Deliverable #2

SE 3A04: Software Design II – Large System Design

Tutorial Number: T01
Group Number: 10
Group Members:

- Vaya, Rishi
- Bahsoun, Mohamad-Hassan
- Giles, Isaac Ryan Godfrey
- Rajkarnikar, Umang

1 Introduction

This document includes but is not limited to; class diagrams and corresponding class responsibility collaboration cards, as well as architectural design (system architecture subsystems). These sections together are intended to provide a high level overview of the system architecture that the application should follow.

1.1 Purpose

The purpose of this document is to outline the high level architecture for the proposed "Taxi Mate" application. This document serves as a reference that should be used throughout the development process to understand the connections between classes in the system/subsystems, how the overall system should be separated out into smaller parts, and the general goal that is to be achieved by each subsystem.

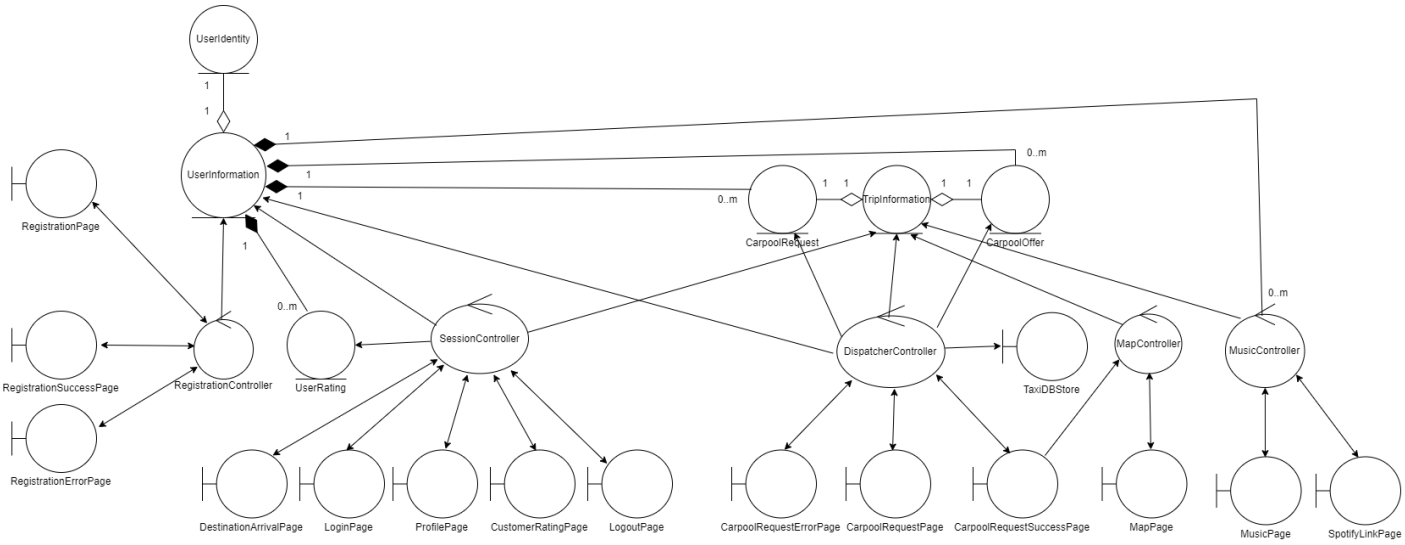
1.2 System Description

The system uses a central dispatcher controller to allow users to list carpooling offers and also to request to join available carpool rides. The dispatcher controller uses an entity class that contains trip information. This trip information class interfaces with a map controller that implements a GPS display based on the trip information. A session controller handles all requests for updating profiles, providing ratings, logging in, and logging out. The central registration controller will handle all requests to register for the application including situations where registration fails. The system also includes a music controller that accesses user information to store a users spotify account, and this controller is used to populate the music page with a users spotify playlists. Furthermore the music controller provides functionality to send spotify information to a vehicle that is associated with the current trip.

1.3 Overview

The breakdown for the rest of this technical document is as follows. In section 2, we break the system down into its corresponding boundary classes, controller classes, and entity classes. These classes are then brought together to form a complete analysis class diagram for the overall system. In section 3, we outline the system architecture that will be implemented for this project, explaining why this architecture is the best option and discussing why alternative design options were eliminated. Furthermore we create a structural architecture diagram to help visualize how our system will be encapsulated by the chosen system architecture. Section 3 will also break the system down into its relevant subsystems. In section 4, CRC cards are provided for all classes in the system.

2 Analysis Class Diagram



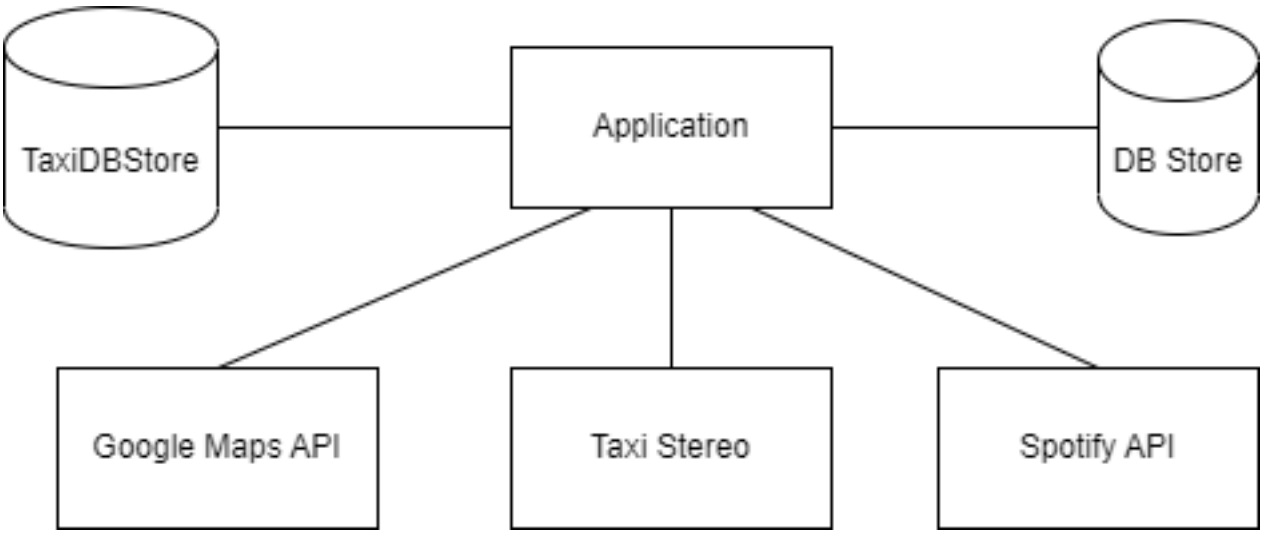
3 Architectural Design

3.1 System Architecture

The overall architecture of this system is a combination of repository and model-view-controller architectural patterns. From a data centered perspective, the system leverages the repository architecture. In a repository styled system, the data store is passive and the clients are active components that get and update the data store. For our application, the users are the clients and they will be primarily driving the flow logic of the system, where they will be sending requests to the system to perform some actions. In this case, these actions consist of matching carpool offers and requests.

Looking at the system from an interaction-oriented perspective, the system is modeled similar to that of a model-view-controller (MVC) styled architecture. In MVC, the model is responsible for handling any data store and business logic, where the view and controller modules can access the system's model. The controller responds to user events and controls the flow of the system. Then, the view can request data from the model and display information using a GUI. Similarly, our system will require a model component to perform actions when storing and managing user information and carpool matches. The view components would be used to read and display these pieces of information. To illustrate, the profile page would read and display the current user's data. Then, the controller would be used to handle input from the user by modifying the model. For example, this could be a request to create a new carpool offer by a user. Furthermore, MVC is a great choice where the business and end-user logic can be separated while also providing a simple way to implement user interfaces and control logic in the app.

One design alternative that had been considered was the presentation abstraction controller (PAC) architecture style. However, the PAC architecture style was more complicated to implement in comparison to the MVC architectural pattern. This is because the PAC style works where the control layer acts as a mediator between the presentation and abstraction layer. This means that the presentation layer cannot make requests to the abstraction layer and must go through the controller. Although this isolates the responsibilities of each component, it increases the complexity of the system where the controller handles all requests between the different agents. Unlike MVC, PAC does not allow the view layer to access the model, and so, it was eliminated from the list of architectural styles.



3.2 Subsystems

Application:
The Application is the main component of the overall system which deals with user registration and carpool matching. It interacts with the data store to create, update, and remove user and carpool information. The application also interacts with the Google Maps API to display the user's current geolocation, as well as the geolocation of their trip. It also leverages Spotify's API to connect the user's Spotify account to the app and display the user's Spotify playlist. Lastly, the application has access to a taxi's stereo system where the user can send a song request to the stereo.

DB Store:
The DB Store provides a means to collect, update, and remove data used by the application. This data can include user data, carpool requests, trip information, carpool offers, user ratings, and carpool data. It interacts with the Application where the Application can read, write, and remove entries from the DB Store.

Taxi DB Store:
The TaxiDBStore has all the information about the taxis which are available for carpools. This is an external subsystem which is accessed by the application when locating an available taxi.

Google Maps API:
The Google Maps API is used to determine the user's geolocation, as well as the geolocation of their trip. This API is interacted with by the Application.

Spotify API:
The Spotify API is used to connect a user's Spotify account to the Application. The application then utilizes the set of APIs to query the user's Spotify playlist.

Taxi Stereo:
The taxi stereo is used by the application to play songs requested by the user. It interacts with the Application where the Application sends a request to play a user selected song.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: RegistrationPage	
Responsibility:	**Collaborators:**
Knows Username	RegistrationController
Knows User Password	
Knows RegistrationController	
Sends User Information to RegistrationController	

Class Name: RegistrationErrorPage	
Responsibility:	**Collaborators:**
Knows RegistrationController	RegistrationController
Displays Registration Error to User	
Displays option to go back to Registration Page	
Handles Click Event of "Go Back To Registration Page" Button	

Class Name: RegistrationSuccessPage	
Responsibility:	**Collaborators:**
Knows RegistrationController	RegistrationController
Prompts User to Access System Features with "Let's Get Started" Button	
Handles Click Event of "Let's Get Started" Button	

Class Name: LoginPage	
Responsibility:	**Collaborators:**
Knows Knows user name	SessionController
Knows user password	
Sends user information to	
Handles click event of "login" button	
SessionController for authentication and create session for the user	

Class Name: LogoutPage	
Responsibility:	**Collaborators:**
Knows Knows user name	SessionController
Knows user password	
Sends user information to	
SessionController to end user's current session	

Class Name: ProfilePage	
Responsibility:	**Collaborators:**
Knows Knows user name	SessionController
Knows user password	
Knows user information	
Knows SessionController	
Allows user to send request to update profile information to SessionController	
Allows user to delete their account to SessionController	

Class Name: DestinationArrivalPage	
Responsibility:	**Collaborators:**
Knows trip information	SessionController
Knows user information	
Knows SessionController	
Displays trip details to the user	

Class Name: CustomerRatingPage	
Responsibility:	**Collaborators:**
Knows trip information	SessionController
Knows user information	
Knows SessionController	
Knows user information about other passengers on the trip	
Displays rating form to the user	
Sends user's ratings to the SessionController to save it	

Class Name: CarpoolRequestPage	
Responsibility:	**Collaborators:**
Knows user name	DispatcherController
Knows user information	
Knows DispatcherController	
Allows user to send carpool request to DispatcherController	

Class Name: CarpoolRequestErrorPage	
Responsibility:	**Collaborators:**
Knows trip information	DispatcherController
Knows user information	
Knows DispatcherController	
Displays error message to user	
Handles click event for "Return to Previous Page" button	

Class Name: CarpoolRequestSuccessPage	
Responsibility:	**Collaborators:**
Knows trip information	DispatcherController
Knows user information	
Knows DispatcherController	
Knows MapController	
Notifies user carpool request is successful	
Handles click event for "View Travel Details" button that triggers MapController	

Class Name: MapPage	
Responsibility:	**Collaborators:**
Knows trip information	MapController
Knows user information	
Knows MapController	
Sends request to MapController to retrieve user's geolocation	
Display user's geolocation on GoogleMaps	

Class Name: MusicPage	
Responsibility:	**Collaborators:**
Knows stereo details of taxi	MusicController
Knows MusicController	
Knows UserInformation	
Sends request to MusicController to play user selected music	

Class Name: SpotifyLinkPage	
Responsibility:	**Collaborators:**
Knows user information	MusicController
Knows MusicController	
Sends request to MusicController to connect user's spotify account	

Class Name: TaxiDBStore	
Responsibility:	**Collaborators:**
External database containing all information relating to each of the taxis in the finite set of taxis in the system.	

Class Name: RegistrationController	
Responsibility:	**Collaborators:**
Knows RegistrationPage	RegistrationPage
Knows RegistrationSucessPage	RegistrationSucessPage
Knows RegistrationErrorPage	RegistrationErrorPage
Knows UserInformation	UserInformation
Knows validation logic for registering new user	
Fetches user information and returns it	

Class Name: SessionController	
Responsibility:	**Collaborators:**
Knows ProfilePage	ProfilePage
Knows CustomerRatingPage	CustomerRatingPage
Knows LoginPage	LoginPage
Knows LogoutPage	LogoutPage
Knows DestinationArrivalPage	DestinationArrivalPage
Knows UserInformation	UserInformation
Knows CustomerRating	CustomerRating
Fetches user information and returns it	
Updates user information	
Removes user information	
Updates passenger rating data	
Fetches passenger rating data	

Class Name: DispatcherController	
Responsibility:	**Collaborators:**
Knows CarpoolRequestPage	CarpoolRequestPage
Knows CarpoolRequestSuccessPage	CarpoolRequestSuccessPage
Knows CarpoolRequestErrorPage	CarpoolRequestErrorPage
Knows UserInformation	UserInformation
Knows TripInformation	TripInformation
Knows CarpoolRequest	CarpoolRequest
Knows CarpoolOffer	CarpoolOffer
Fetches user information and returns it	
Fetches trip information	
Updates trip information	
Creates carpool request	
Fetches carpool request(s)	
Creates carpool offer	
Fetches carpool offer(s)	

Class Name: MapController	
Responsibility:	**Collaborators:**
Knows MapPage	MapPage
Knows TripInformation	TripInformation
Fetches trip information	
Calculates user's geolocation using Google	
Maps API and returns it	

Class Name: MusicController	
Responsibility:	**Collaborators:**
Knows MusicPage	MusicPage
Knows TripInformation	
Knows Sends user's song selection to taxi's stereo system	

Class Name: UserIdentity	
Responsibility:	**Collaborators:**
Knows user identification number	UserInformation
Knows user password	
Knows UserInformation	

Class Name: UserInformation	
Responsibility:	**Collaborators:**
Knows user name	UserIdentity
Knows user password	
Knows user data	
Knows UserIdentity	
Create user information instance	
Update user's information	
Delete user's information	

Class Name: CustomerRating	
Responsibility:	**Collaborators:**
Knows user name	UserInformation
Knows user data	
Knows UserInformation	
Create customer rating(s)	
Update customer rating(s)	
Delete customer rating(s)	

Class Name: CarpoolRequest	
Responsibility:	**Collaborators:**
Knows user name	UserInformation
Knows user data	TripInformation
Knows trip details	
Knows UserInformation	
Knows TripInformation	
Create carpool request	
Update status of carpool request	

Class Name: CarpoolOffer	
Responsibility:	**Collaborators:**
Knows user name	UserInformation
Knows user data	TripInformation
Knows trip details	
Knows UserInformation	
Knows TripInformation	
Create carpool request	
Update status of carpool offer	

Class Name: TripInformation	
Responsibility:	**Collaborators:**
Knows trip details	CarpoolRequest
Knows CarpoolRequest	CarpoolOffer
Knows CarpoolOffer	
Create trip information instance	
Update trip information entry	

A Division of Labour

Mohamad-Hassan Bahsoun Sections: Fixed Deliverable 1, 4(CRC 1-6), 2(came up with Boundary classes), 3.1(repository section)

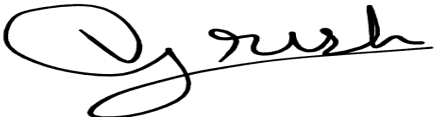
Rishi Vaya Sections: Fixed Deliverable 1, 4CRC(13-18),2(came up with Entity classes),3.2(the first 3 subsystems)

Isaac Giles Sections: 1, 4(CRC 7-12), 2(came up with Controller classes), 3.2(The last 3 subsystems)

Umang Rajkarnikar Sections: 4(CRC 19-26), 2(Came up with Entity Classes, Created the diagram), 3.1 (MVC, and PAC Section)

Mohamad-Hassan Bahsoun

Rishi Vaya



Isaac Giles

Umang Rajkarnikar