO	Ending point (university)

Methods	
+createToken : JwtToken	Creating a JWT token for login and session of the application.

3.3 Models



3.3.1 User

Attributes	
-string userName	Unique identifier of users
-string password	Password of users
-string mail	University mail of users Required for registration
-Location location	Location of the user
-string studentId	Student identification number

3.3.2 Driver

Attributes	
-string userName	Unique identifier of users
-string password	Password of users
-string mail	University mail of users Required for registration
-Location location	Location of the user
-string studentInd	Student identification number
-string carPlate	Car plate of the driver Required for drivers
-string studentId	Student identification number

3.3.3 Passenger

Attributes	
-string userName	Unique identifier of users
-string password	Password of users
-string mail	University mail of users Required for registration
-Location location	Location of the user
-string studentId	Student identification number

3.3.4 Trips

Attributes	
-Driver driver	Driver of the current trip
-Passenger[] passengers	All passengers of the current trip
-Route tripRoute	Optimal route of the current trip

-Date date	Date of the trip
-int duration	Estimated duration of the trip
-Time startTime	Starting time of the trip

4. References

[1] IEEE Periodicals Transactions/Journals Department, “IEEE Reference Guide - IEEE Author Center,” IEEE Author Center. [Online]. Available: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf. [Accessed: 26-Feb-2022].

[2] “Welcome to UML web site!,” Welcome To UML Web Site! [Online]. Available: https://www.uml.org/. [Accessed: 26-Feb-2022].