Deliverable #3: TaxiMate

SE 3A04: Software Design II – Large System Design

1 Introduction

This document includes but is not limited to; State Charts for each controller class, Sequence Diagrams for each use case of the system, and a detailed Class Diagram of the system. These sections together are intended to provide how different components of the system are interacting with each other.

1.1 Purpose

The purpose of this document is to outline how different components communicate and operate 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 how different parts of the system operate. The engineers, developers and/or whoever is going to be building this software system is who this document is intended for.

1.2 System Description

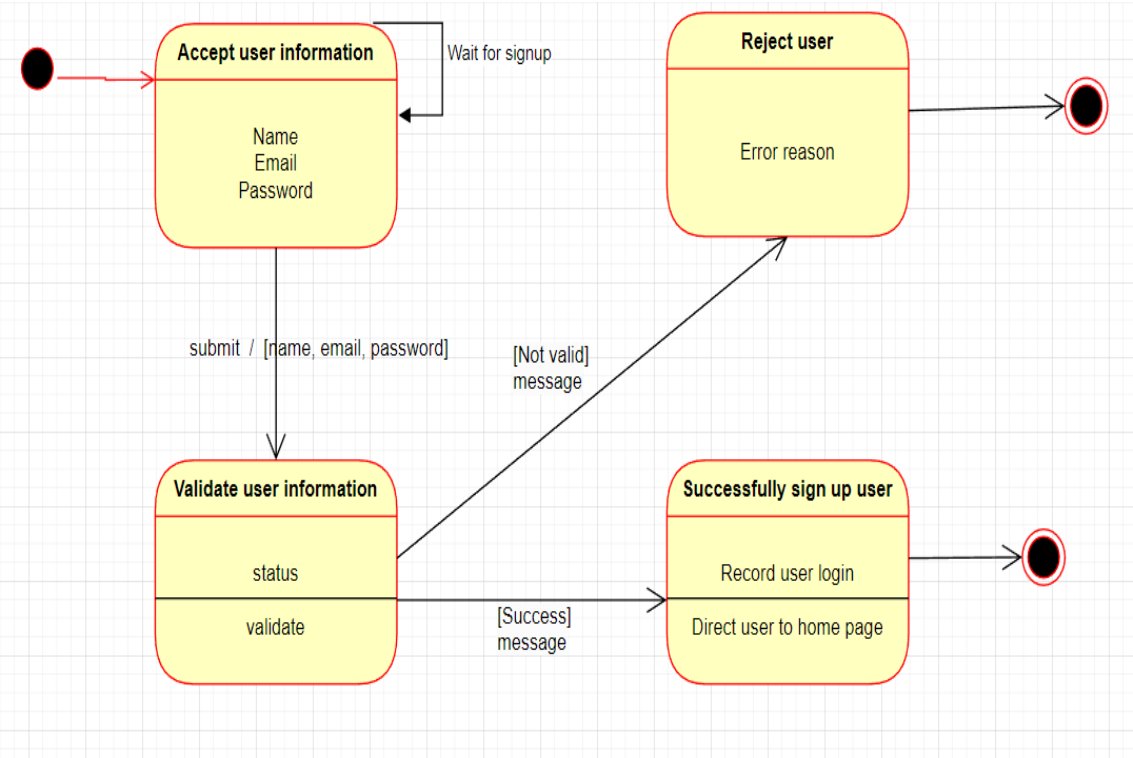
The system uses a central dispatcher controller to allow users to list carpooling offers and also to request to join available carpool rides. The dispatcher controller uses an entity class that contains trip information. This trip information class interfaces with a map controller that implements a GPS 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

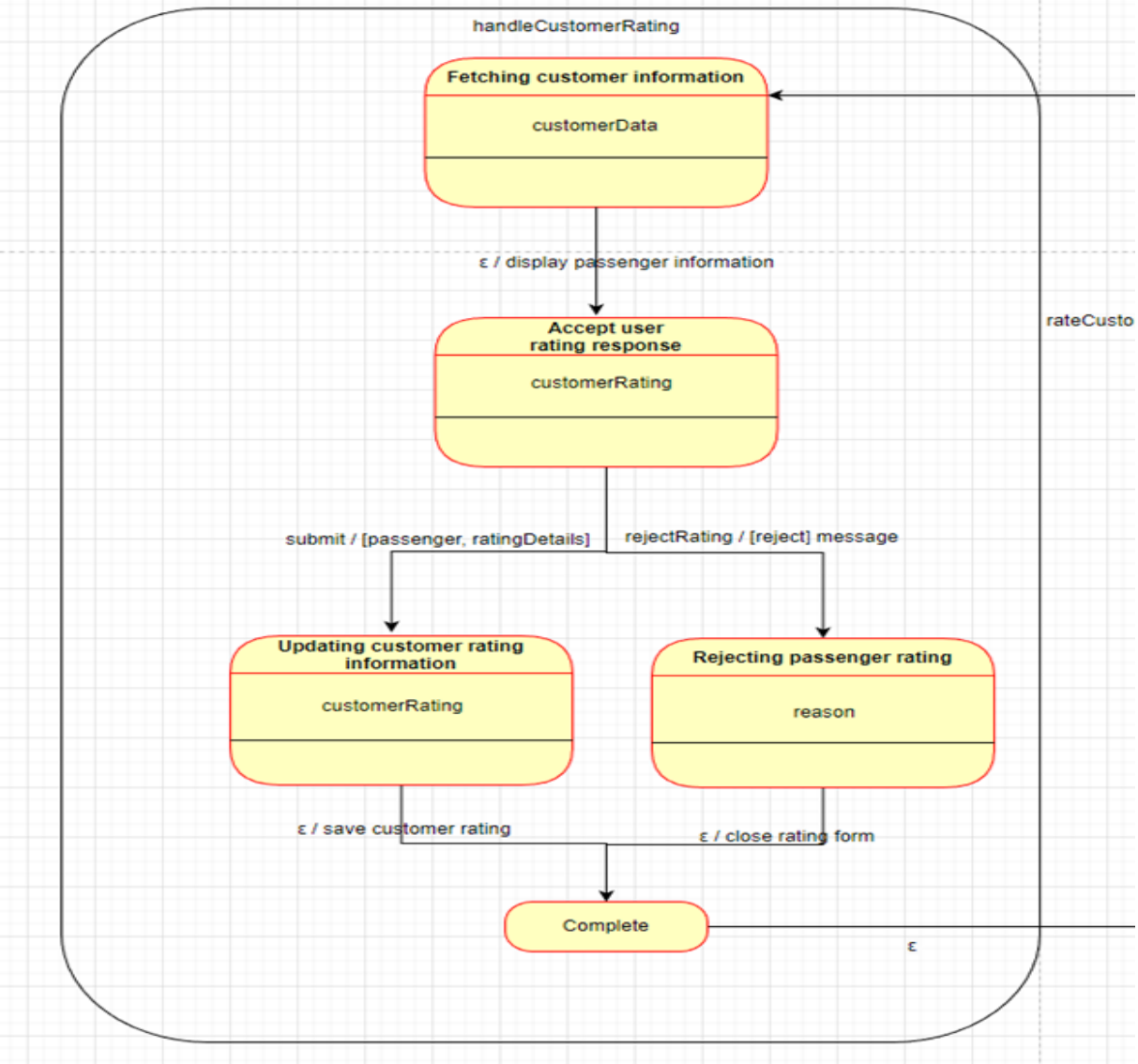
The breakdown for the rest of this technical document is as follows. In section 2, there are state charts that express what each controller class is doing, it communicates what was wanted from them in the CRC cards (in a more formal and serialized way) from Deliverable 2. In section 3, we go through each of the system's use cases and create a sequence diagram. This provides an understanding of which objects are going to be used/needed for a particular use case. In section 4, a detailed class diagram is shown. it complements the sequence diagrams, provides internal details of classes (methods, attributes), formalizes CRC cards, and communicates how classes are related to one another.

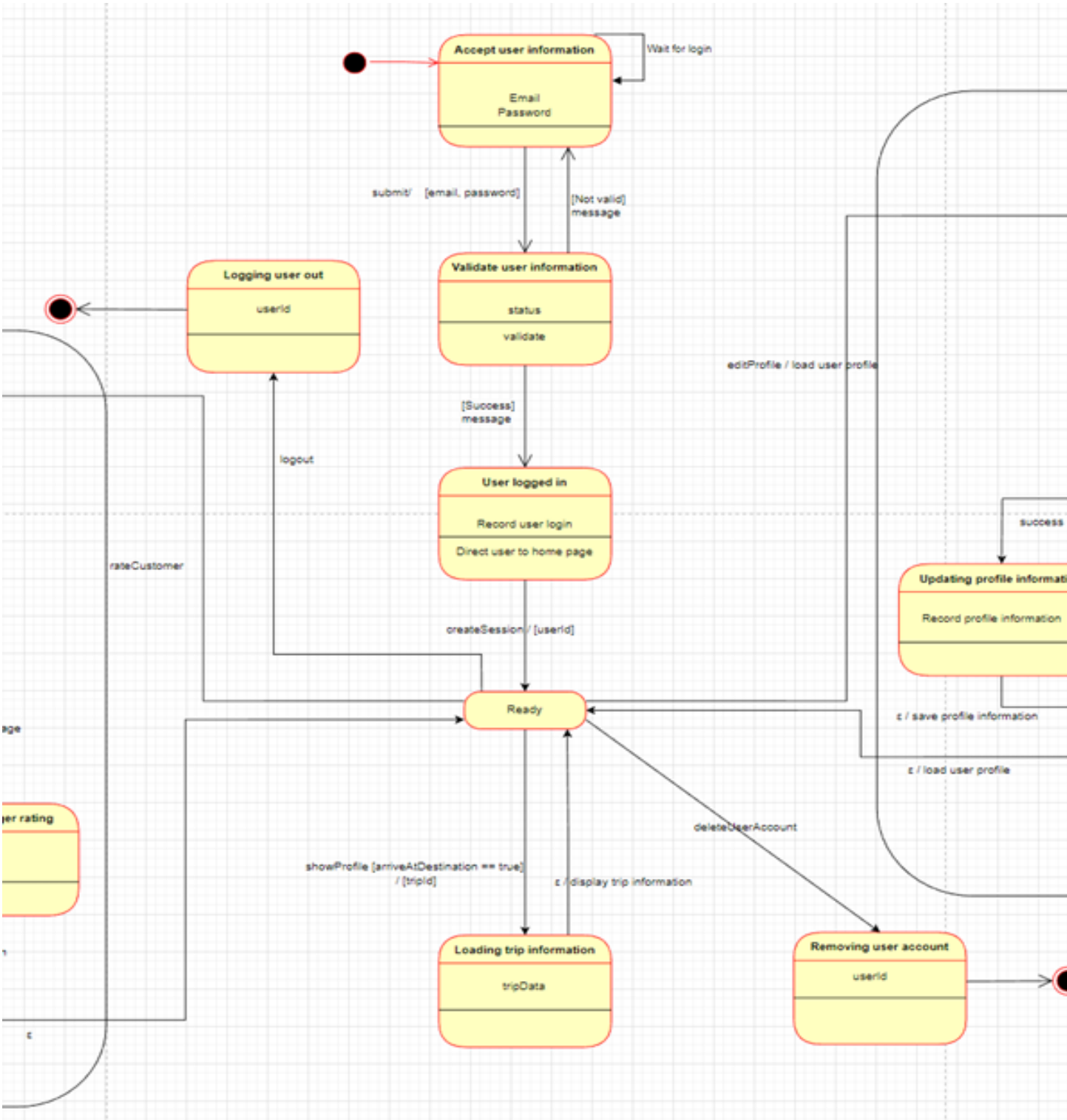
2 State Charts for Controller Classes

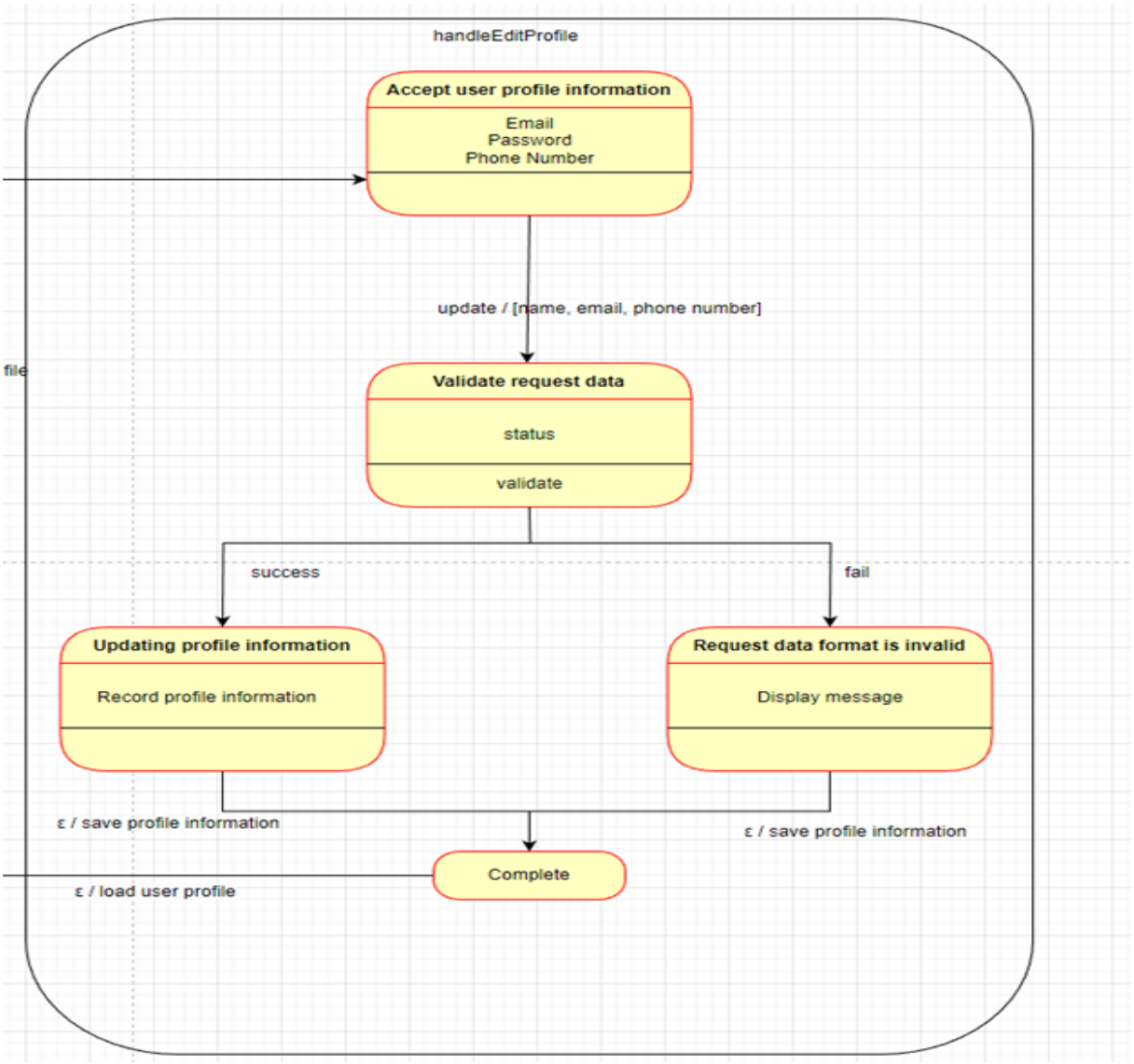
RegistrationController Class



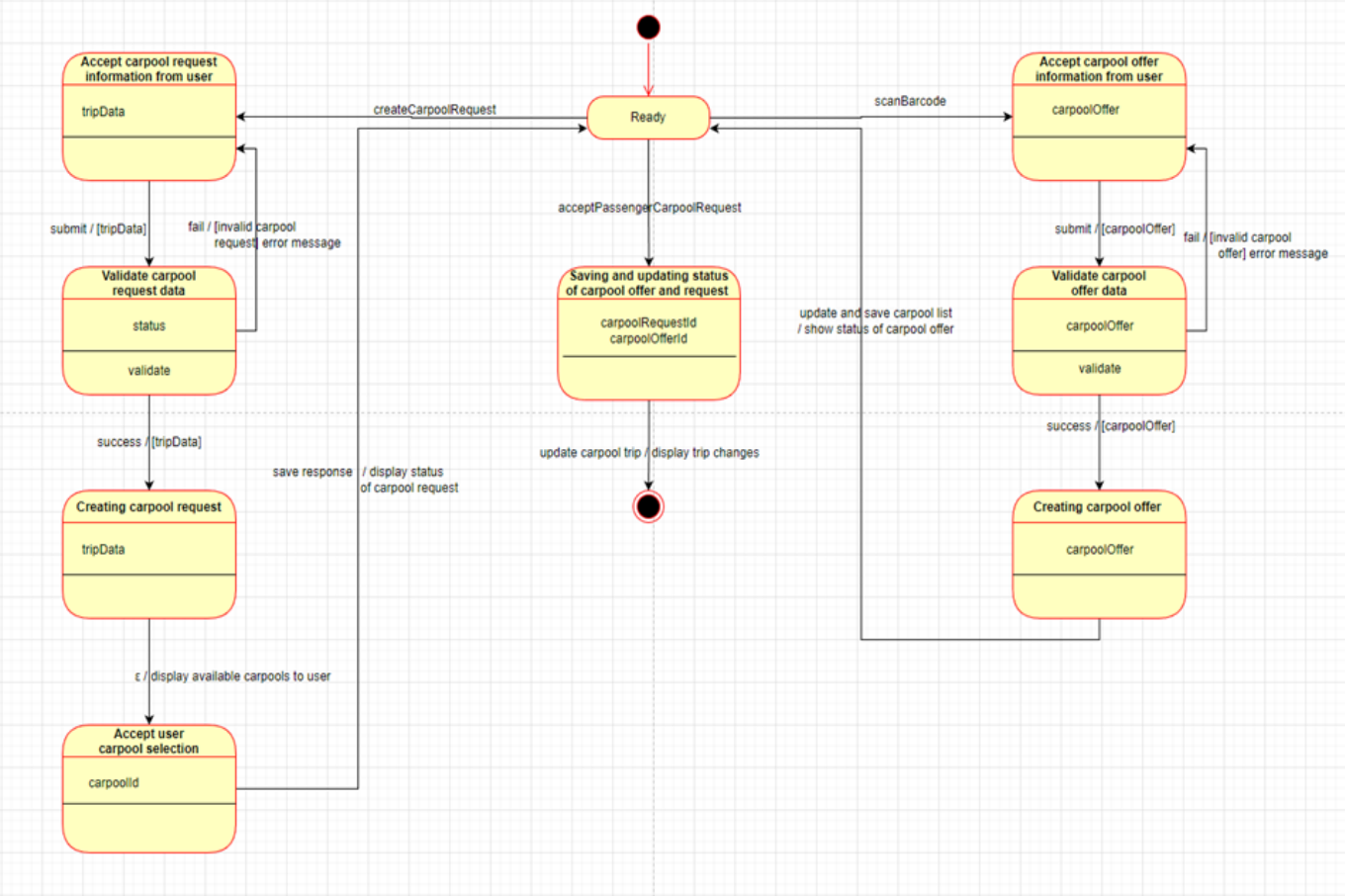
SessionController Class


handleCustomerRating
Fetching customer information
customerData
ε / display passenger information
Accept user
rating response
customerRating
rateCusto
submit / [passenger, ratingDetails]
rejectRating / [reject] message
Updating customer rating
information
customerRating
Rejecting passenger rating
reason
ε / save customer rating
ε / close rating form
Complete
ε

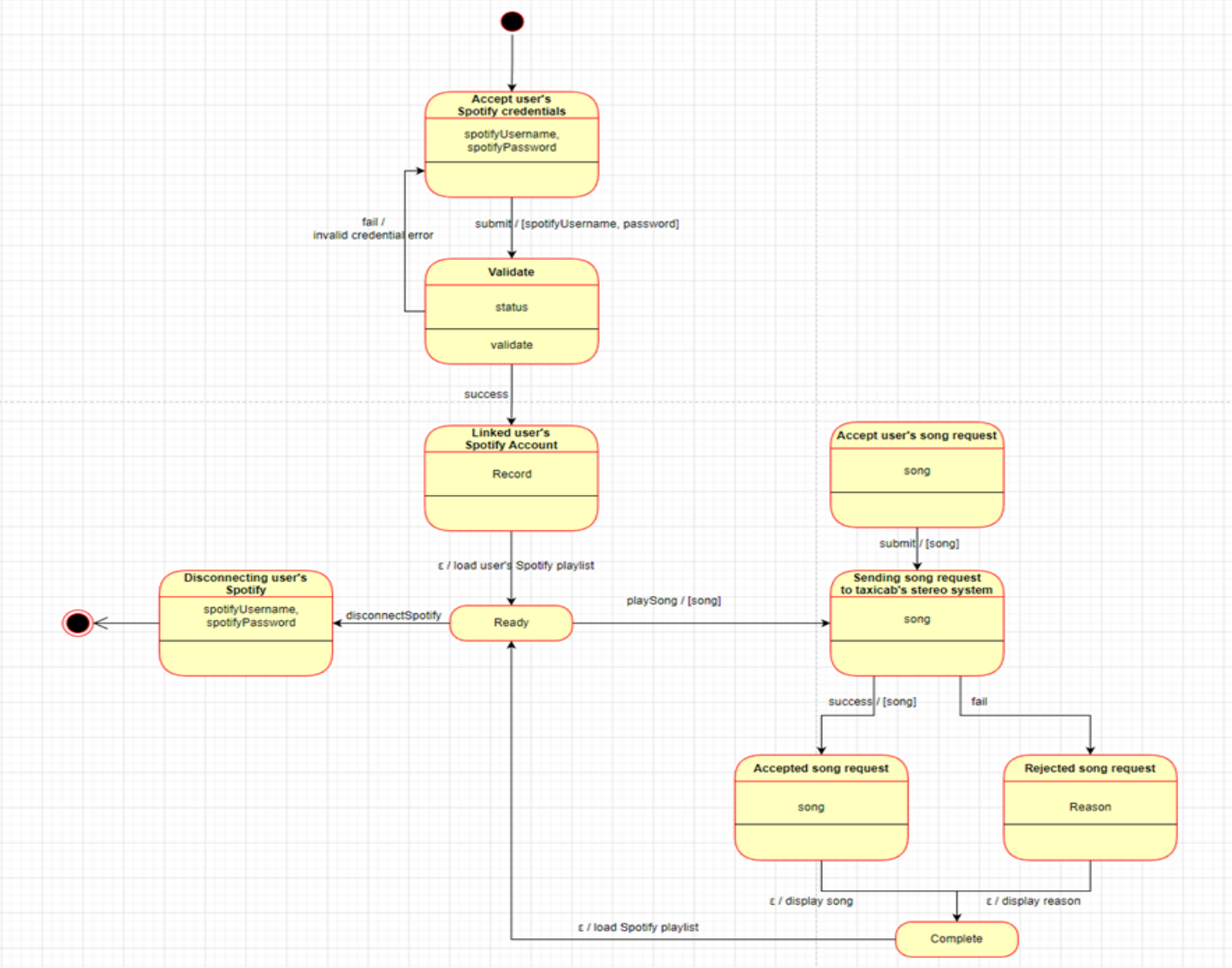

Accept user information
Email
Password
Wait for login
submit/ [email, password]
[Not valid] message
Validate user information
status
validate
Logging user out
userId
[Success] message
logout
User logged in
Record user login
Direct user to home page
createSession / [userId]
Ready
editProfile / load user profile
success
Updating profile informati
Record profile information
ε / save profile information
ε / load user profile
rateCustomer
deleteUserAccount
showProfile [arriveAtDestination == true] / [tripId]
ε / display trip information
Loading trip information
tripData
Removing user account
userId



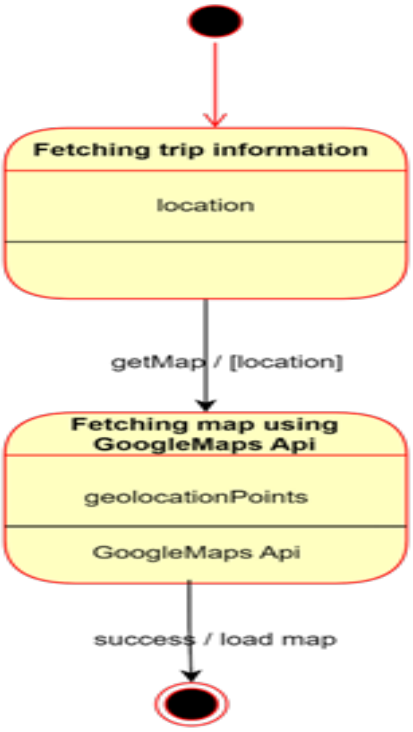
DispatcherController Class



MusicController Class

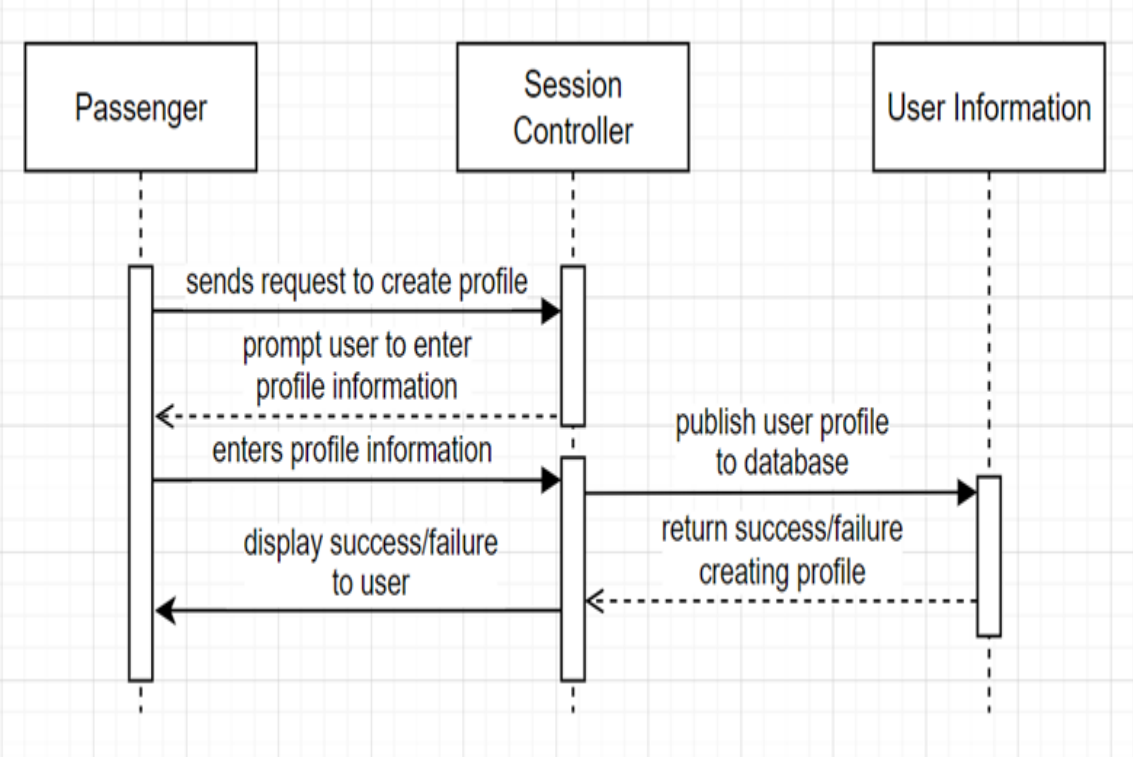


MapController Class

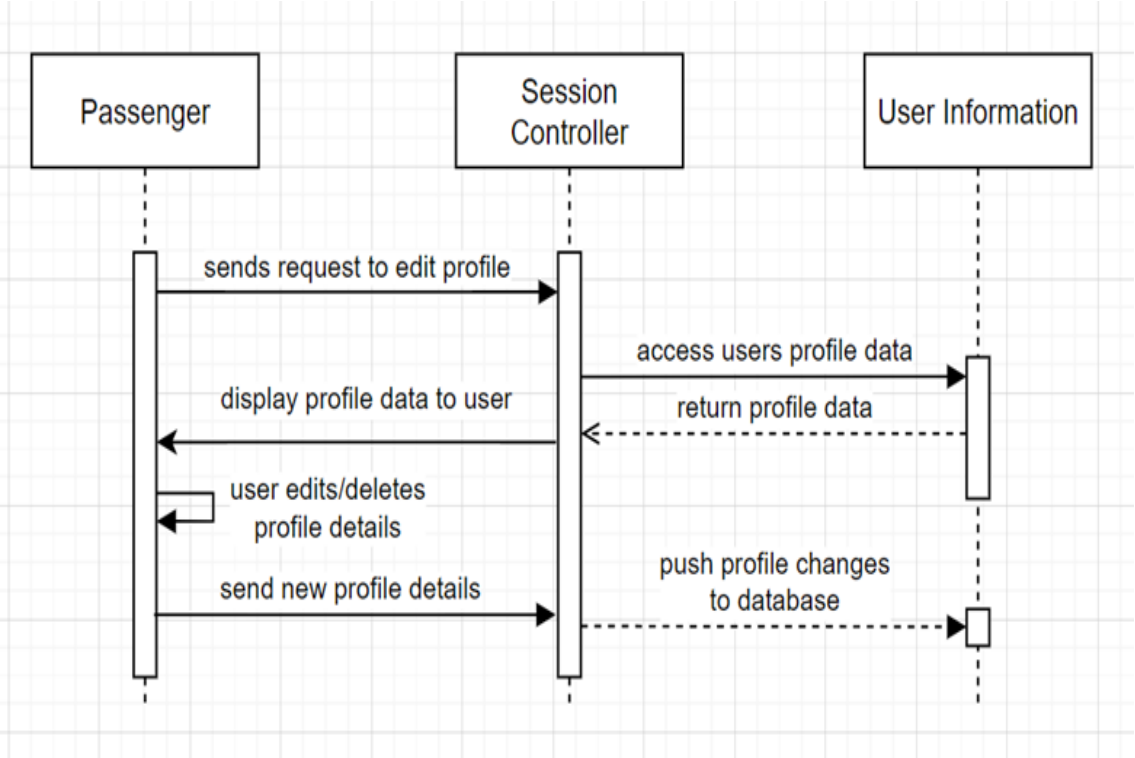


3 Sequence Diagrams

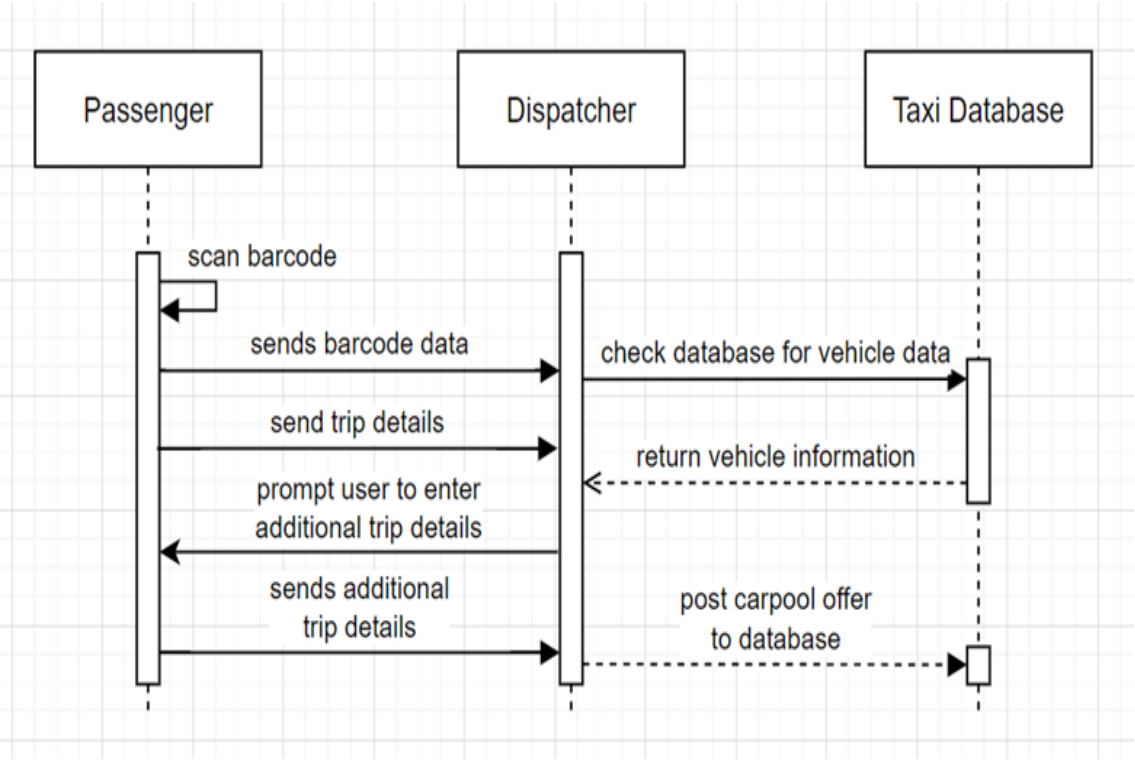
Use Case: Commuter Signs up



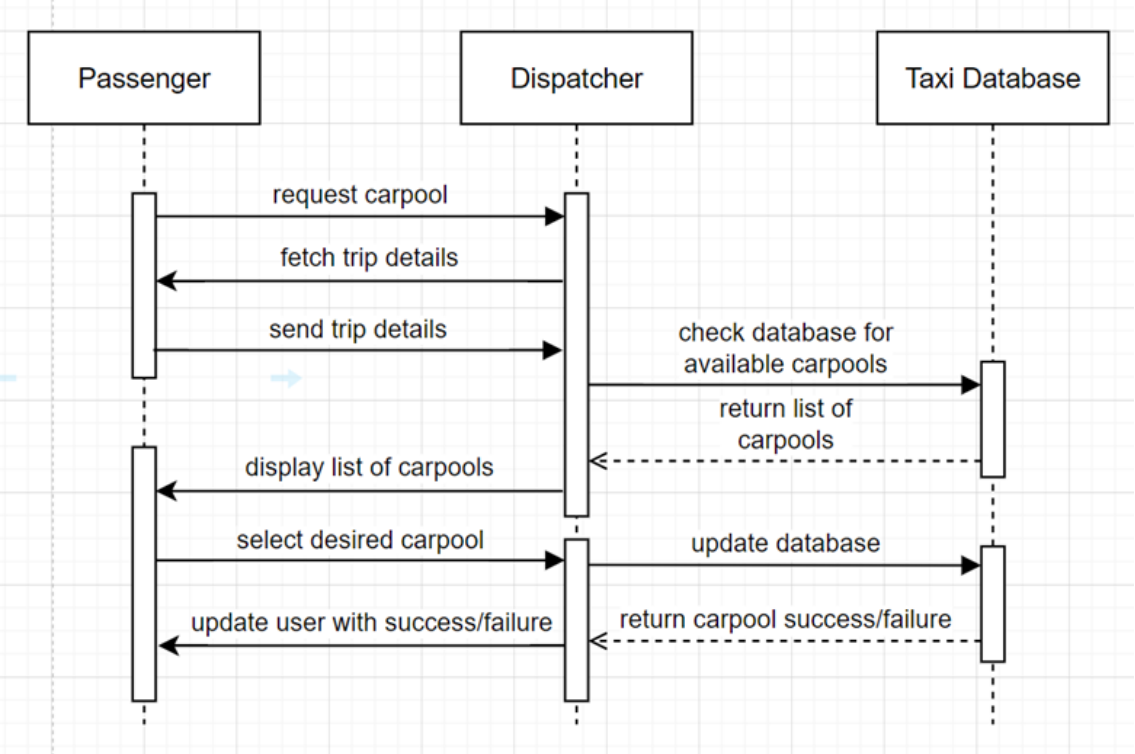
Use Case: User updates their Profile



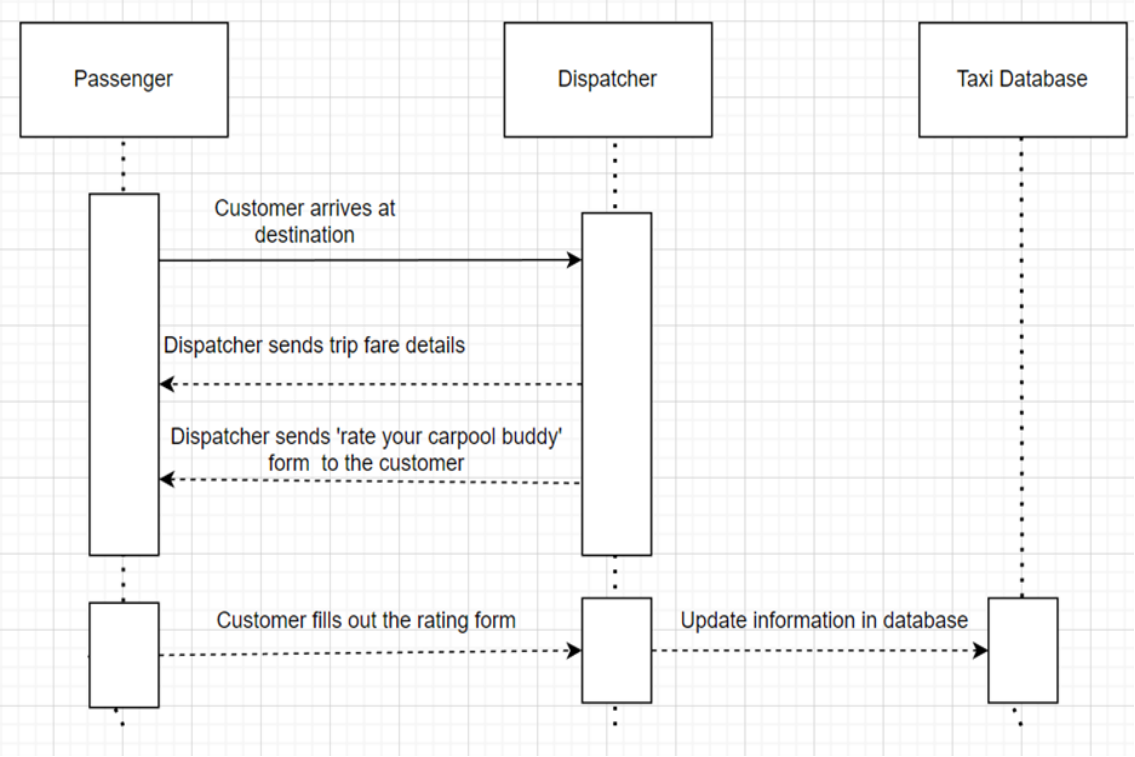
Use Case: Offer Taxi Carpool



Use Case: Request Carpool

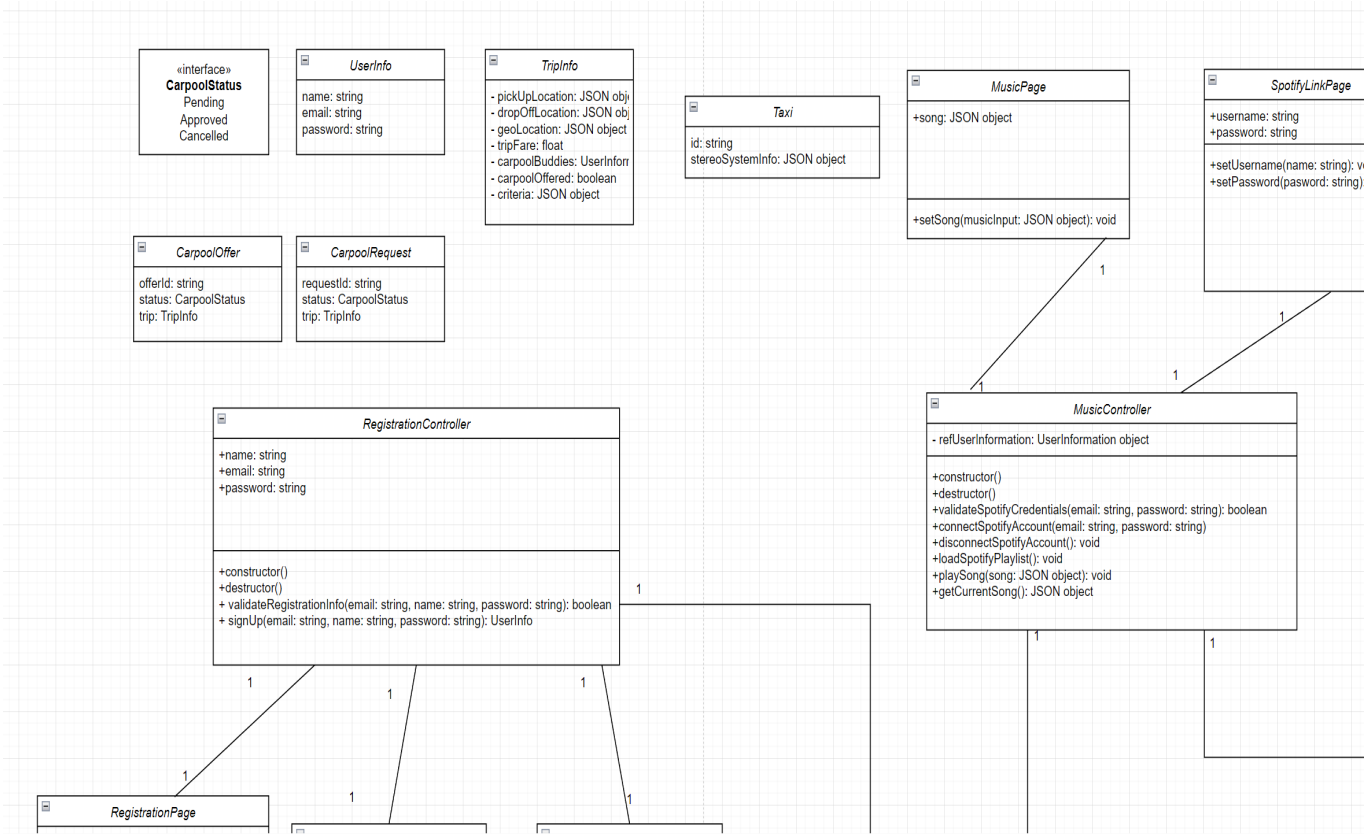


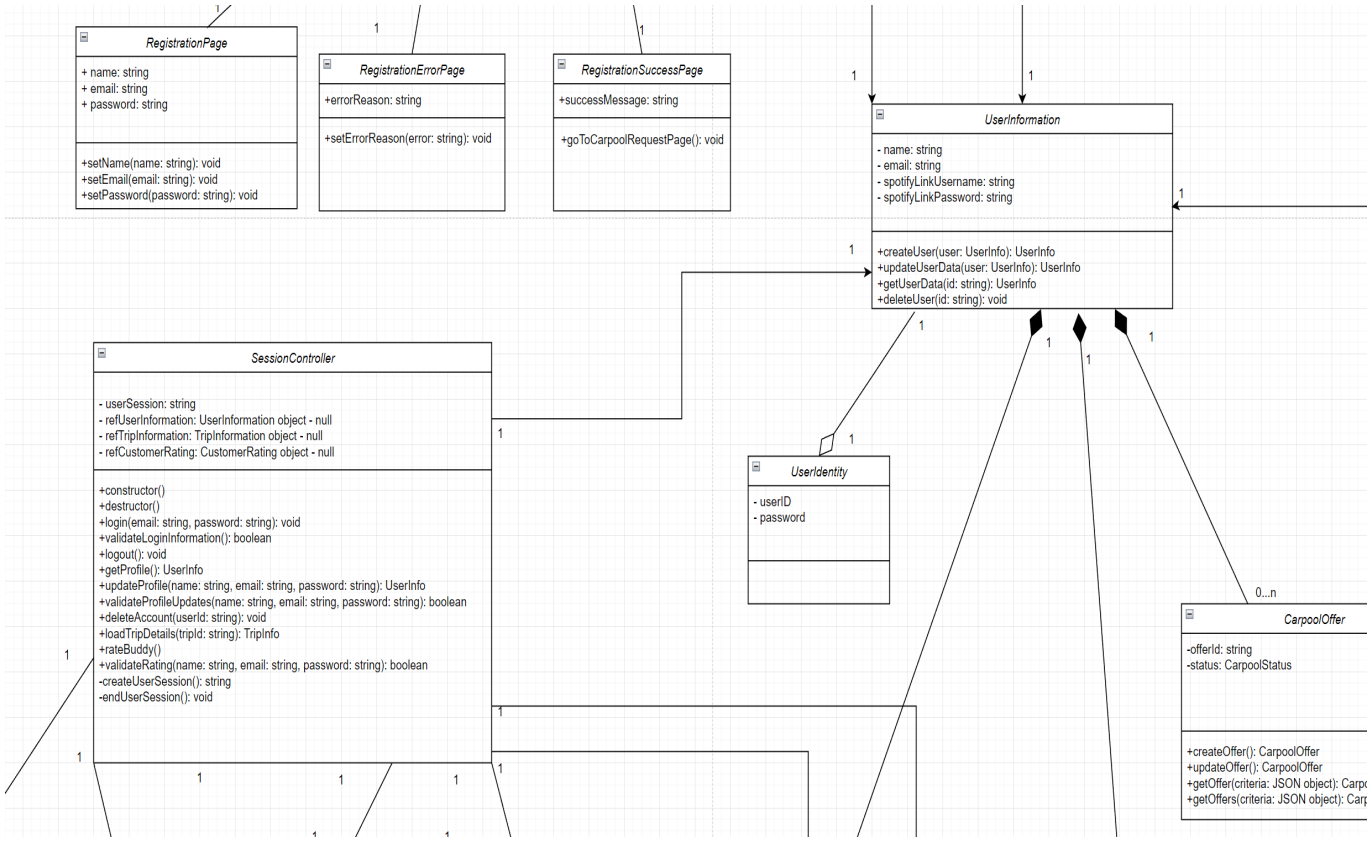
Use Case: Customer Arrives At Destination

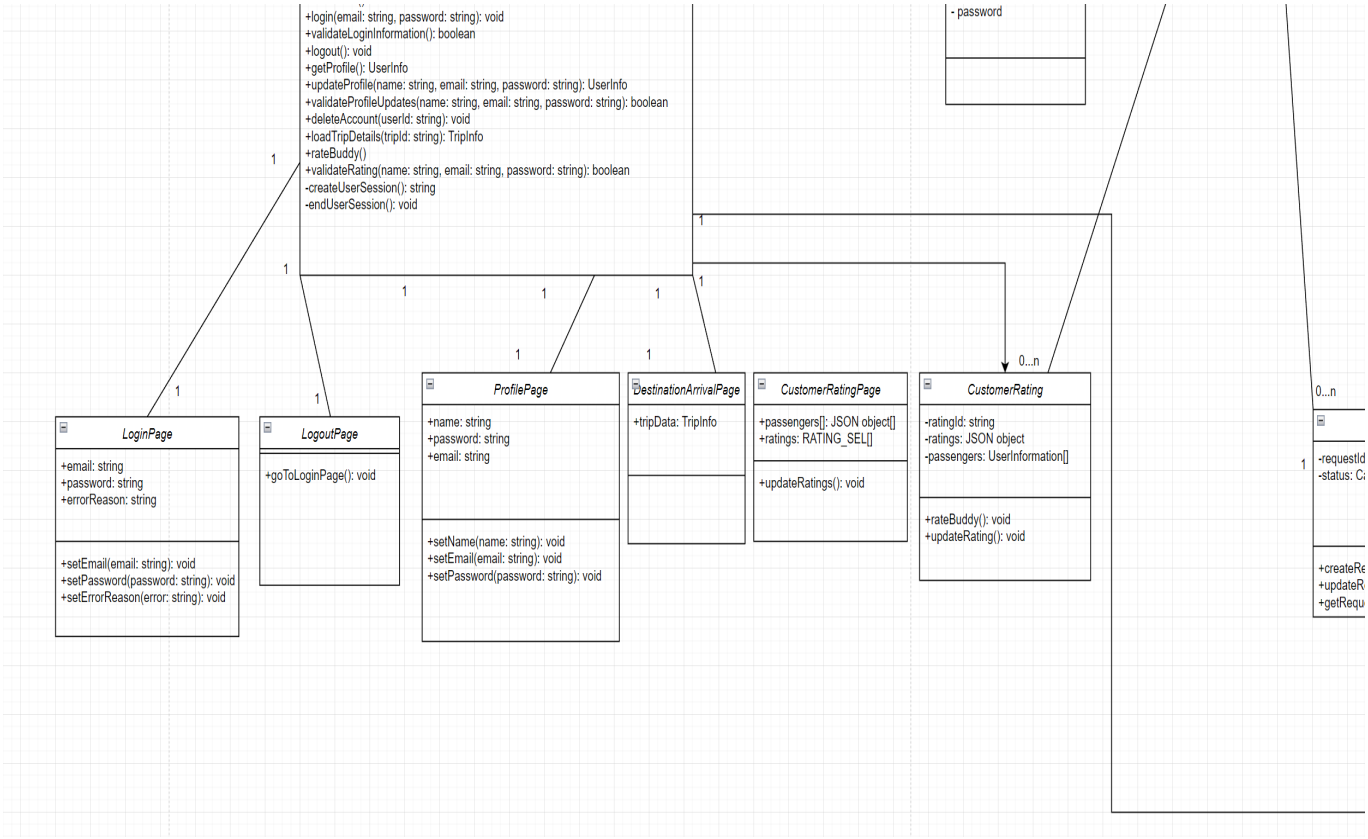


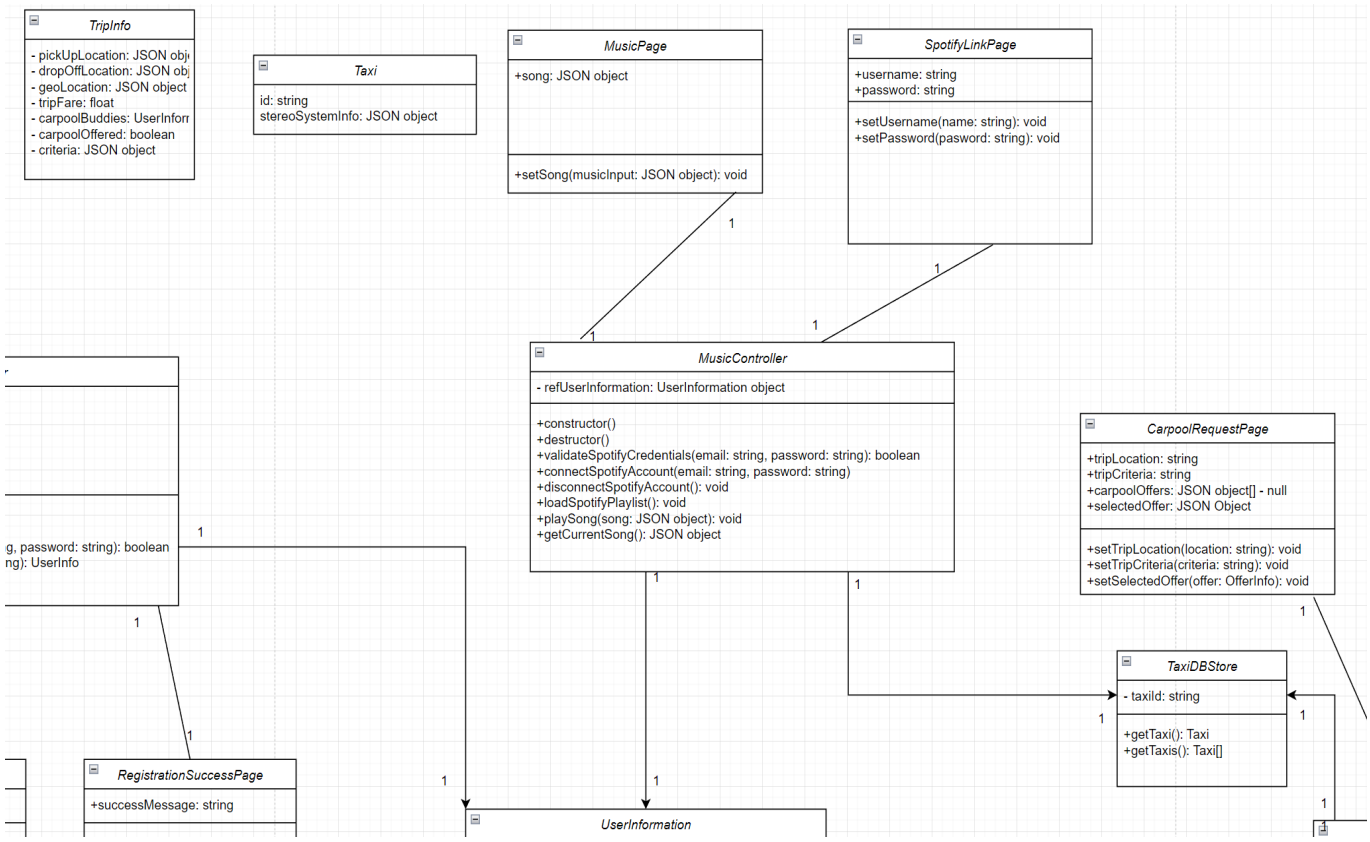
4 Detailed Class Diagram

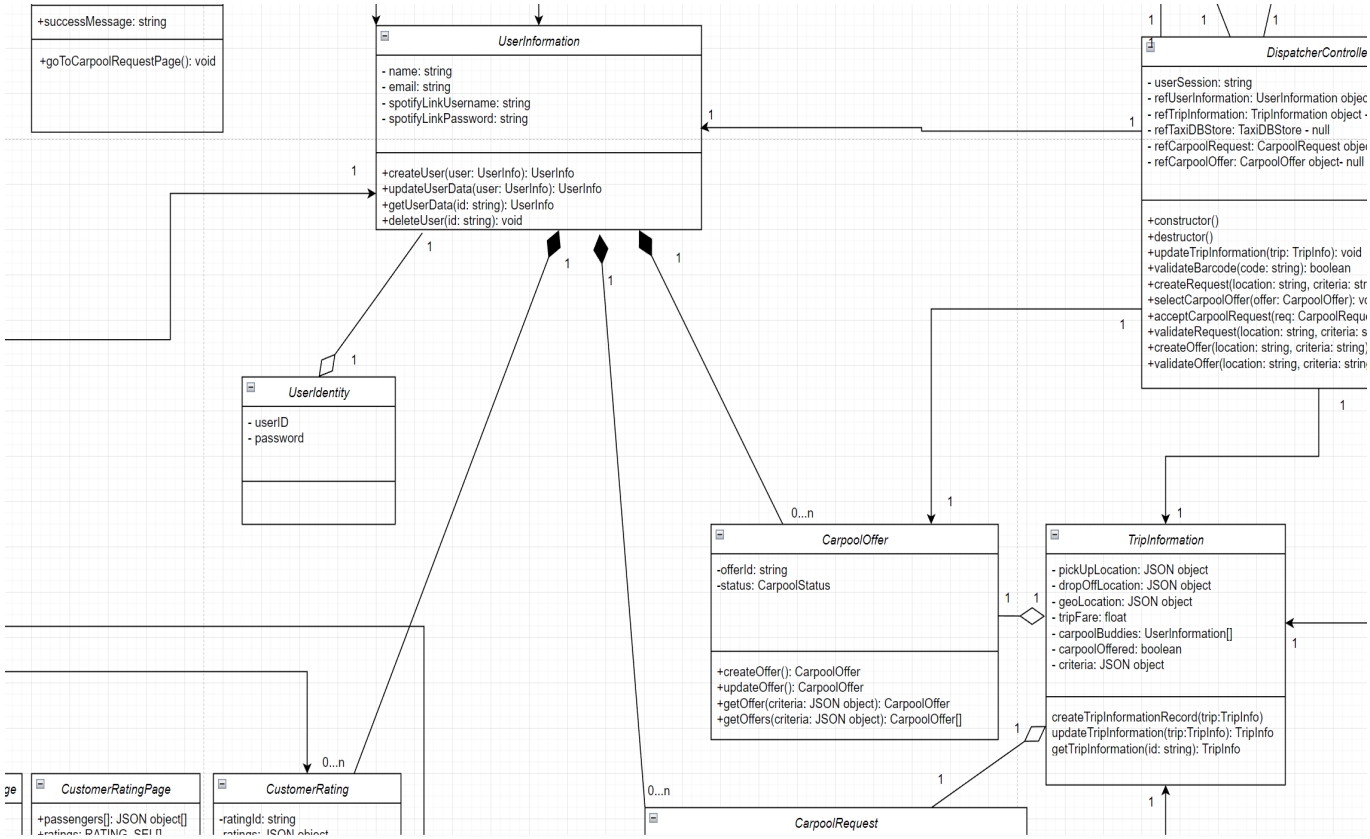
For a better view
https://tinyurl.com/3edy4kf2

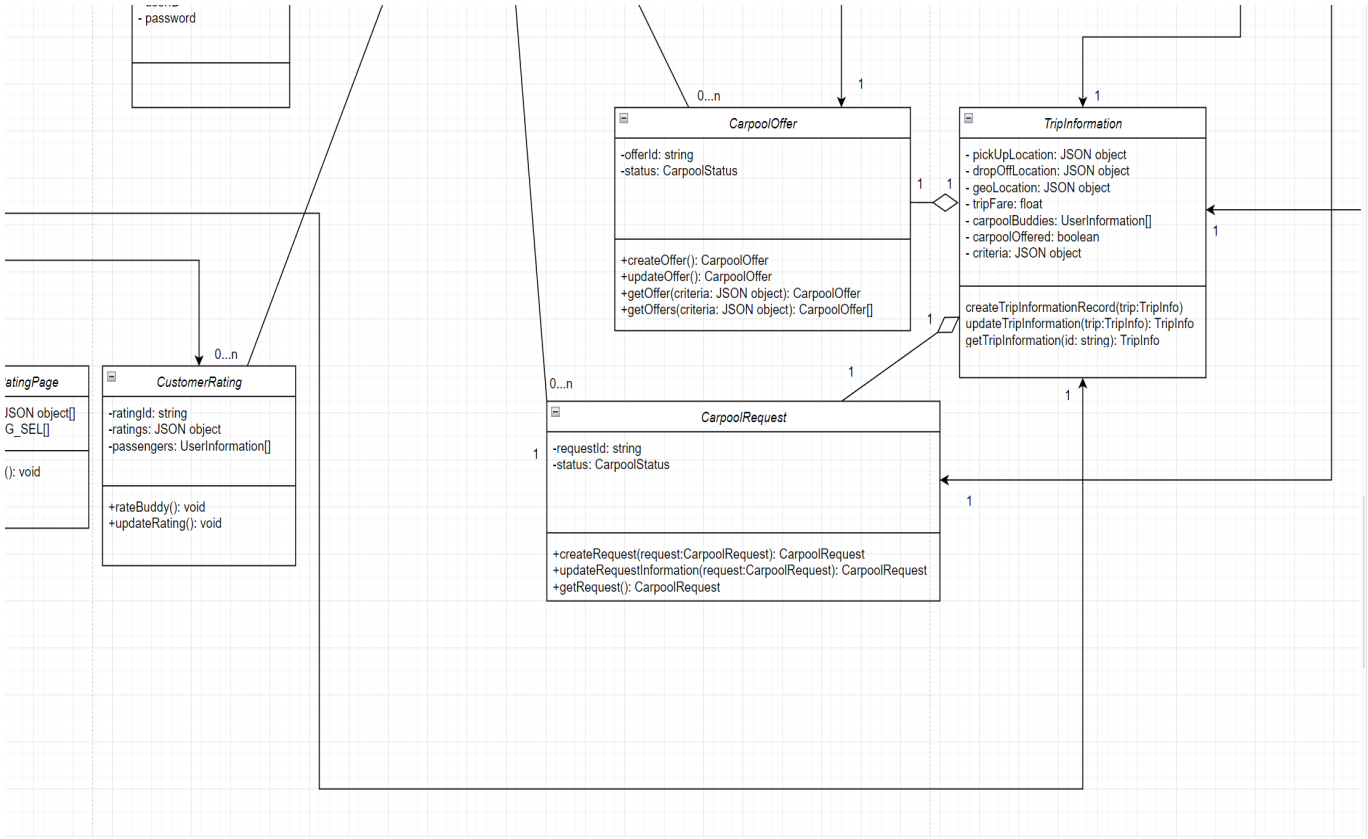


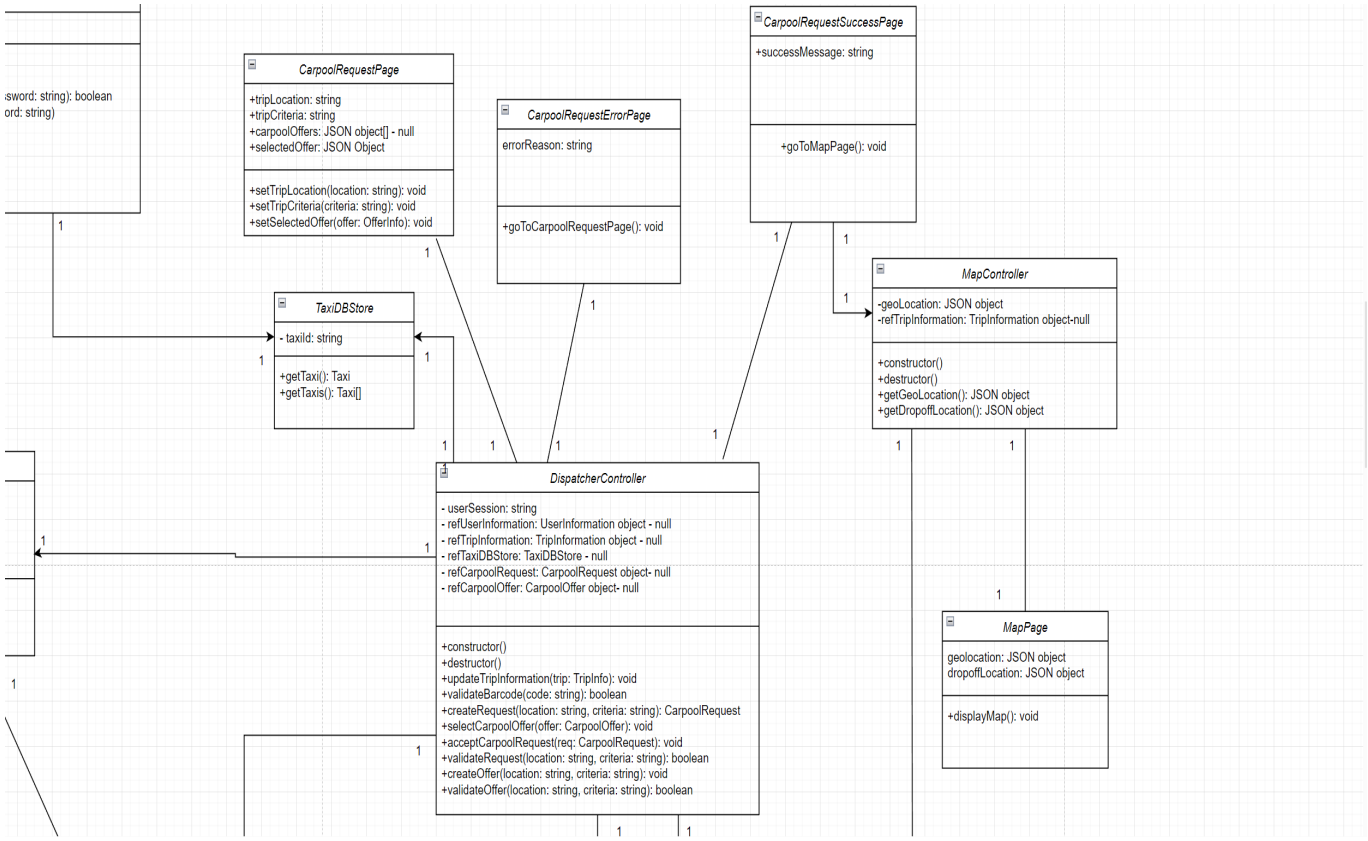

RegistrationPage
+ name: string
+ email: string
+ password: string
+setName(name: string): void
+setEmail(email: string): void
+setPassword(password: string): void
RegistrationErrorPage
+errorReason: string
+setErrorReason(error: string): void
RegistrationSuccessPage
+successMessage: string
+goToCarpoolRequestPage(): void
UserInformation
- name: string
- email: string
- spotifyLinkUsername: string
- spotifyLinkPassword: string
+createUser(user: UserInfo): UserInfo
+updateUserData(user: UserInfo): UserInfo
+getUserData(id: string): UserInfo
+deleteUser(id: string): void
SessionController
- userSession: string
- refUserInformation: UserInformation object - null
- refTripInformation: TripInformation object - null
- refCustomerRating: CustomerRating object - null
+constructor()
+destructor()
+login(email: string, password: string): void
+validateLoginInformation(): boolean
+logout(): void
+getProfile(): UserInfo
+updateProfile(name: string, email: string, password: string): UserInfo
+validateProfileUpdates(name: string, email: string, password: string): boolean
+deleteAccount(userId: string): void
+loadTripDetails(tripId: string): TripInfo
+rateBuddy()
+validateRating(name: string, email: string, password: string): boolean
-createUserSession(): string
-endUserSession(): void
UserIdentity
- userID
- password
CarpoolOffer
-offerId: string
-status: CarpoolStatus
+createOffer(): CarpoolOffer
+updateOffer(): CarpoolOffer
+getOffer(criteria: JSON object): Carp
+getOffers(criteria: JSON object): Carp
1
0...n

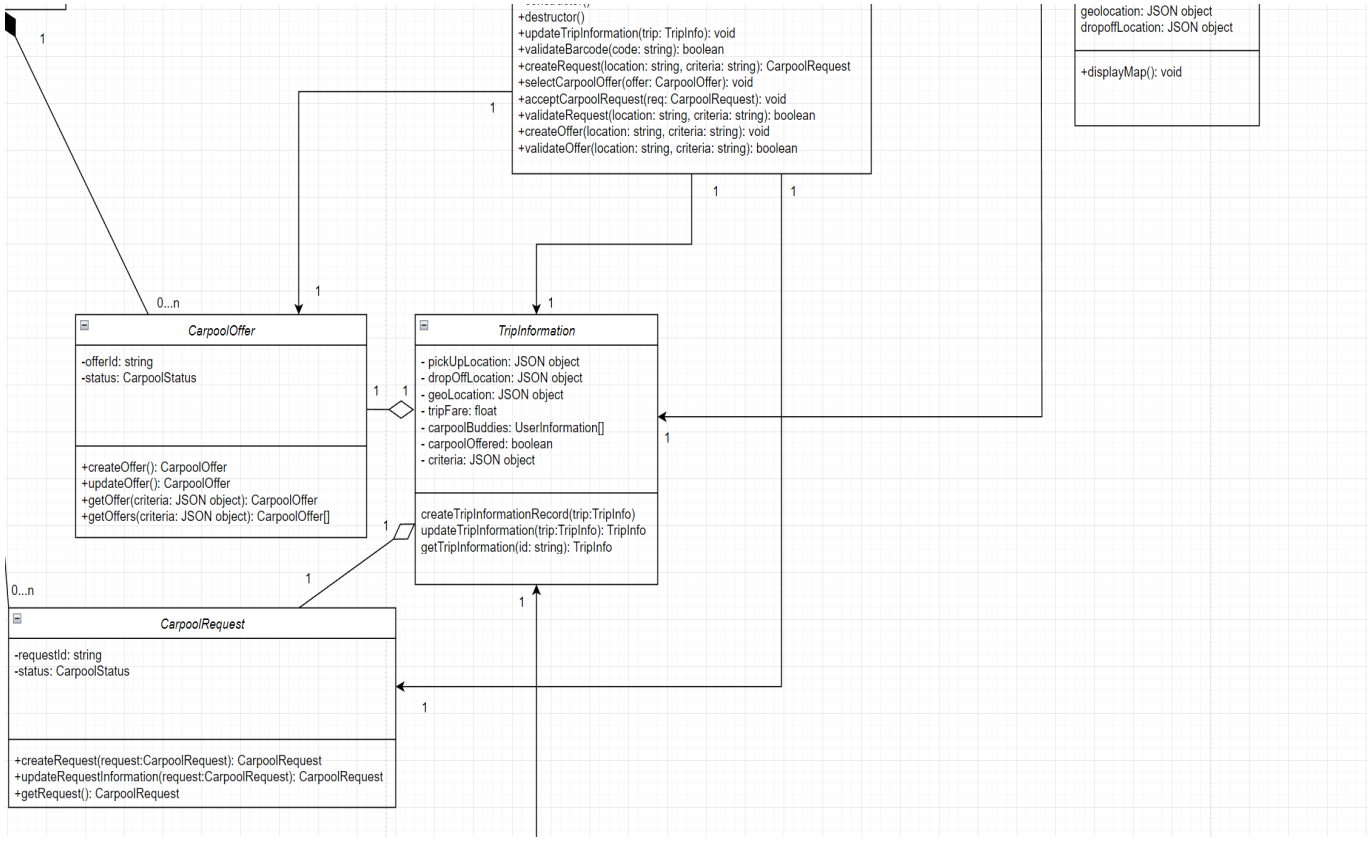

+login(email: string, password: string): void
+validateLoginInformation(): boolean
+logout(): void
+getProfile(): UserInfo
+updateProfile(name: string, email: string, password: string): UserInfo
+validateProfileUpdates(name: string, email: string, password: string): boolean
+deleteAccount(userId: string): void
+loadTripDetails(tripId: string): TripInfo
+rateBuddy()
+validateRating(name: string, email: string, password: string): boolean
-createUserSession(): string
-endUserSession(): void
- password
LoginPage
+email: string
+password: string
+errorReason: string
+setEmail(email: string): void
+setPassword(password: string): void
+setErrorReason(error: string): void
LogoutPage
+goToLoginPage(): void
ProfilePage
+name: string
+password: string
+email: string
+setName(name: string): void
+setEmail(email: string): void
+setPassword(password: string): void
DestinationArrivalPage
+tripData: TripInfo
CustomerRatingPage
+passengers[]: JSON object[]
+ratings: RATING_SEL[]
+updateRatings(): void
CustomerRating
-ratingId: string
-ratings: JSON object
-passengers: UserInformation[]
+rateBuddy(): void
+updateRating(): void
1
0...n


TripInfo
- pickUpLocation: JSON obj
- dropOffLocation: JSON obj
- geoLocation: JSON object
- tripFare: float
- carpoolBuddies: UserInfor
- carpoolOffered: boolean
- criteria: JSON object
Taxi
id: string
stereoSystemInfo: JSON object
MusicPage
+song: JSON object
+setSong(musicInput: JSON object): void
SpotifyLinkPage
+username: string
+password: string
+setUsername(name: string): void
+setPassword(pasword: string): void
MusicController
- refUserInformation: UserInformation object
+constructor()
+destructor()
+validateSpotifyCredentials(email: string, password: string): boolean
+connectSpotifyAccount(email: string, password: string)
+disconnectSpotifyAccount(): void
+loadSpotifyPlaylist(): void
+playSong(song: JSON object): void
+getCurrentSong(): JSON object
CarpoolRequestPage
+tripLocation: string
+tripCriteria: string
+carpoolOffers: JSON object[] - null
+selectedOffer: JSON Object
+setTripLocation(location: string): void
+setTripCriteria(criteria: string): void
+setSelectedOffer(offer: OfferInfo): void
g, password: string): boolean
ng): UserInfo
TaxiDBStore
- taxiId: string
+getTaxi(): Taxi
+getTaxis(): Taxi[]
RegistrationSuccessPage
+successMessage: string
UserInformation
1


+successMessage: string
+goToCarpoolRequestPage(): void
UserInformation
- name: string
- email: string
- spotifyLinkUsername: string
- spotifyLinkPassword: string
+createUser(user: UserInfo): UserInfo
+updateUserData(user: UserInfo): UserInfo
+getUserData(id: string): UserInfo
+deleteUser(id: string): void
DispatcherControlle
- userSession: string
- refUserInformation: UserInformation objec
- refTripInformation: TripInformation object -
- refTaxiDBStore: TaxiDBStore - null
- refCarpoolRequest: CarpoolRequest obje
- refCarpoolOffer: CarpoolOffer object- null
+constructor()
+destructor()
+updateTripInformation(trip: TripInfo): void
+validateBarcode(code: string): boolean
+createRequest(location: string, criteria: str
+selectCarpoolOffer(offer: CarpoolOffer): vo
+acceptCarpoolRequest(req: CarpoolRequ
+validateRequest(location: string, criteria: s
+createOffer(location: string, criteria: string
+validateOffer(location: string, criteria: strin
UserIdentity
- userID
- password
CarpoolOffer
-offerId: string
-status: CarpoolStatus
+createOffer(): CarpoolOffer
+updateOffer(): CarpoolOffer
+getOffer(criteria: JSON object): CarpoolOffer
+getOffers(criteria: JSON object): CarpoolOffer[]
TripInformation
- pickUpLocation: JSON object
- dropOffLocation: JSON object
- geoLocation: JSON object
- tripFare: float
- carpoolBuddies: UserInformation[]
- carpoolOffered: boolean
- criteria: JSON object
createTripInformationRecord(trip:TripInfo)
updateTripInformation(trip:TripInfo): TripInfo
getTripInformation(id: string): TripInfo
CustomerRatingPage
+passengers[]: JSON object[]
CustomerRating
-ratingId: string
CarpoolRequest
0...n
1


- password
0...n
1
1
CarpoolOffer
-offerId: string
-status: CarpoolStatus
+createOffer(): CarpoolOffer
+updateOffer(): CarpoolOffer
+getOffer(criteria: JSON object): CarpoolOffer
+getOffers(criteria: JSON object): CarpoolOffer[]
1
1
TripInformation
- pickUpLocation: JSON object
- dropOffLocation: JSON object
- geoLocation: JSON object
- tripFare: float
- carpoolBuddies: UserInformation[]
- carpoolOffered: boolean
- criteria: JSON object
createTripInformationRecord(trip:TripInfo)
updateTripInformation(trip:TripInfo): TripInfo
getTripInformation(id: string): TripInfo
1
1
1
atingPage
JSON object[]
G_SEL[]
(): void
0...n
CustomerRating
-ratingId: string
-ratings: JSON object
-passengers: UserInformation[]
+rateBuddy(): void
+updateRating(): void
0...n
1
1
CarpoolRequest
-requestId: string
-status: CarpoolStatus
+createRequest(request:CarpoolRequest): CarpoolRequest
+updateRequestInformation(request:CarpoolRequest): CarpoolRequest
+getRequest(): CarpoolRequest
1


sword: string): boolean
ord: string)
CarpoolRequestPage
+tripLocation: string
+tripCriteria: string
+carpoolOffers: JSON object[] - null
+selectedOffer: JSON Object
+setTripLocation(location: string): void
+setTripCriteria(criteria: string): void
+setSelectedOffer(offer: OfferInfo): void
CarpoolRequestErrorPage
errorReason: string
+goToCarpoolRequestPage(): void
CarpoolRequestSuccessPage
+successMessage: string
+goToMapPage(): void
MapController
-geoLocation: JSON object
-refTripInformation: TripInformation object-null
+constructor()
+destructor()
+getGeoLocation(): JSON object
+getDropoffLocation(): JSON object
TaxiDBStore
- taxiId: string
+getTaxi(): Taxi
+getTaxis(): Taxi[]
DispatcherController
- userSession: string
- refUserInformation: UserInformation object - null
- refTripInformation: TripInformation object - null
- refTaxiDBStore: TaxiDBStore - null
- refCarpoolRequest: CarpoolRequest object- null
- refCarpoolOffer: CarpoolOffer object- null
+constructor()
+destructor()
+updateTripInformation(trip: TripInfo): void
+validateBarcode(code: string): boolean
+createRequest(location: string, criteria: string): CarpoolRequest
+selectCarpoolOffer(offer: CarpoolOffer): void
+acceptCarpoolRequest(req: CarpoolRequest): void
+validateRequest(location: string, criteria: string): boolean
+createOffer(location: string, criteria: string): void
+validateOffer(location: string, criteria: string): boolean
MapPage
geolocation: JSON object
dropoffLocation: JSON object
+displayMap(): void



The way the class diagram is to be read is as follows, the first 3 are considered left top, left middle, left bottom, the next three are middle top, middle middle, middle bottom, and the last 2 are right top, right middle

A Division of Labour

Mohamad-Hassan Bahsoun Sections: Class Diagram

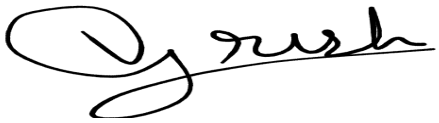
Rishi Vaya Sections: Sequence Diagrams (4, 5), state diagram

Isaac Giles Sections: Sequence Diagrams (1, 2, 3)

Umang Rajkarnikar Sections: State Chart, Class Diagram

Mohamad-Hassan Bahsoun

Rishi Vaya



Isaac Giles

Umang Rajkarnikar