Deliverable #3

SE 3A04: Software Design II – Large System Design

Tutorial Number: T01
Group Number: Group 5
Group Members: Arash, Daniel, Matthew, Waleed, Willie

1 Introduction

1.1 Purpose

The purpose of this file is to document the low-level architecture of the software system behind our taxi-sharing application. It is to serve as a detailed guide for implementation to developers.

The intended audience for this document is primarily the developers responsible for building the app. Additionally, the team responsible for quality assurance and testing is an audience, as it is important to understand how the system is supposed to appear and function to help determine adequate tests that must be performed.

1.2 System Description

The system is a taxi carpool matching system that allows users to give others an opportunity to carpool with them. This is done by using the application to “offer a carpool” to other application users that would like to also reach the same destination. Main riders “offer a carpool”, and other users must “request a carpool” to be able to be considered additional riders. Within the carpooling vehicle, riders will be able to access an additional gambling feature that allows riders to wager their fares with other riders, this way the challenger can either reduce or increase their fare.

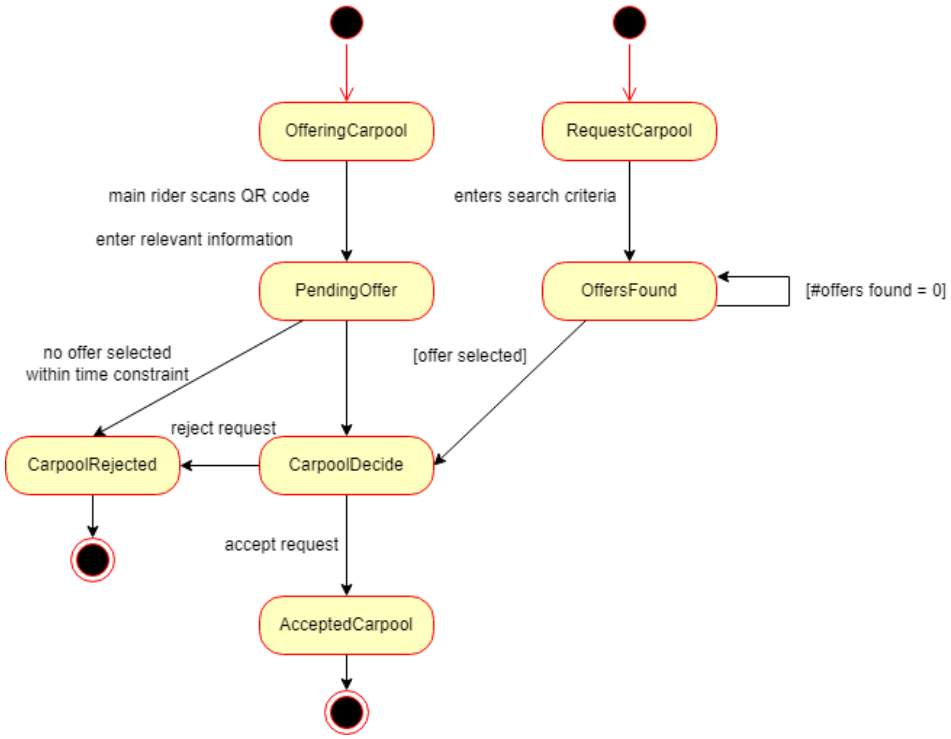
To allow users to request and match with each other for carpooling, each user must have their own account and profile to distinctively identify one another, and maintain past ride history. The system contains an account login and registration, profile management, ride offer and requests, taxi map, and gambling game.

1.3 Overview

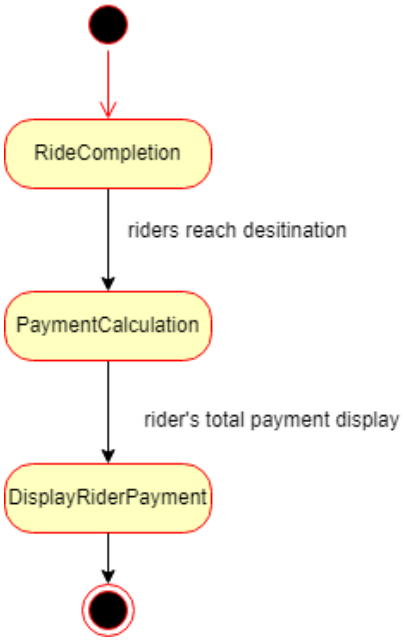
The rest of the document contains detailed UML diagrams of the software system. These include State Charts, Sequence Diagrams, and a Detailed Class Diagram of how the system application will function between subsystems, and how it will function individually for specific pages of the application.

2 State Charts for Controller Classes

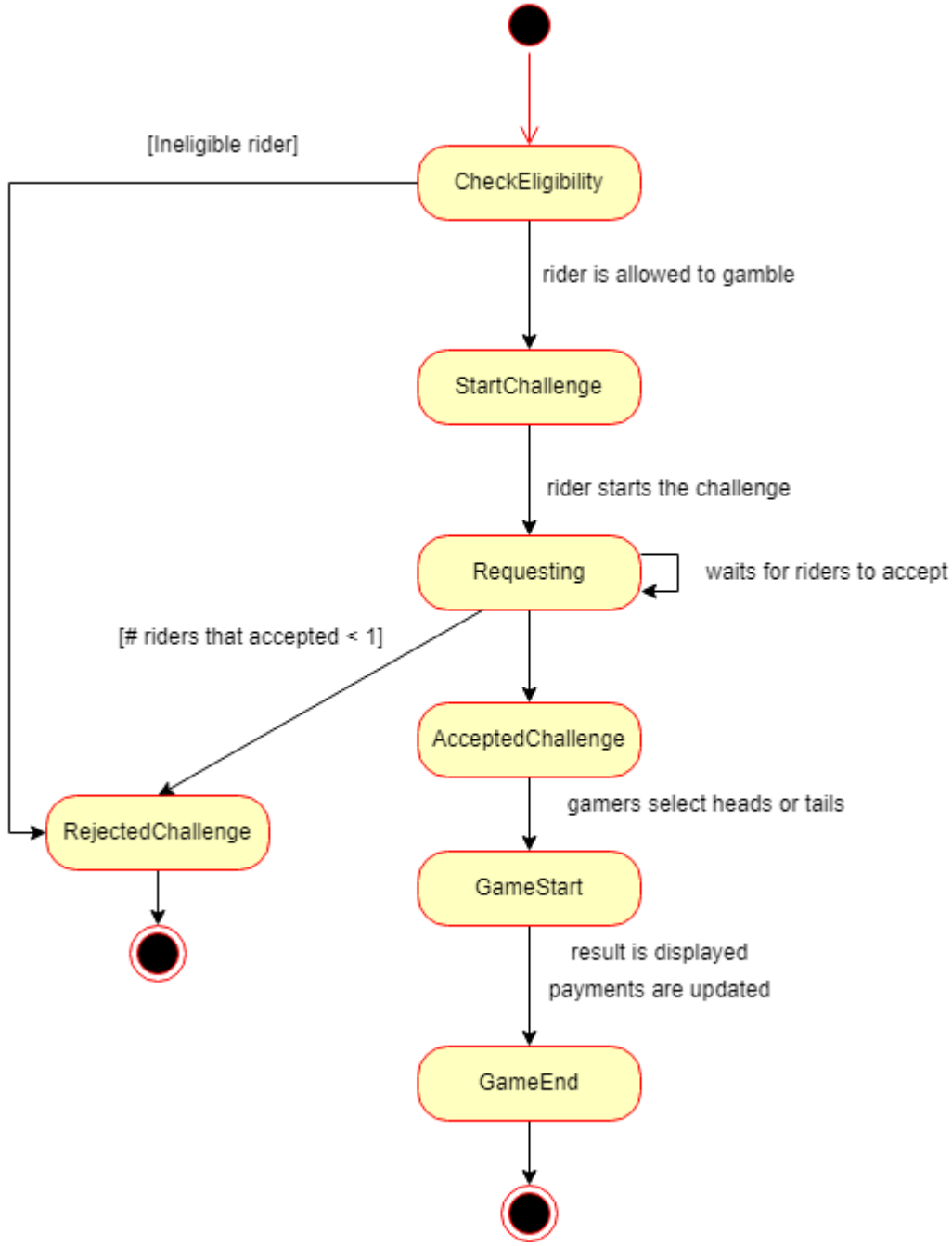
Dispatch system controller class State Chart:



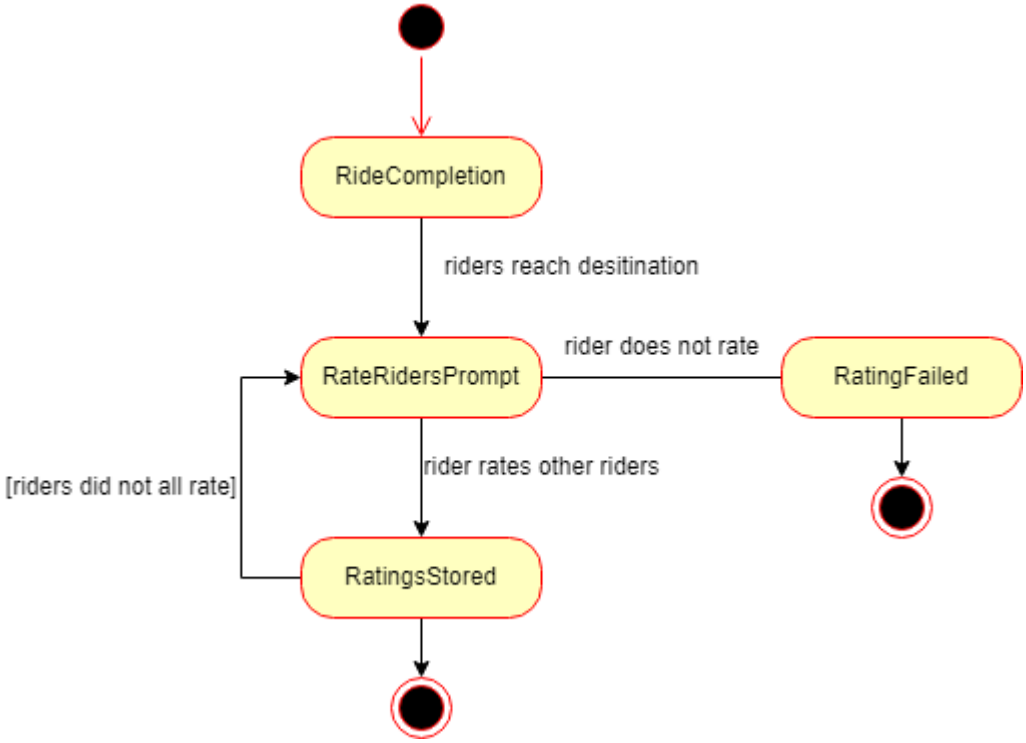
Payment system controller State Chart:



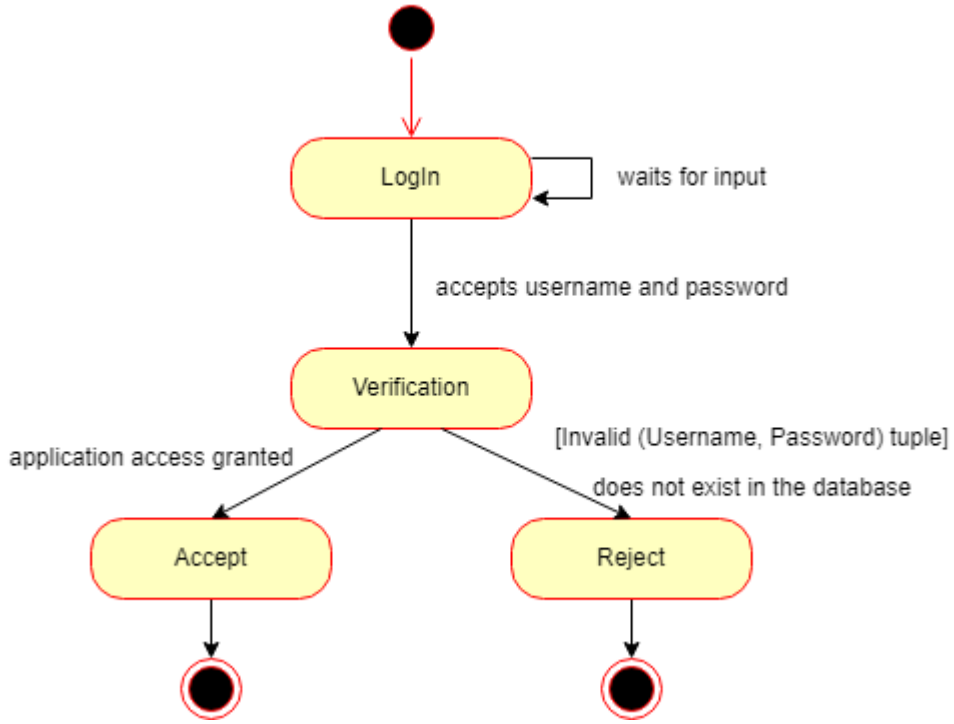
Gambling challenge controller State Chart:



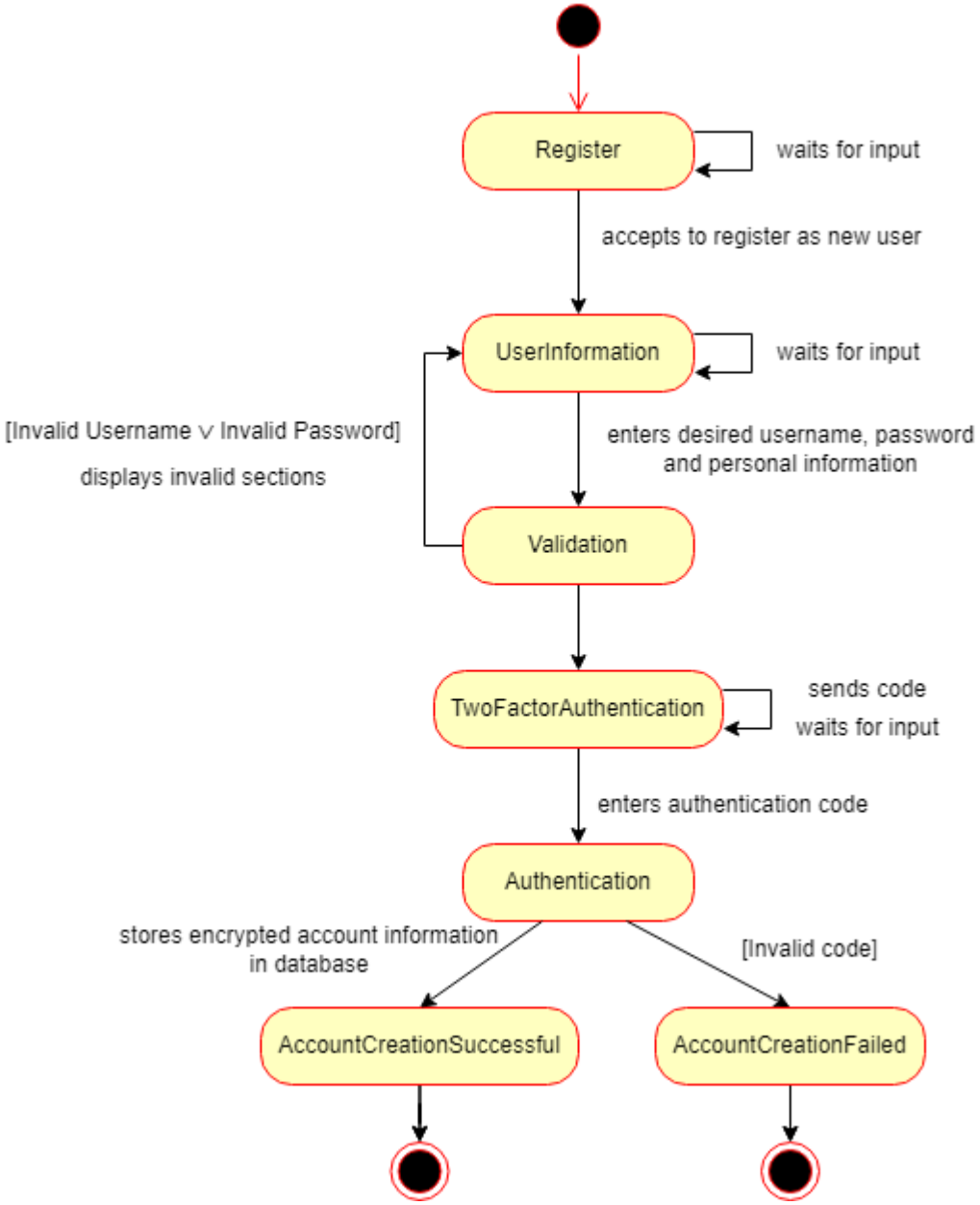
Passenger rating controller State Chart:



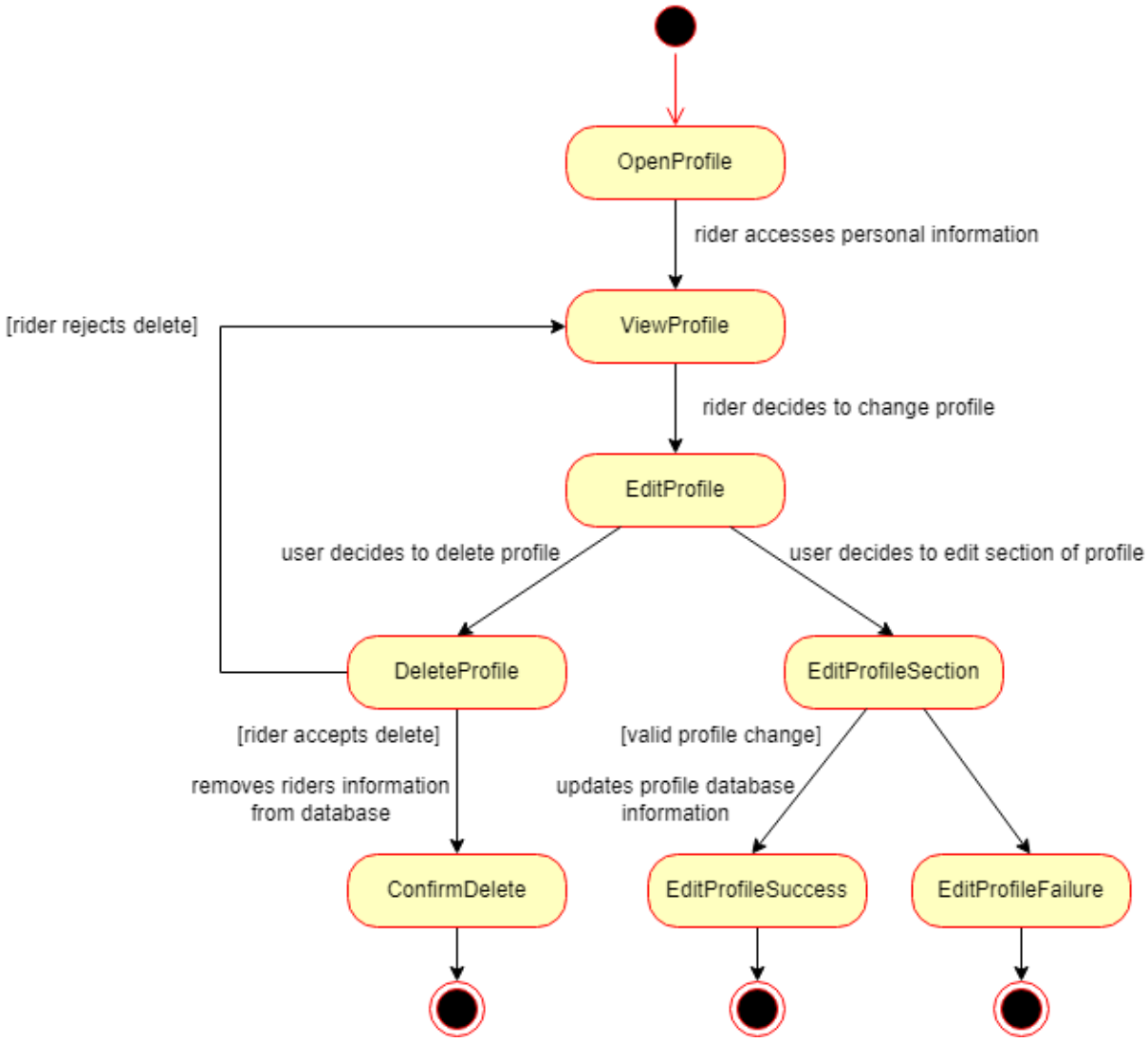
Login controller State Chart:



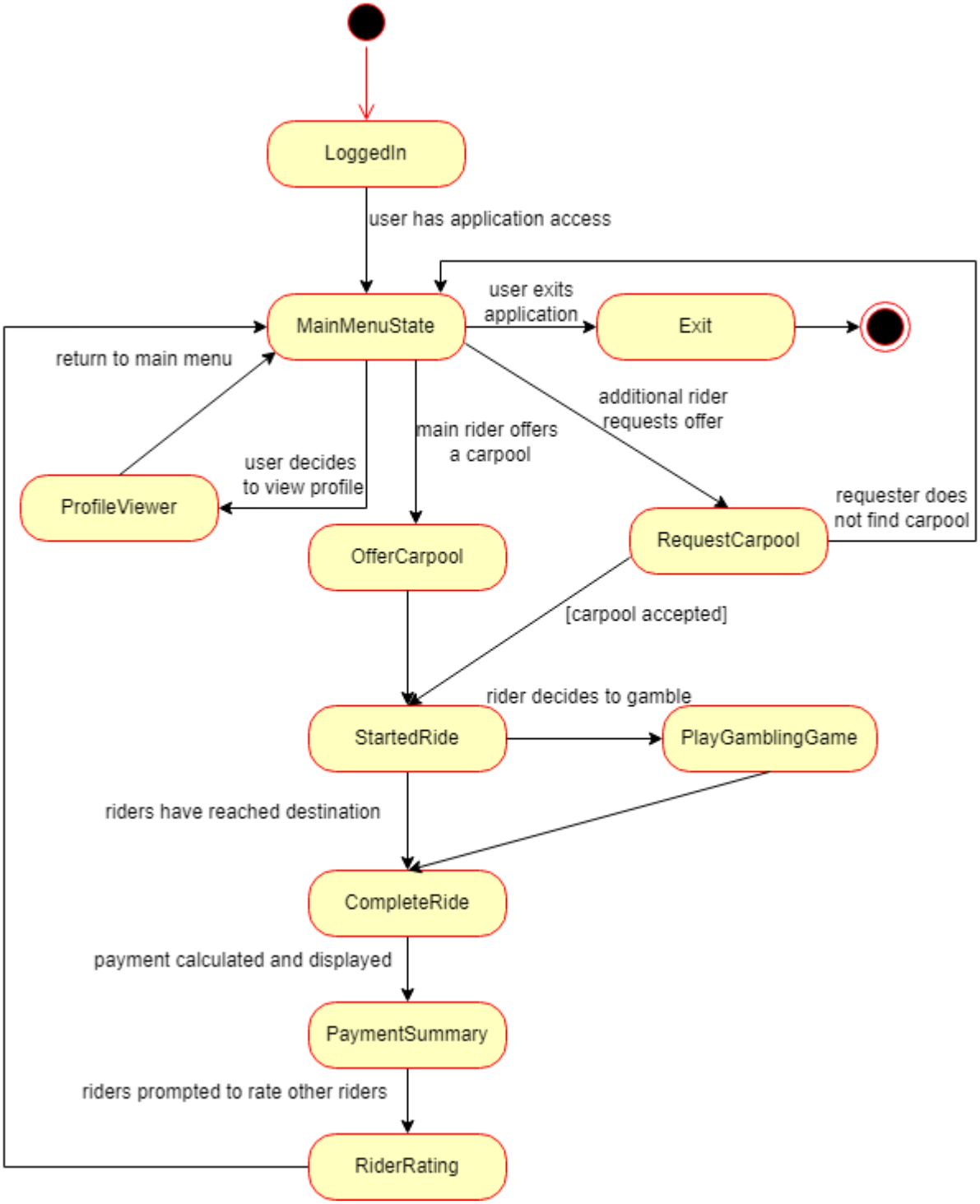
User registration controller State Chart:



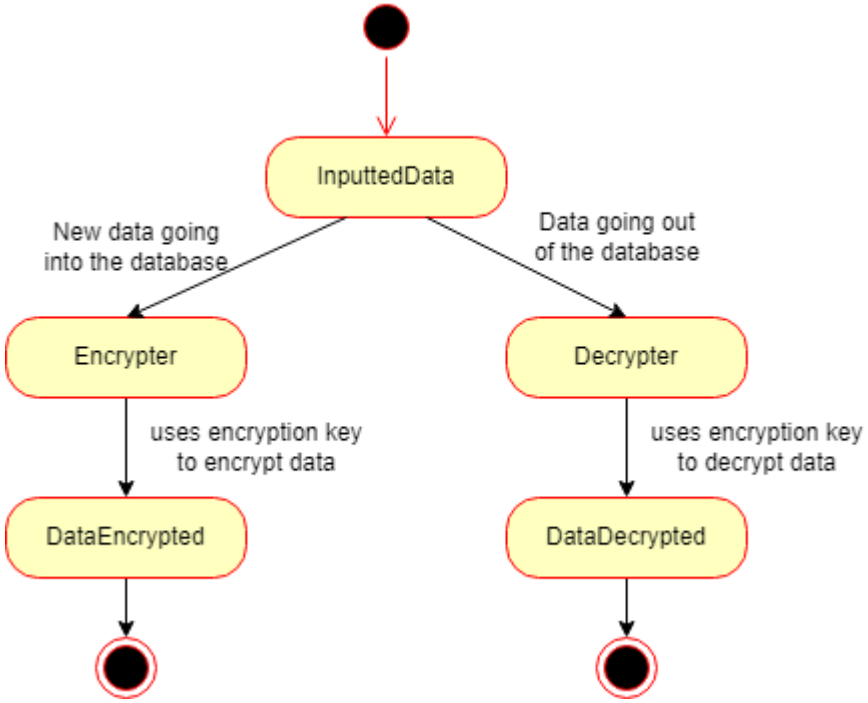
Profile management controller State Chart:



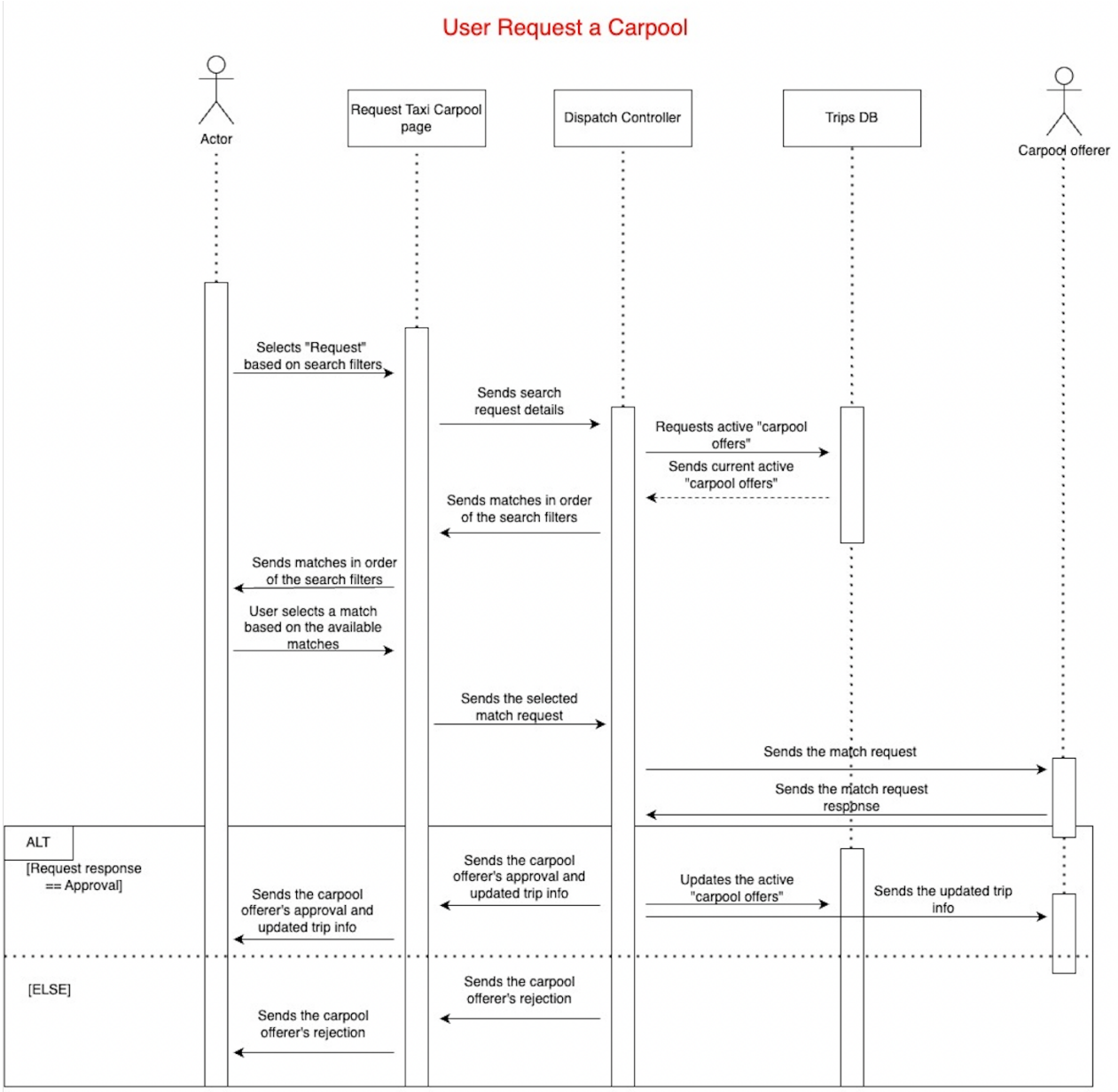
Session controller State Chart:

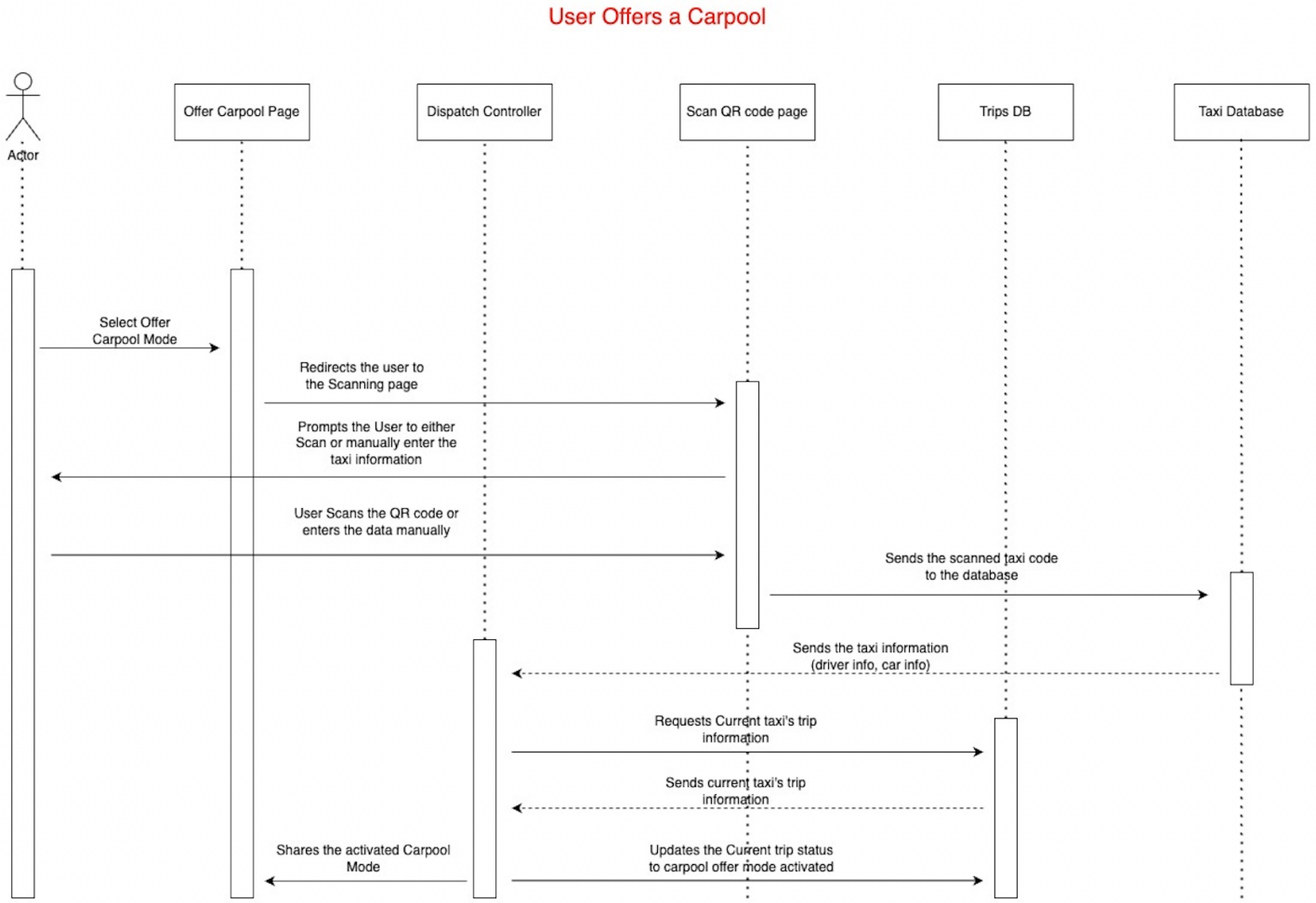


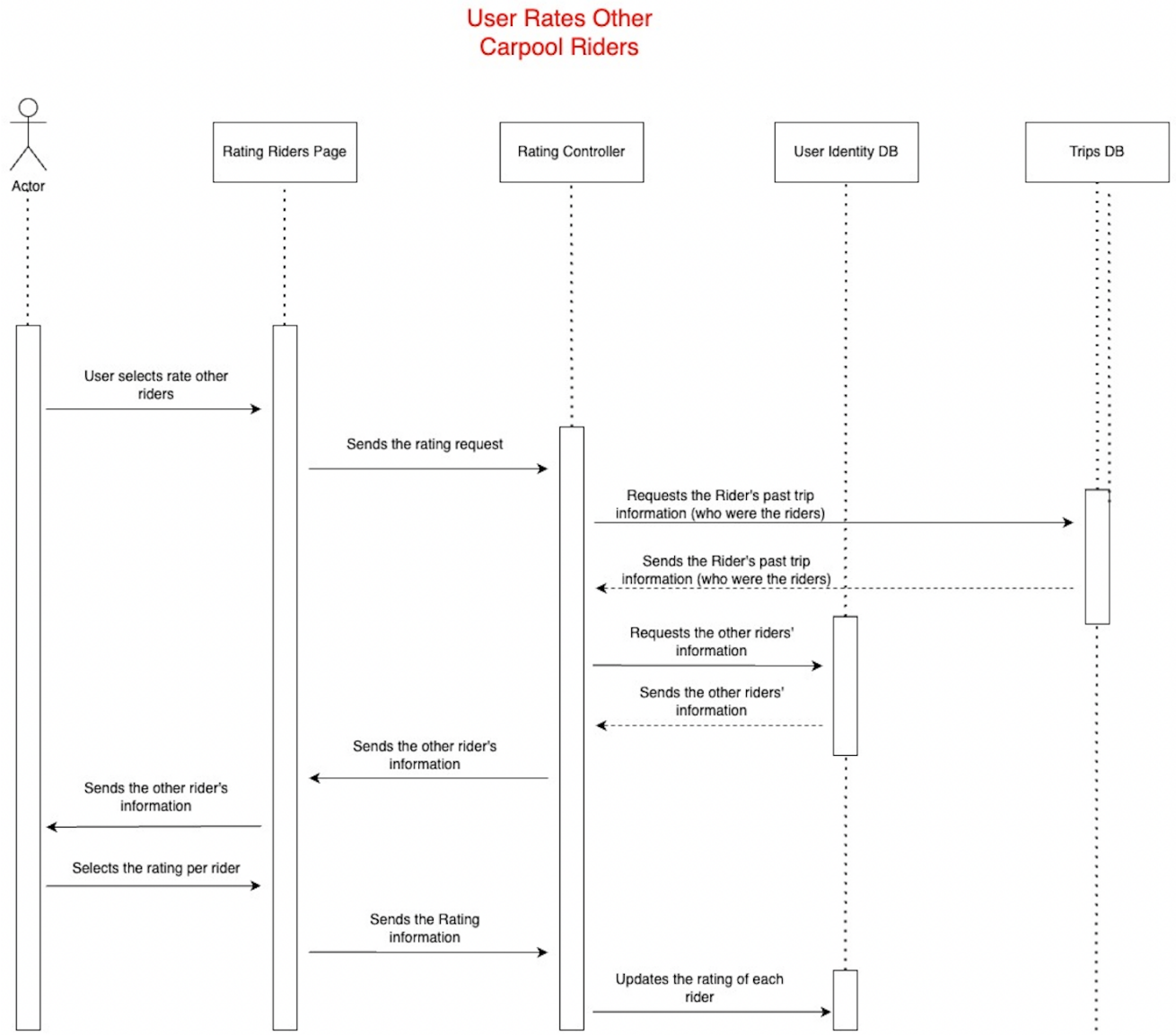
Encryption controller State Chart:

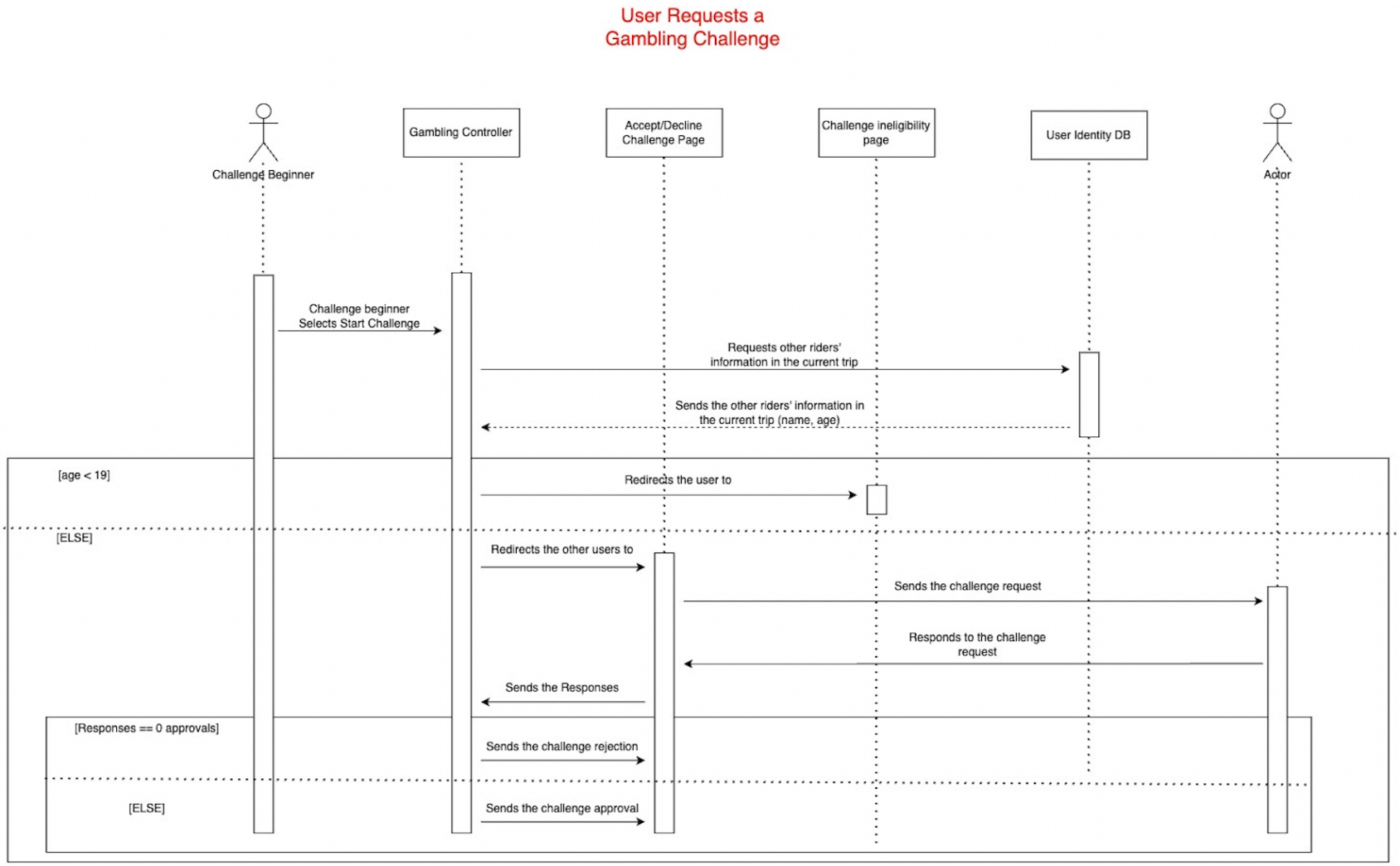


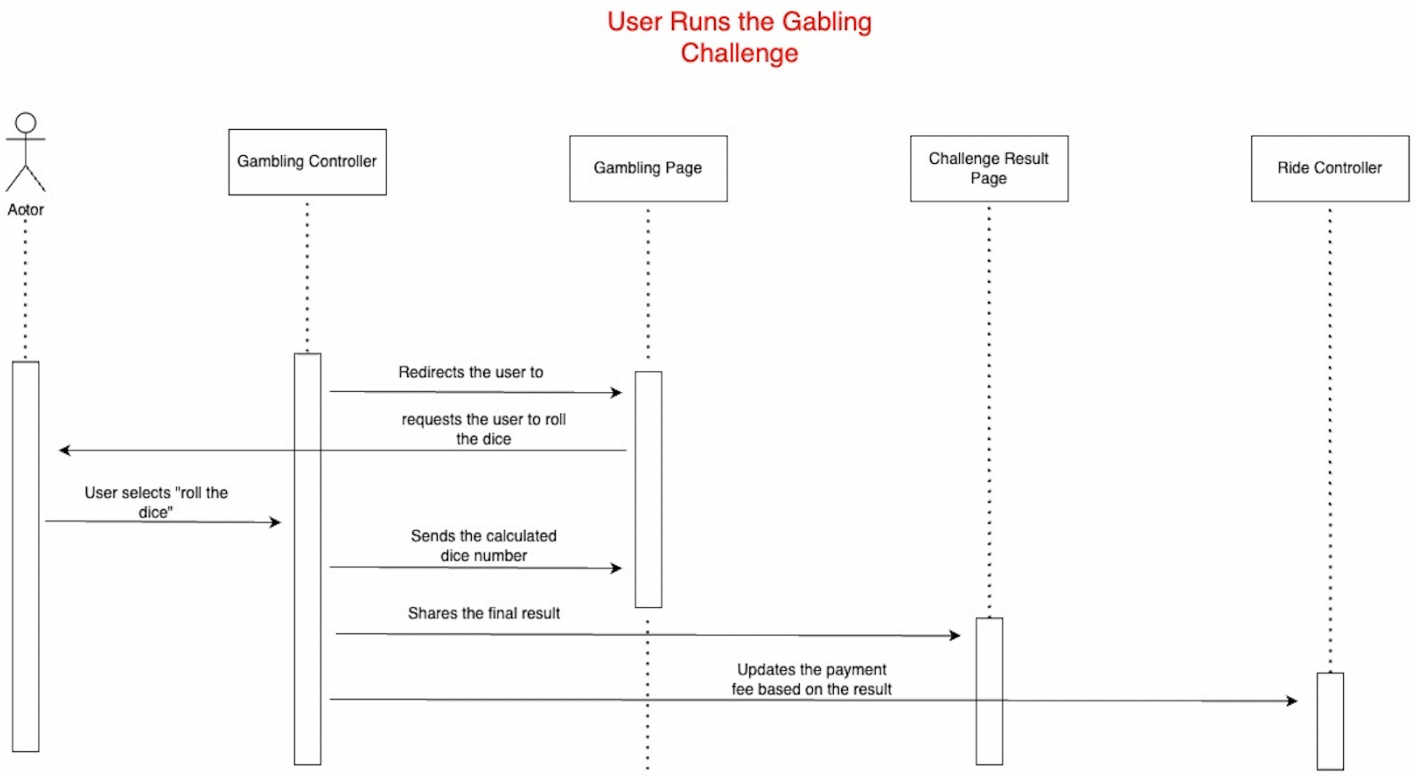
3 Sequence Diagrams

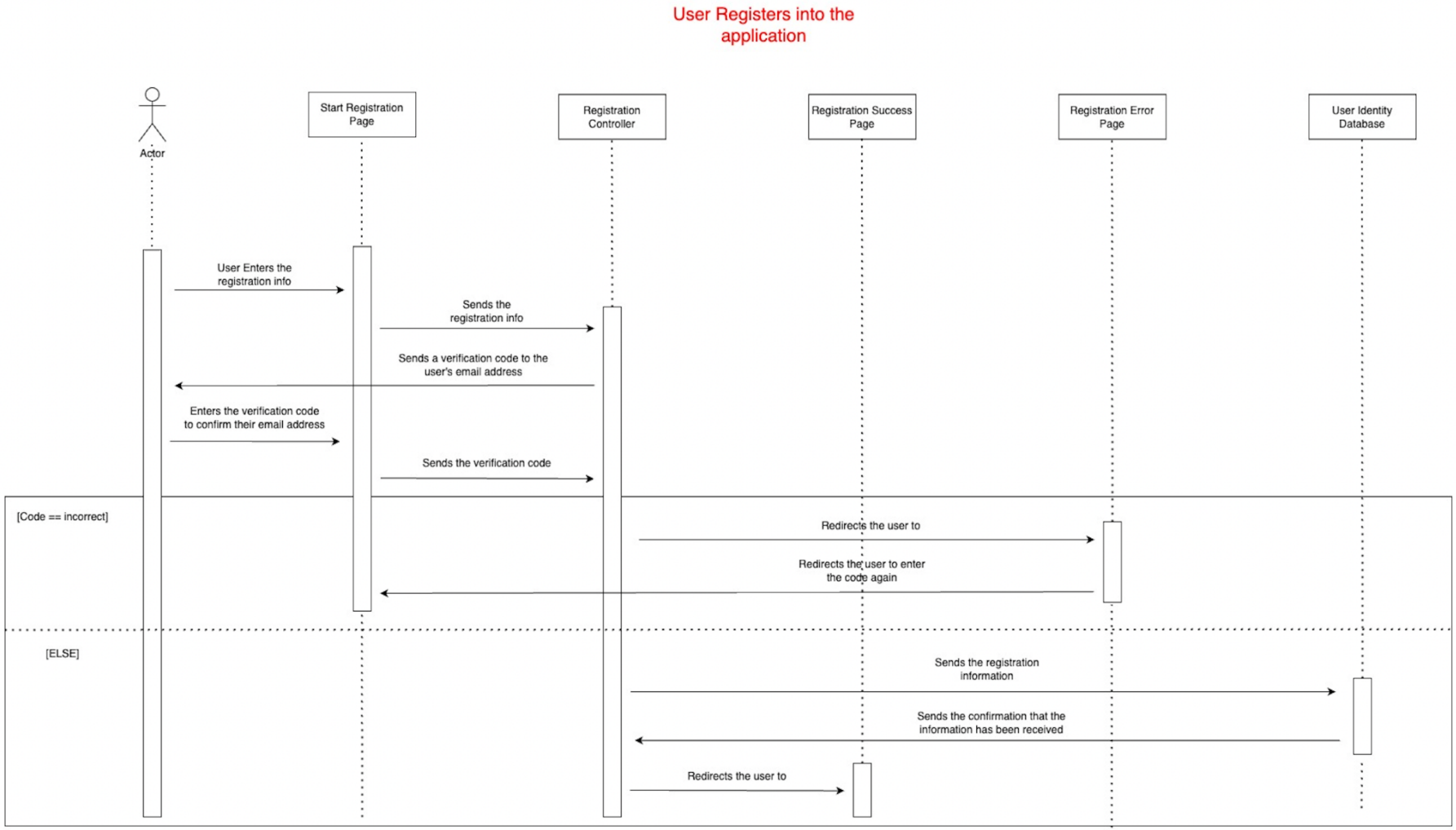

User Request a Carpool
Actor
Request Taxi Carpool page
Dispatch Controller
Trips DB
Carpool offerer
Selects "Request" based on search filters
Sends search request details
Requests active "carpool offers"
Sends current active "carpool offers"
Sends matches in order of the search filters
Sends matches in order of the search filters
User selects a match based on the available matches
Sends the selected match request
Sends the match request
Sends the match request response
ALT
[Request response == Approval]
Sends the carpool offerer's approval and updated trip info
Updates the active "carpool offers"
Sends the updated trip info
Sends the carpool offerer's approval and updated trip info
[ELSE]
Sends the carpool offerer's rejection
Sends the carpool offerer's rejection

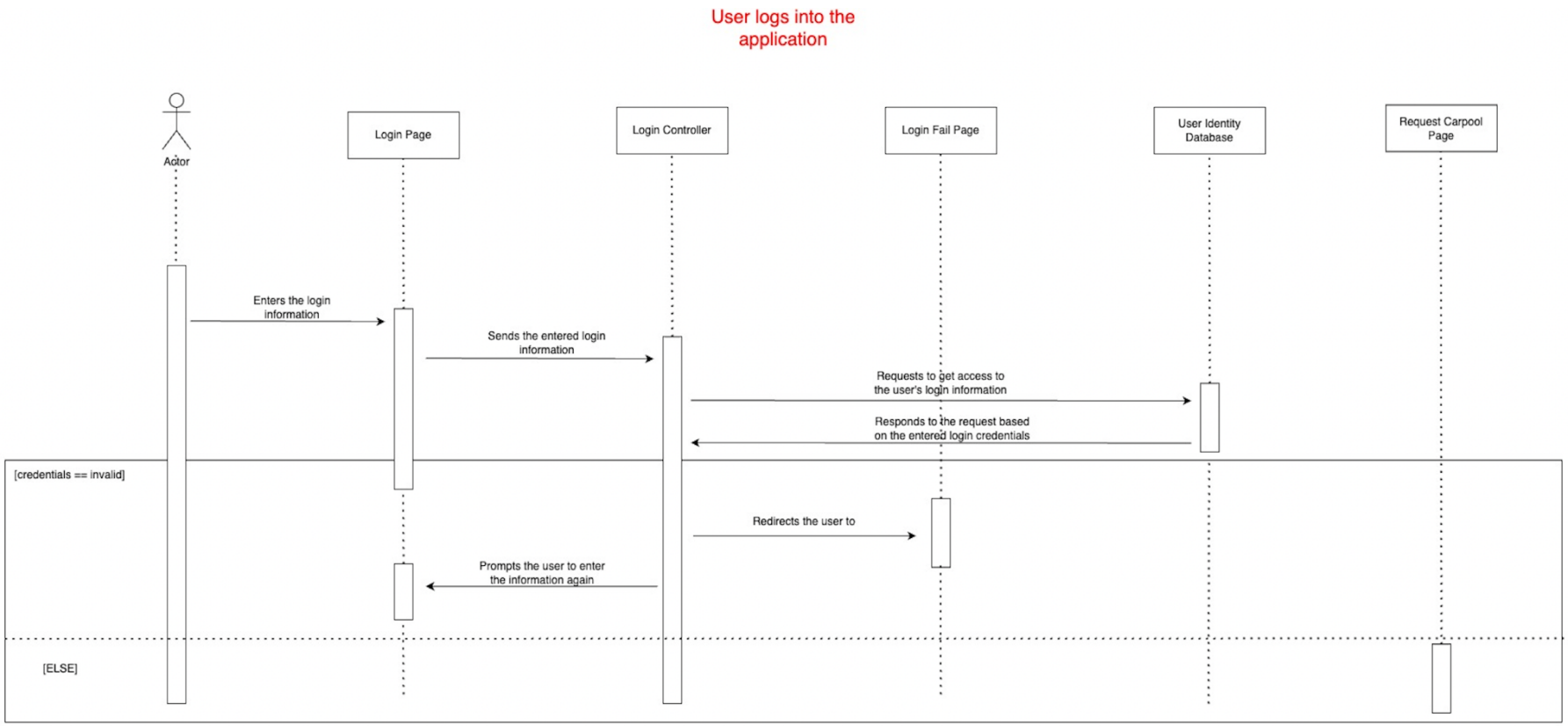

User Offers a Carpool
Actor
Offer Carpool Page
Dispatch Controller
Scan QR code page
Trips DB
Taxi Database
Select Offer Carpool Mode
Redirects the user to the Scanning page
Prompts the User to either Scan or manually enter the taxi information
User Scans the QR code or enters the data manually
Sends the scanned taxi code to the database
Sends the taxi information (driver info, car info)
Requests Current taxi's trip information
Sends current taxi's trip information
Shares the activated Carpool Mode
Updates the Current trip status to carpool offer mode activated

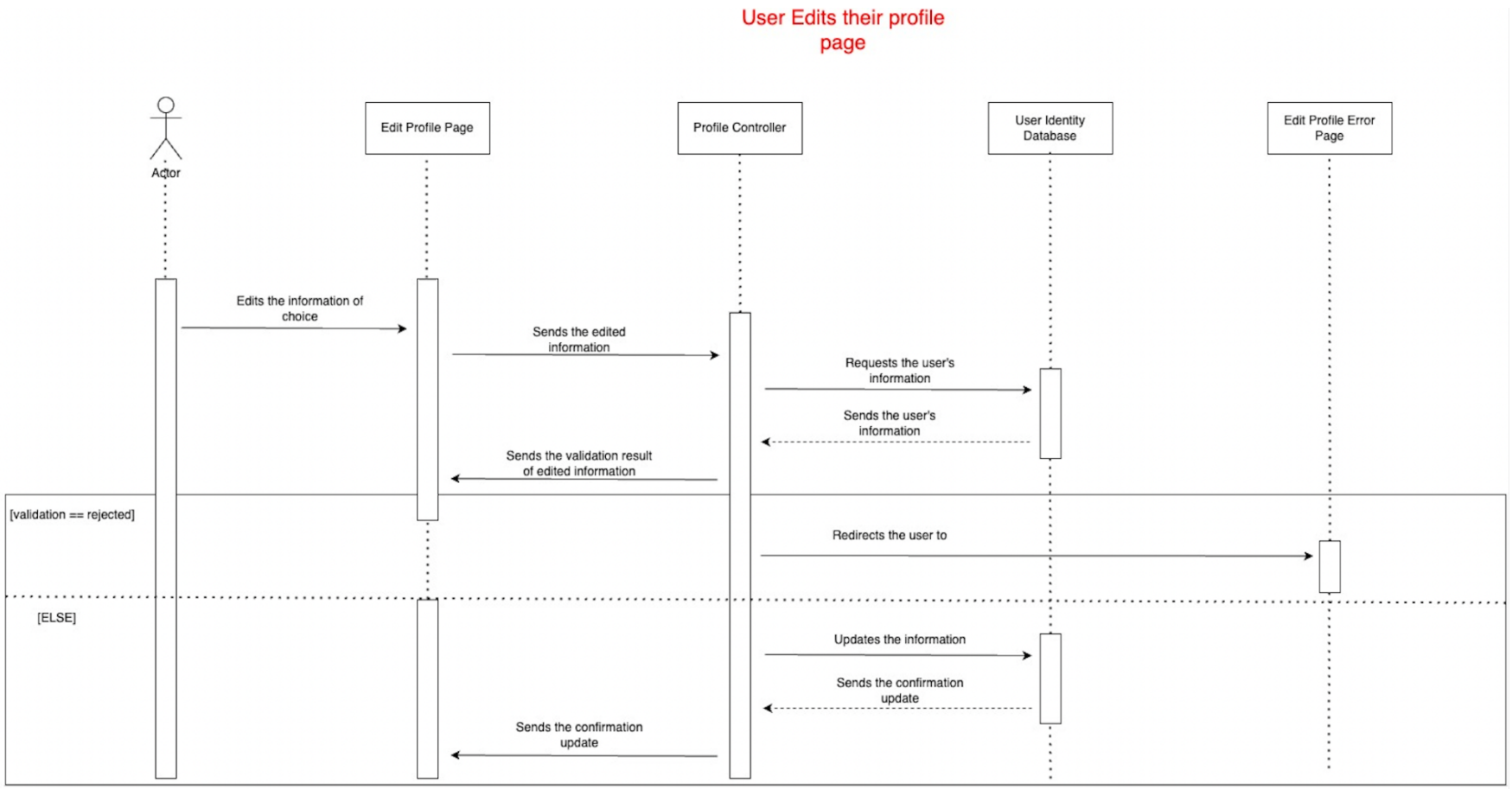

User Rates Other Carpool Riders
Actor
Rating Riders Page
Rating Controller
User Identity DB
Trips DB
User selects rate other riders
Sends the rating request
Requests the Rider's past trip information (who were the riders)
Sends the Rider's past trip information (who were the riders)
Requests the other riders' information
Sends the other riders' information
Sends the other rider's information
Sends the other rider's information
Selects the rating per rider
Sends the Rating information
Updates the rating of each rider


User Requests a
Gambling Challenge
Challenge Beginner
Gambling Controller
Accept/Decline
Challenge Page
Challenge ineligibility
page
User Identity DB
Actor
Challenge beginner
Selects Start Challenge
Requests other riders'
information in the current trip
Sends the other riders' information in
the current trip (name, age)
[age < 19]
Redirects the user to
[ELSE]
Redirects the other users to
Sends the challenge request
Responds to the challenge
request
Sends the Responses
[Responses == 0 approvals]
Sends the challenge rejection
[ELSE]
Sends the challenge approval


User Runs the Gabling Challenge
Aotor
Gambling Controller
Gambling Page
Challenge Result Page
Ride Controller
Redirects the user to
requests the user to roll the dice
User selects "roll the dice"
Sends the calculated dice number
Shares the final result
Updates the payment fee based on the result

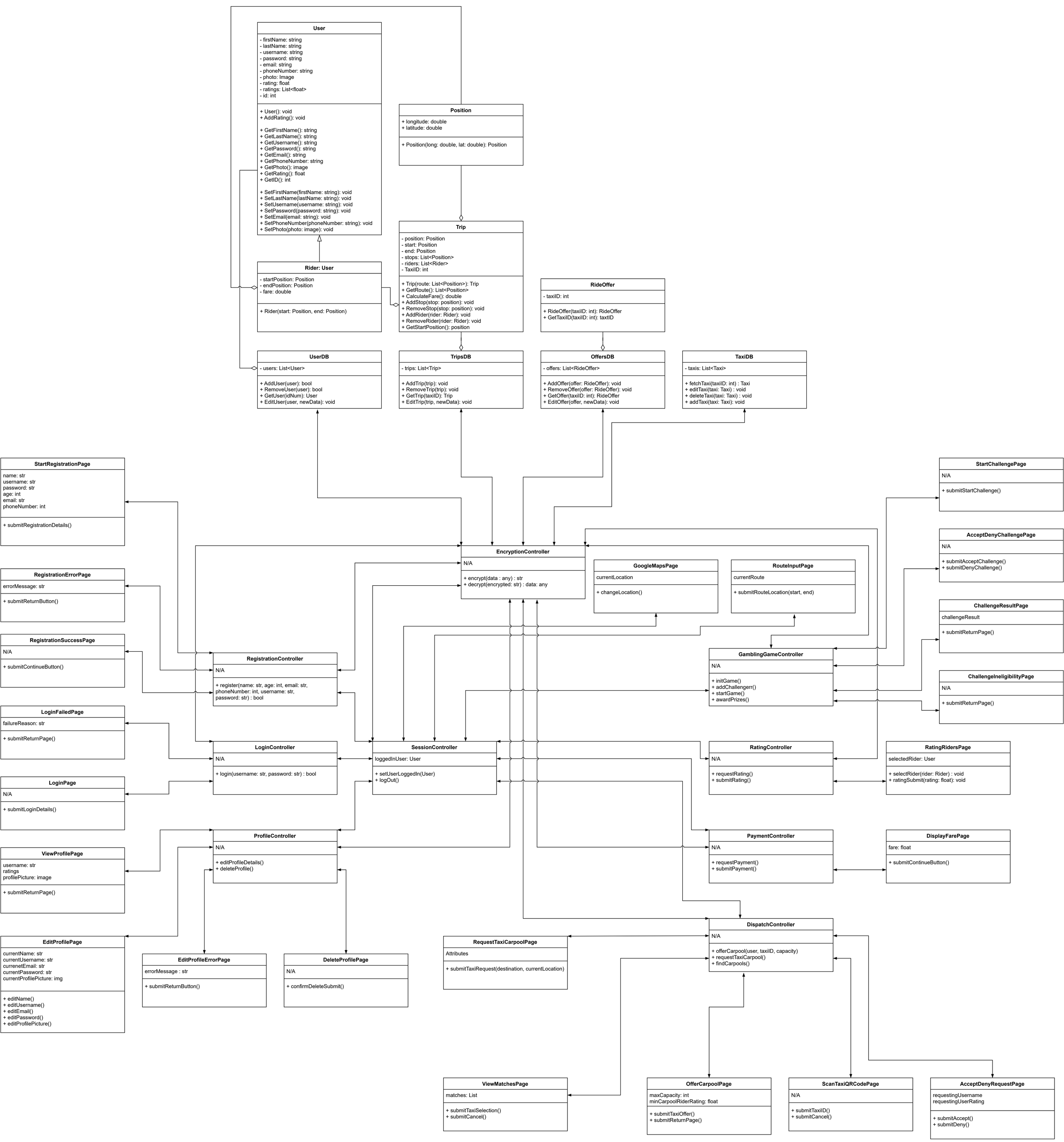

User Registers into the application
Actor
Start Registration Page
Registration Controller
Registration Success Page
Registration Error Page
User Identity Database
User Enters the registration info
Sends the registration info
Sends a verification code to the user's email address
Enters the verification code to confirm their email address
Sends the verification code
[Code == incorrect]
Redirects the user to
Redirects the user to enter the code again
[ELSE]
Sends the registration information
Sends the confirmation that the information has been received
Redirects the user to


User logs into the application
Actor
Login Page
Login Controller
Login Fail Page
User Identity Database
Request Carpool Page
Enters the login information
Sends the entered login information
Requests to get access to the user's login information
Responds to the request based on the entered login credentials
[credentials == invalid]
Redirects the user to
Prompts the user to enter the information again
[ELSE]


User Edits their profile page
Actor
Edit Profile Page
Profile Controller
User Identity Database
Edit Profile Error Page
Edits the information of choice
Sends the edited information
Requests the user's information
Sends the user's information
Sends the validation result of edited information
[validation == rejected]
Redirects the user to
[ELSE]
Updates the information
Sends the confirmation update
Sends the confirmation update

4 Detailed Class Diagram

(Full diagram, on the next page. Feel free to zoom in)


User
- firstName: string
- lastName: string
- username: string
- password: string
- email: string
- phoneNumber: string
- photo: Image
- rating: float
- ratings: List<float>
- id: int
+ User(): void
+ AddRating(): void
+ GetFirstName(): string
+ GetLastName(): string
+ GetUsername(): string
+ GetPassword(): string
+ GetEmail(): string
+ GetPhoneNumber: string
+ GetPhoto(): image
+ GetRating(): float
+ GetID(): int
+ SetFirstName(firstName: string): void
+ SetLastName(lastName: string): void
+ SetUsername(username: string): void
+ SetPassword(password: string): void
+ SetEmail(email: string): void
+ SetPhoneNumber(phoneNumber: string): void
+ SetPhoto(photo: image): void
Position
+ longitude: double
+ latitude: double
+ Position(long: double, lat: double): Position
Rider: User
- startPosition: Position
- endPosition: Position
- fare: double
+ Rider(start: Position, end: Position)
Trip
- position: Position
- start: Position
- end: Position
- stops: List<Position>
- riders: List<Rider>
- TaxiID: int
+ Trip(route: List<Position>): Trip
+ GetRoute(): List<Position>
+ CalculateFare(): double
+ AddStop(stop: position): void
+ RemoveStop(stop: position): void
+ AddRider(rider: Rider): void
+ RemoveRider(rider: Rider): void
+ GetStartPosition(): position
RideOffer
- taxiID: int
+ RideOffer(taxiID: int): RideOffer
+ GetTaxiID(taxiID: int): taxtID
UserDB
- users: List<User>
+ AddUser(user): bool
+ RemoveUser(user): bool
+ GetUser(idNum): User
+ EditUser(user, newData): void
TripsDB
- trips: List<Trip>
+ AddTrip(trip): void
+ RemoveTrip(trip): void
+ GetTrip(taxiID): Trip
+ EditTrip(trip, newData): void
OffersDB
- offers: List<RideOffer>
+ AddOffer(offer: RideOffer): void
+ RemoveOffer(offer: RideOffer): void
+ GetOffer(taxiID: int): RideOffer
+ EditOffer(offer, newData): void
TaxiDB
- taxis: List<Taxi>
+ fetchTaxi(taxiID: int) : Taxi
+ editTaxi(taxi: Taxi) : void
+ deleteTaxi(taxi: Taxi) : void
+ addTaxi(taxi: Taxi): void
StartRegistrationPage
name: str
username: str
password: str
age: int
email: str
phoneNumber: int
+ submitRegistrationDetails()
EncryptionController
N/A
+ encrypt(data : any) : str
+ decrypt(encrypted: str) : data: any
GoogleMapsPage
currentLocation
+ changeLocation()
RouteInputPage
currentRoute
+ submitRouteLocation(start, end)
StartChallengePage
N/A
+ submitStartChallenge()
AcceptDenyChallengePage
N/A
+ submitAcceptChallenge()
+ submitDenyChallenge()
RegistrationErrorPage
errorMessage: str
+ submitReturnButton()
RegistrationSuccessPage
N/A
+ submitContinueButton()
RegistrationController
N/A
+ register(name: str, age: int, email: str, phoneNumber: int, username: str, password: str) : bool
GamblingGameController
N/A
+ initGame()
+ addChallengerr()
+ startGame()
+ awardPrizes()
ChallengeResultPage
challengeResult
+ submitReturnPage()
ChallengeIneligibilityPage
N/A
+ submitReturnPage()
LoginFailedPage
failureReason: str
+ submitReturnPage()
LoginController
N/A
+ login(username: str, password: str) : bool
SessionController
loggedInUser: User
+ setUserLoggedIn(User)
+ logOut()
RatingController
N/A
+ requestRating()
+ submitRating()
RatingRidersPage
selectedRider: User
+ selectRider(rider: Rider) : void
+ ratingSubmit(rating: float): void
LoginPage
N/A
+ submitLoginDetails()
ProfileController
N/A
+ editProfileDetails()
+ deleteProfile()
PaymentController
N/A
+ requestPayment()
+ submitPayment()
DisplayFarePage
fare: float
+ submitContinueButton()
ViewProfilePage
username: str
ratings
profilePicture: image
+ submitReturnPage()
DispatchController
N/A
+ offerCarpool(user, taxiID, capacity)
+ requestTaxiCarpool()
+ findCarpools()
EditProfilePage
currentName: str
currentUsername: str
currenetEmail: str
currentPassword: str
currentProfilePicture: img
+ editName()
+ editUsername()
+ editEmail()
+ editPassword()
+ editProfilePicture()
EditProfileErrorPage
errorMessage : str
+ submitReturnButton()
DeleteProfilePage
N/A
+ confirmDeleteSubmit()
RequestTaxiCarpoolPage
Attributes
+ submitTaxiRequest(destination, currentLocation)
ViewMatchesPage
matches: List
+ submitTaxiSelection()
+ submitCancel()
OfferCarpoolPage
maxCapacity: int
minCarpoolRiderRating: float
+ submitTaxiOffer()
+ submitReturnPage()
ScanTaxiQRCodePage
N/A
+ submitTaxiID()
+ submitCancel()
AcceptDenyRequestPage
requestingUsername
requestingUserRating
+ submitAccept()
+ submitDeny()

A Division of Labour

Waleed: Completed all the state charts, all of section 2

Signature:

Willie: Completed detailed class diagrams (Section 4), for controllers, pages, databases, and extra types

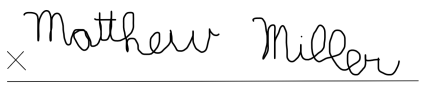
Signature:

Daniel: Contributed to detailed class diagram and section 1

Signature:

Matt: Contributed to detailed class diagram

Signature:



Arash: Designed all the sequence diagrams in section 3 with respect to the updated class diagram

Signature: