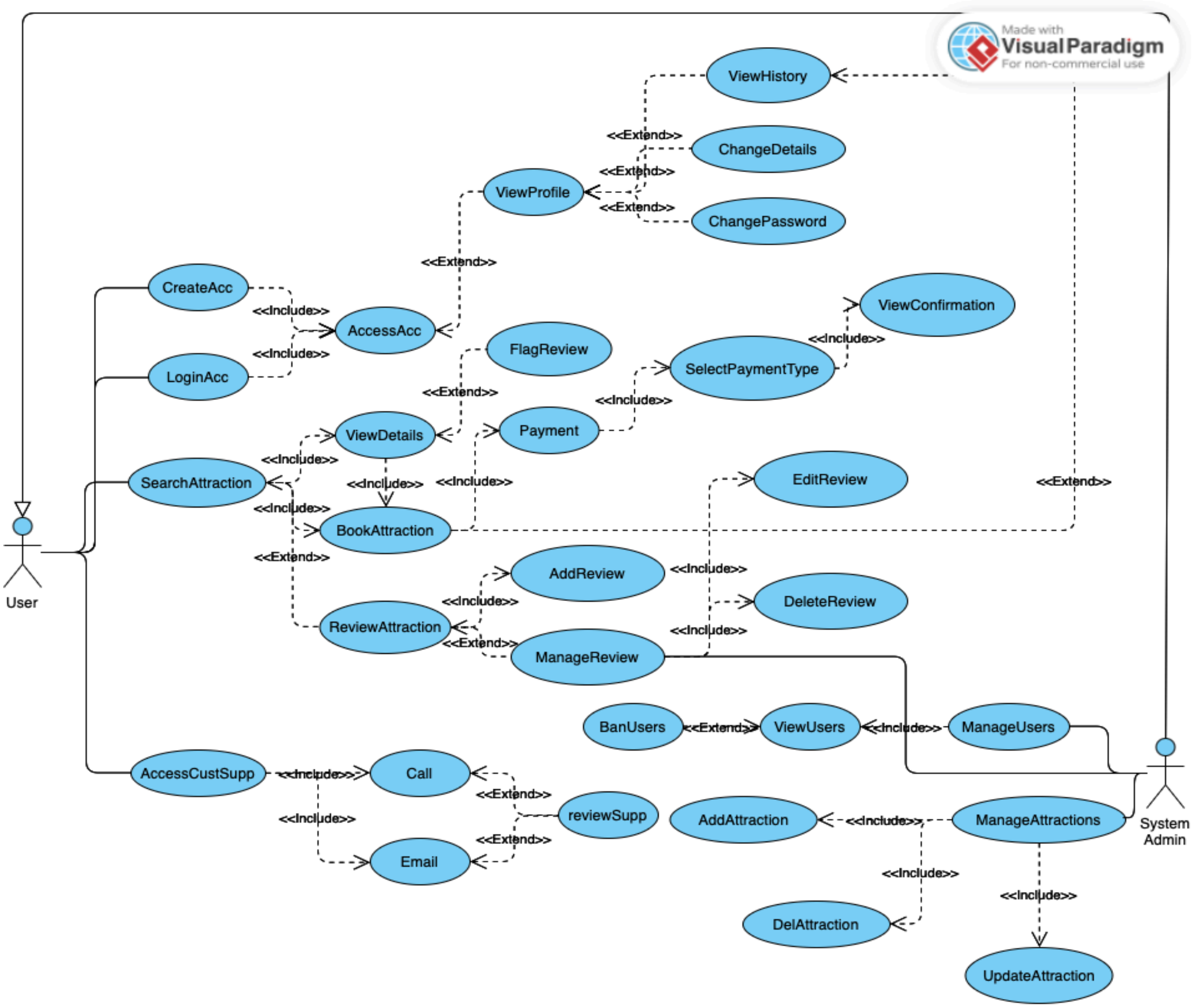
NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

# SC2006 - Software Engineering
# LAB 3 Deliverable

| | |
|---|---|
| **Lab Group** | FDAC |
| **Team** | SingScape |
| **Members** | Dhaded Aditya Mahalingeshwar |
| | Darren Jong Jet Ren |
| | Jacob Tong Wai Hong |
| | Savanur Akash |
| | Xu Junpeng |

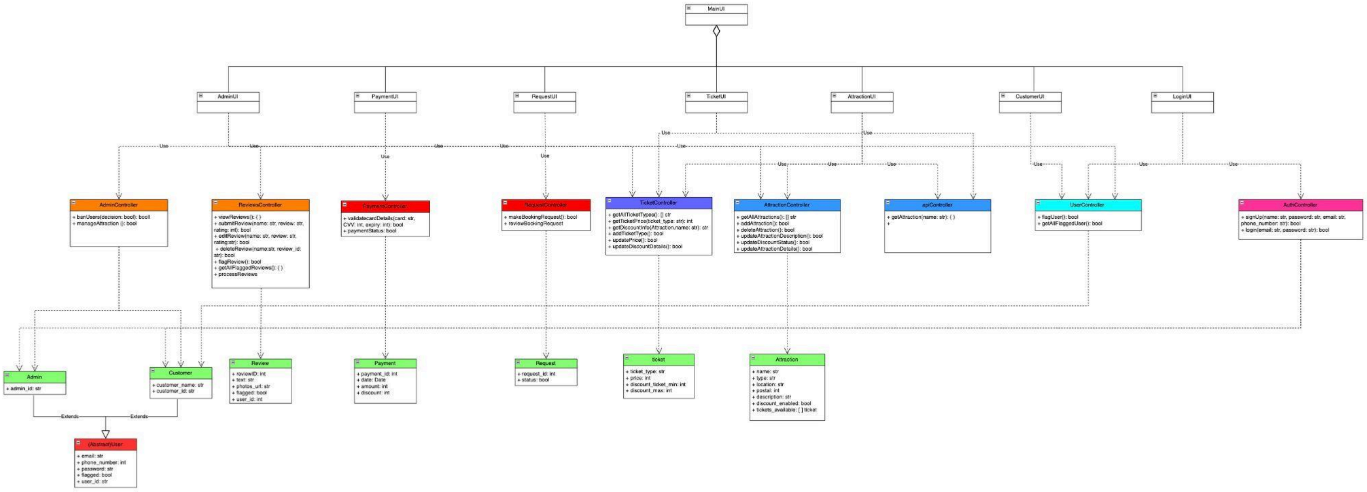
# Table of Contents

## I. Use Case Model:



# II. Design Model

## a. Key Classes Diagram



### User Interfaces (UI):

- **UserUI:** This is the main UI for users. It involves Singscape's common application features that are accessible to all users.

- **AdminUI:** Pages specific to admins. eg: Attraction Management Screen that allows admins to add, update or delete attraction listed on the website.

- **LoginUI:** Authentication pages that include Login and Signup features.

### Controllers (Facade Pattern):

Application of the Facade Pattern is when the controllers serve as "interfaces" for interacting with the application's business logic, acting as entry points to the application. There are multiple controllers, depending on the specific use case and/or role of the User.

- **Auth Controller:** This is an Authentication Controller. It lets Users login and sign up. It has access to User entity.
- **Admin Controller:** This controller specifies the accessibility of different endpoints to users of different roles (Admin/Non-Admin). It has access to the User entity.

- **Attraction Controller:** This is the controller for attractions. It allows viewing and modification of attraction listings. It has access to the Attraction entity.
- **User Controller:** This is the controller that controls creation, modification and deletion of user profiles. It has access to the User entity
- **Payment Controller:** This is the controller for Payments. It allows users to make payments. It has access to the Payment entity.
- **Request Controller:** This controller handles booking requests prior to confirmation. It has access to the Request entity.
- **Booking Controller:** This is the controller for bookings. It allows users to make and view bookings. It has access to the Booking entity.
- **Review Controller:** This is the controller for reviews. It allows users to submit reviews and access submitted reviews.
- **Ticket Controller:** This is the controller for tickets. It handles generation of tickets upon successful booking. It has access to the Ticket entity.

## Data Access Layer

To enable efficient data retrieval and data manipulation at the cost of scalability, the system utilises an optimally designed Data Access Layer that hides database activity. Utilising the Factory Pattern and the Strategy Pattern adds further to the modularity and flexibility and extensibility of the architecture and therefore enables the addition of new data types without modification to the fundamental logic.

### Factory Pattern

The Factory Pattern provides a framework to defer the process of creating objects so that data-related objects can be dynamically instantiated at run time. Instead of explicitly instantiating data-access objects, a factory class would take charge and produce the requisite object required (for example, User, Booking, or Payment). This approach allows components to be separated from one another and with the creation process from the business logic involved.

### Strategy Pattern

The Strategy Pattern allows different data-access methods to be chosen at runtime and hence adds flexibility to the system with regard to changing storage and retrieval requirements. Instead of hard coding CRUD (Create, Read, Update, Delete) operations in every entity, a data access method interface is provided with dedicated implementations controlling operations based on dedicated entities. This allows evolution without requiring changes to the existing code because the inclusion of a new data management strategy becomes very simple.

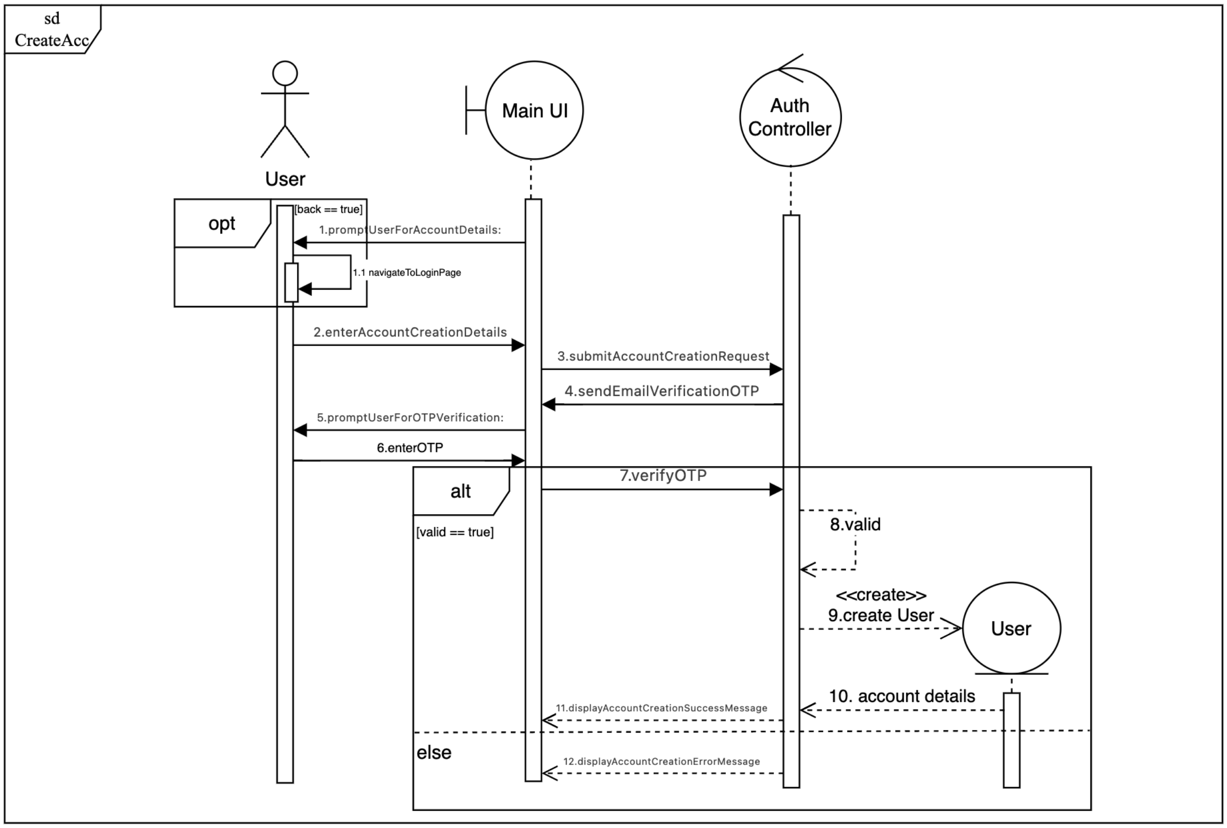
### Major Benefits

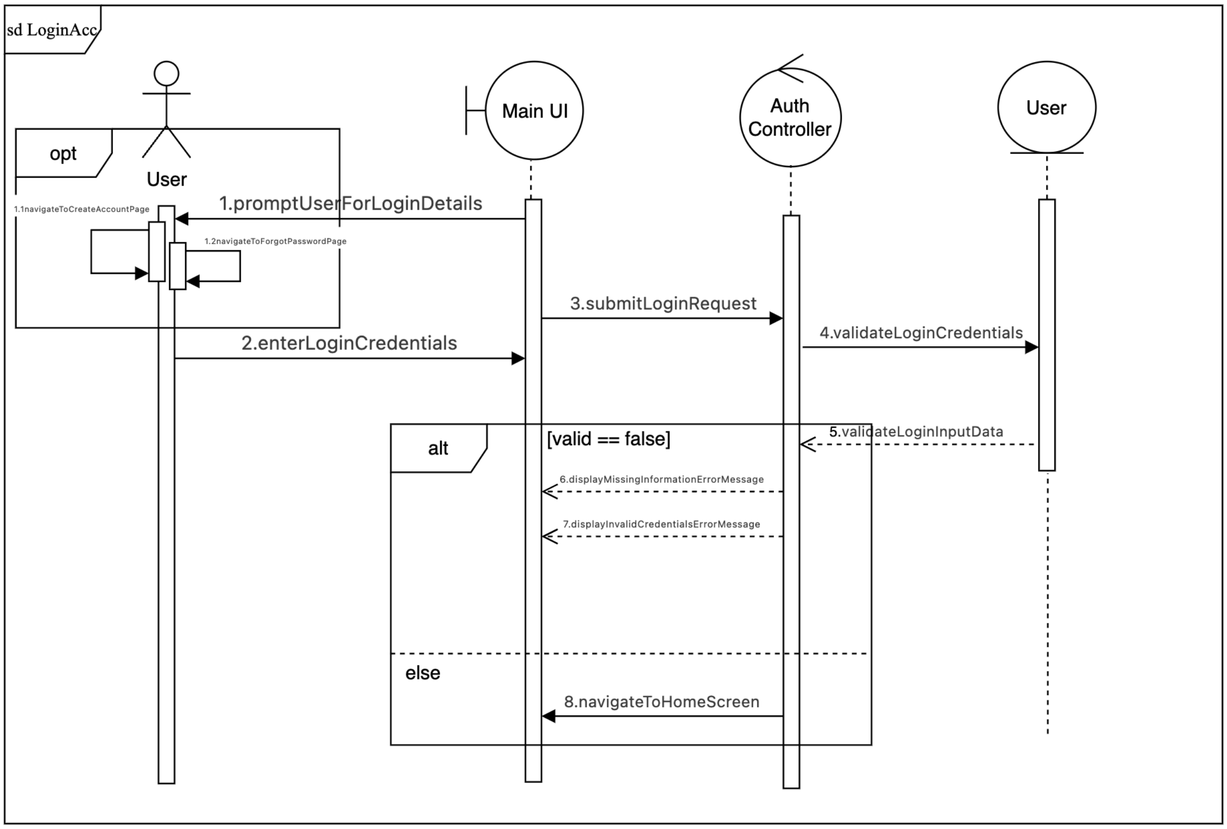
- Scalability – The system can handle increasing data operations without compromising performance.
- Flexibility – New data access methods or storage mechanisms can be added without modifying core logic.

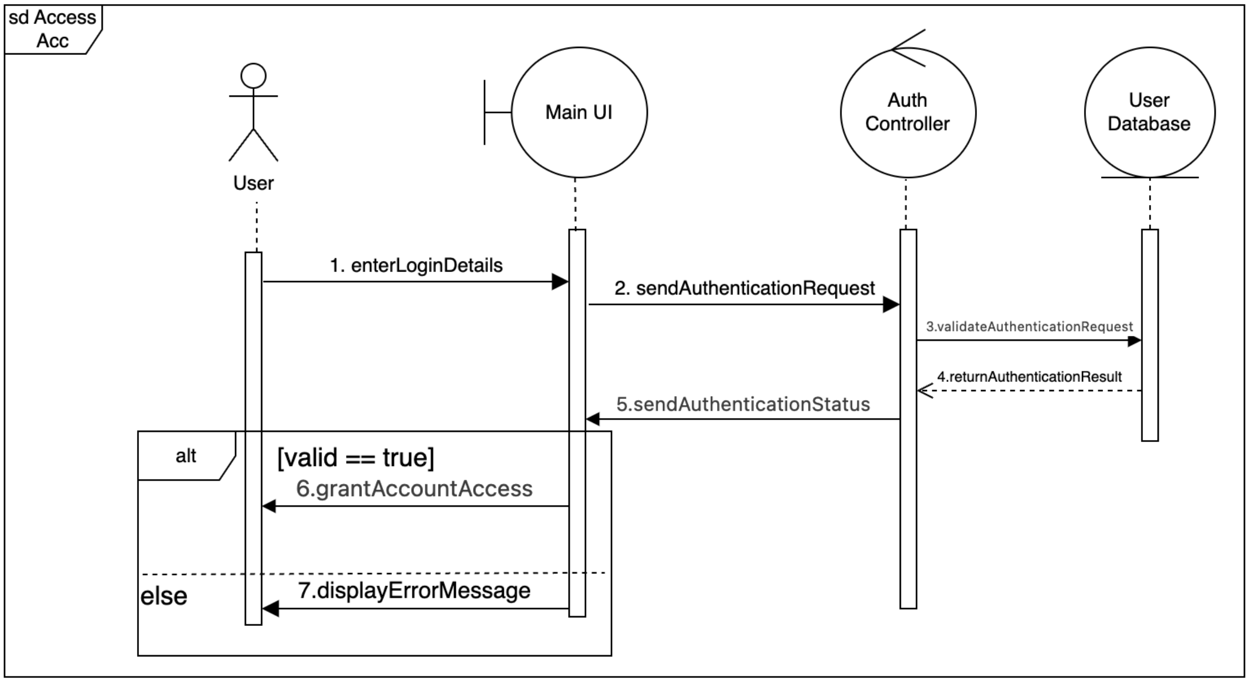
- Maintainability - Isolates business logic from database management and hence clear and well-structured code.

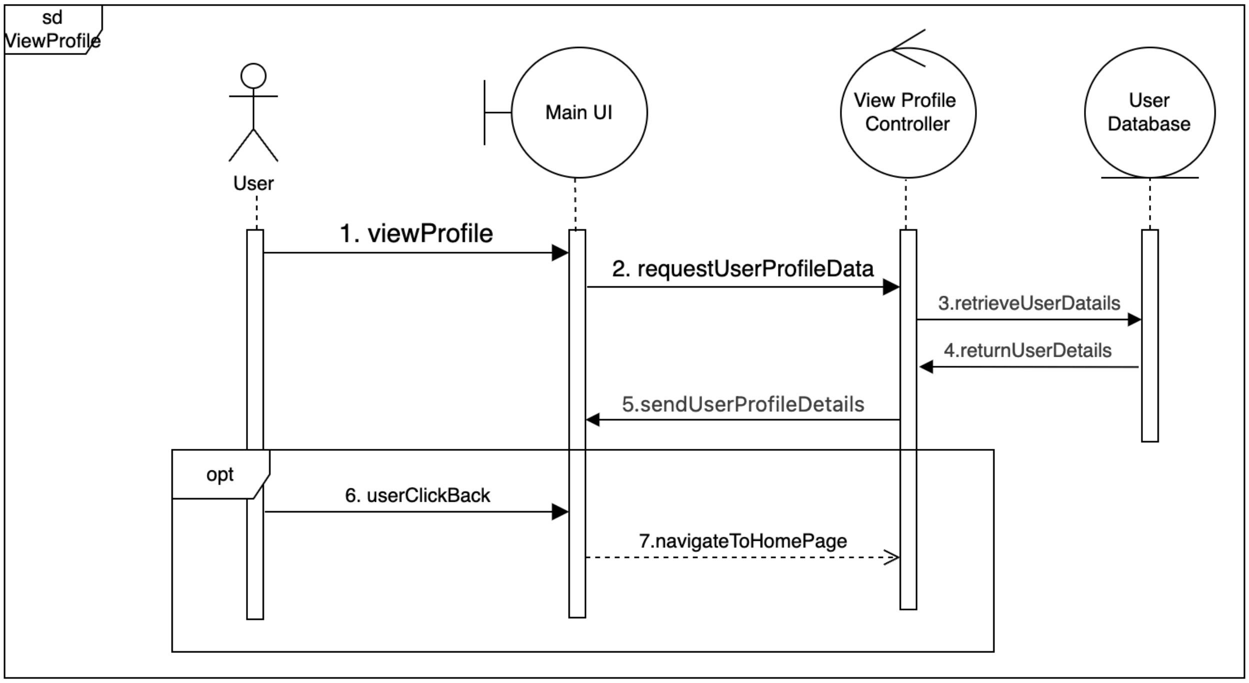
By utilizing such design patterns, the Data Access Layer provides a solid foundation that ensures effective and adaptable data management, hence enhancing consistency and scalability in the application.

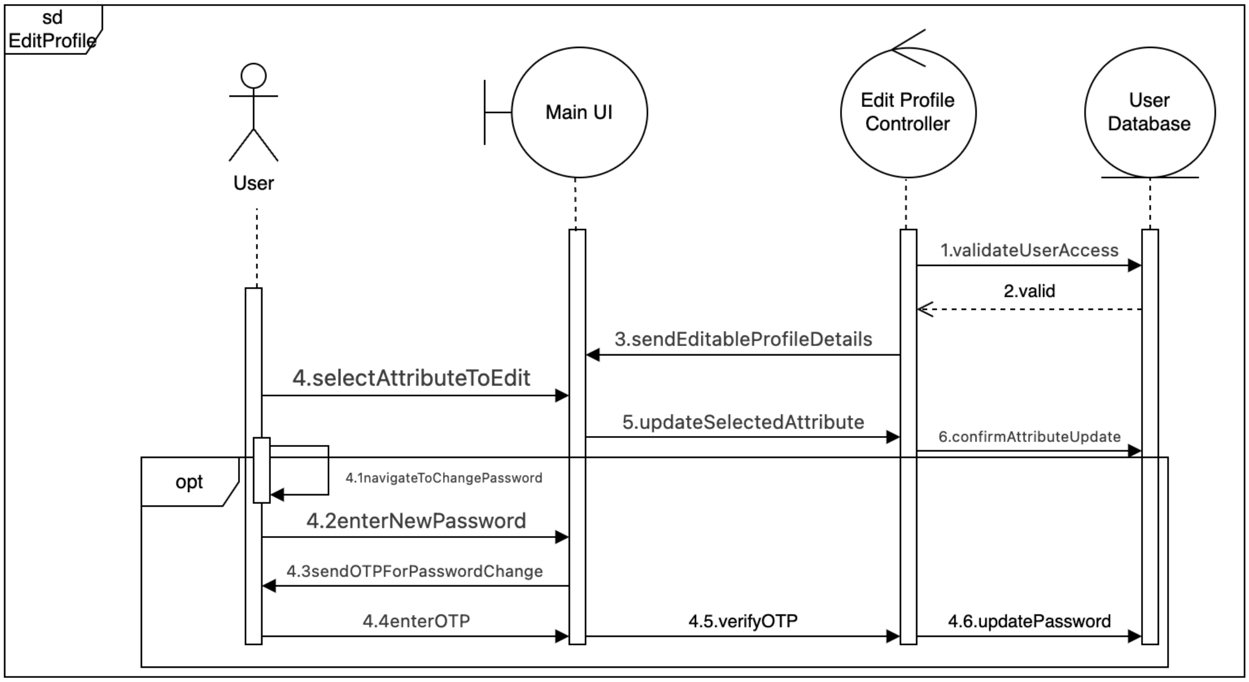
## b. Sequence Diagrams of Use Cases:

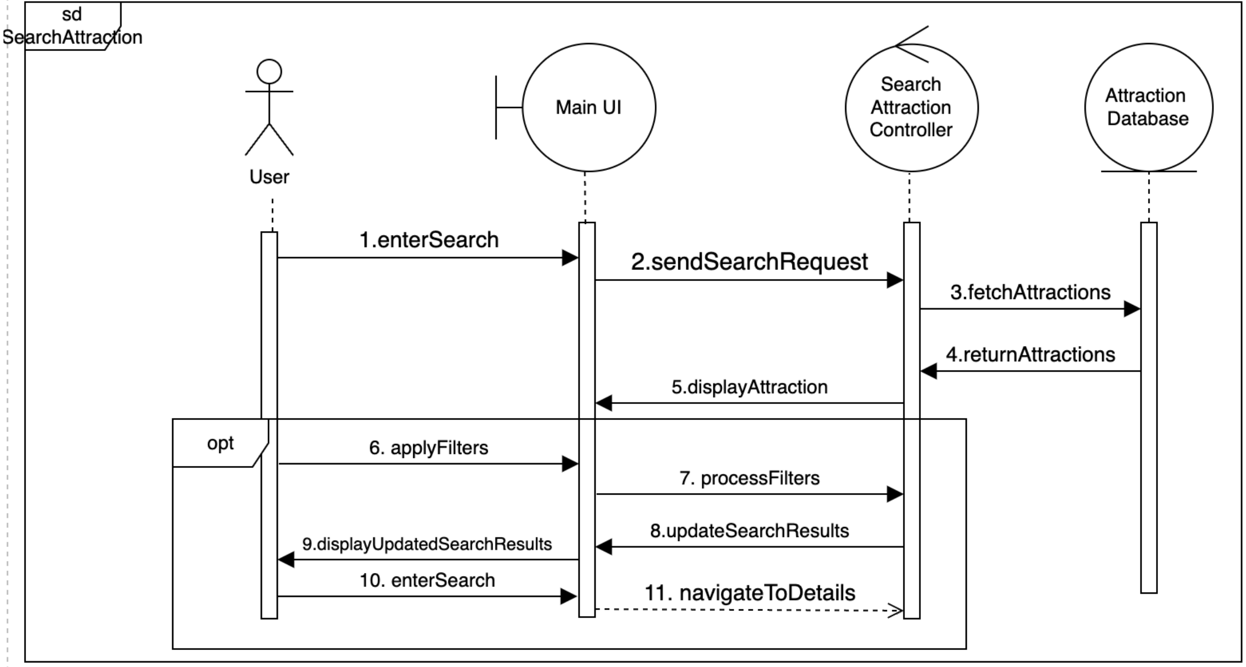

sd
CreateAcc
User
Main UI
Auth
Controller
opt
[back == true]
1.promptUserForAccountDetails:
1.1 navigateToLoginPage
2.enterAccountCreationDetails
3.submitAccountCreationRequest
4.sendEmailVerificationOTP
5.promptUserForOTPVerification:
6.enterOTP
alt
7.verifyOTP
[valid == true]
8.valid
<<create>>
9.create User
User
10. account details
11.displayAccountCreationSuccessMessage
else
12.displayAccountCreationErrorMessage

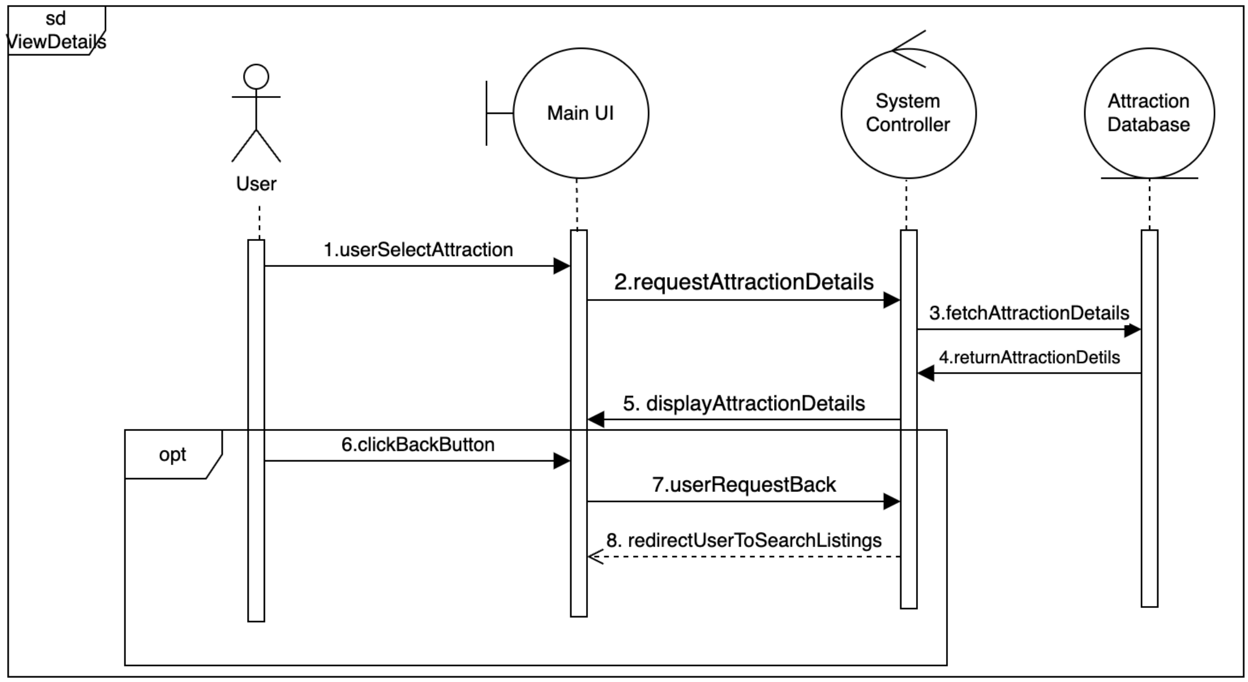

sd LoginAcc
User
Main UI
Auth Controller
User
opt
1.promptUserForLoginDetails
1.1navigateToCreateAccountPage
1.2navigateToForgotPasswordPage
2.enterLoginCredentials
3.submitLoginRequest
4.validateLoginCredentials
alt
[valid == false]
5.validateLoginInputData
6.displayMissingInformationErrorMessage
7.displayInvalidCredentialsErrorMessage
else
8.navigateToHomeScreen

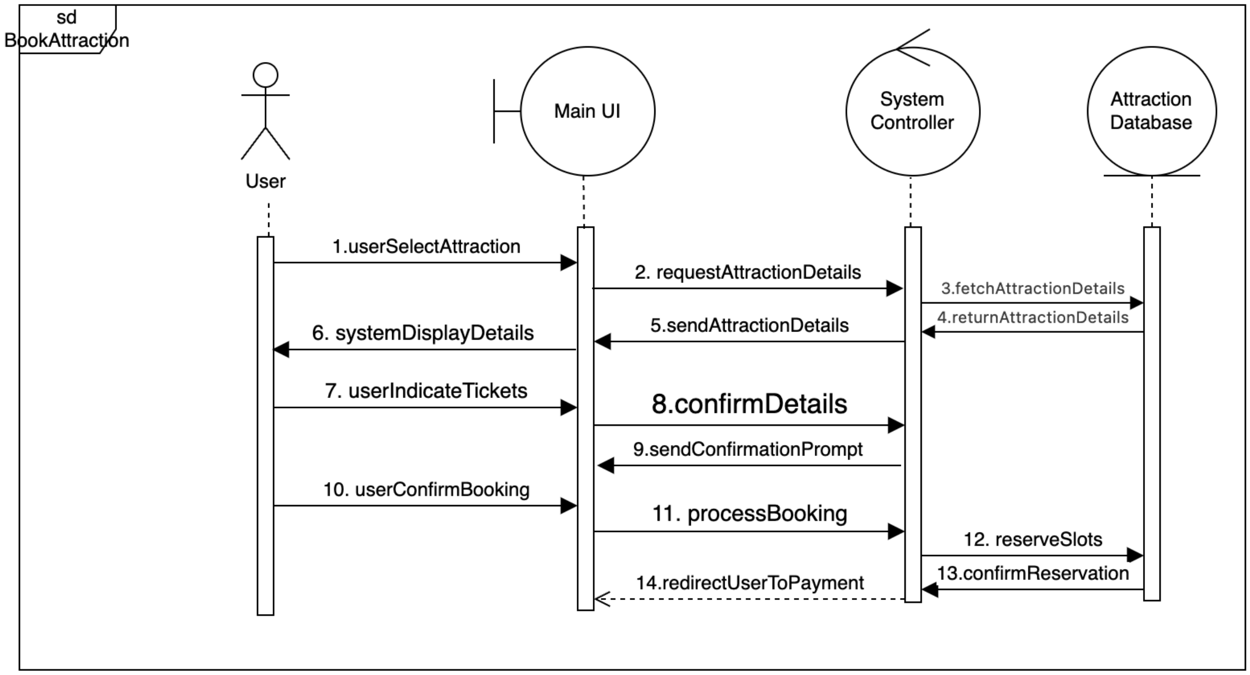

sd Access Acc
User
Main UI
Auth Controller
User Database
1. enterLoginDetails
2. sendAuthenticationRequest
3.validateAuthenticationRequest
4.returnAuthenticationResult
5.sendAuthenticationStatus
alt
[valid == true]
6.grantAccountAccess
else
7.displayErrorMessage

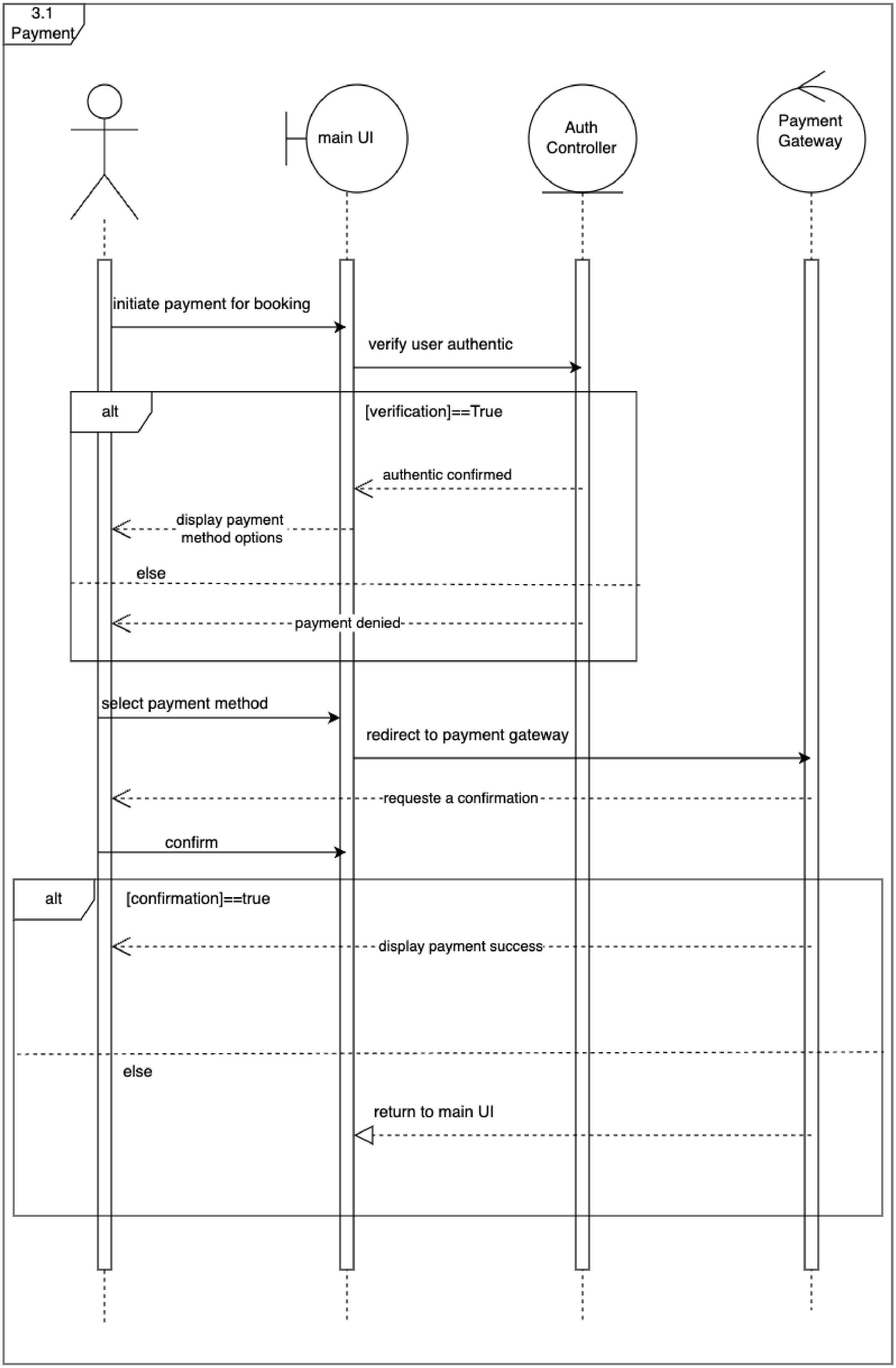

sd ViewProfile
User
Main UI
View Profile Controller
User Database
1. viewProfile
2. requestUserProfileData
3.retrieveUserDatails
4.returnUserDetails
5.sendUserProfileDetails
opt
6. userClickBack
7.navigateToHomePage

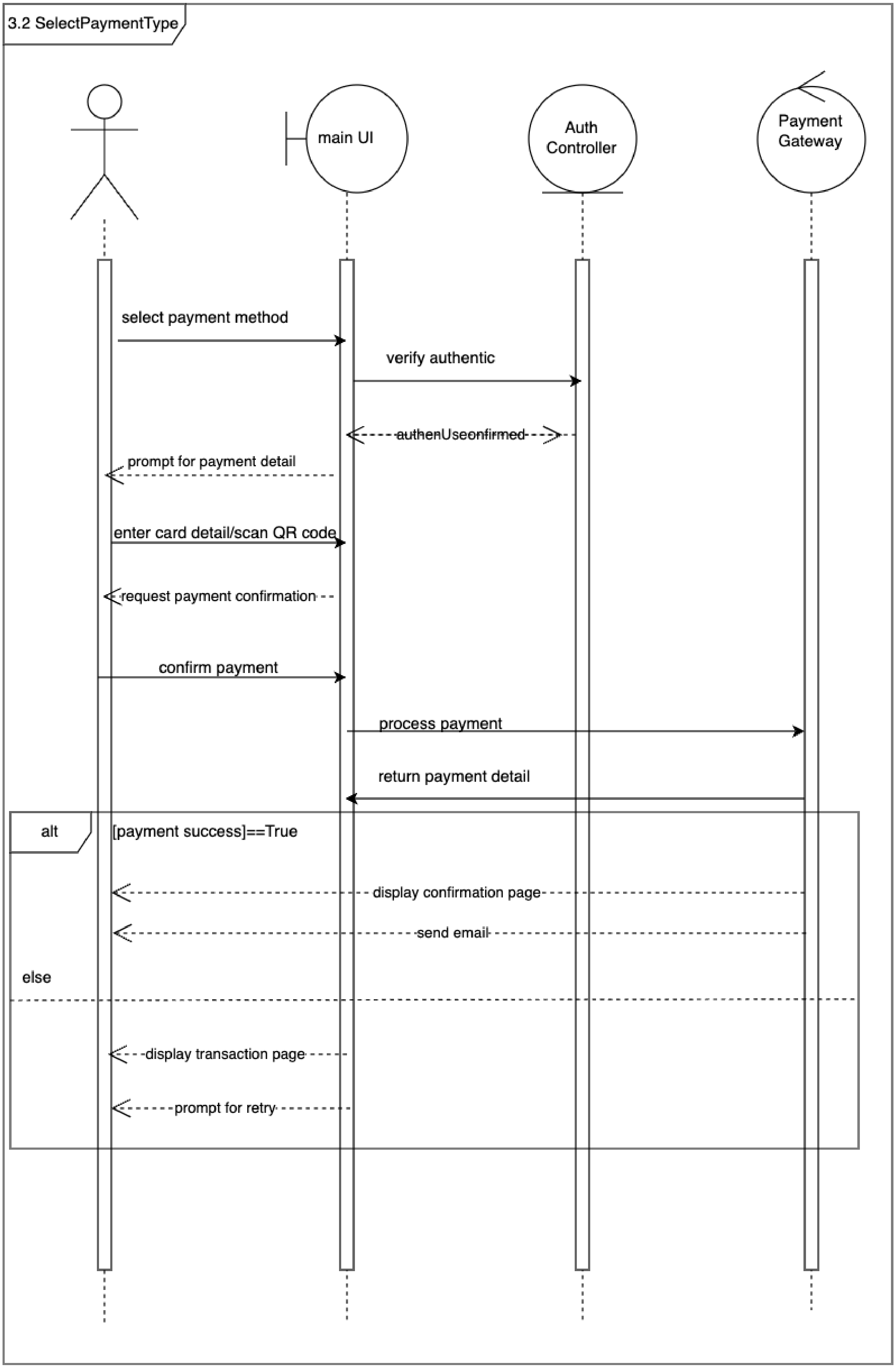

sd
EditProfile
User
Main UI
Edit Profile Controller
User Database
1.validateUserAccess
2.valid
3.sendEditableProfileDetails
4.selectAttributeToEdit
5.updateSelectedAttribute
6.confirmAttributeUpdate
opt
4.1navigateToChangePassword
4.2enterNewPassword
4.3sendOTPForPasswordChange
4.4enterOTP
4.5.verifyOTP
4.6.updatePassword

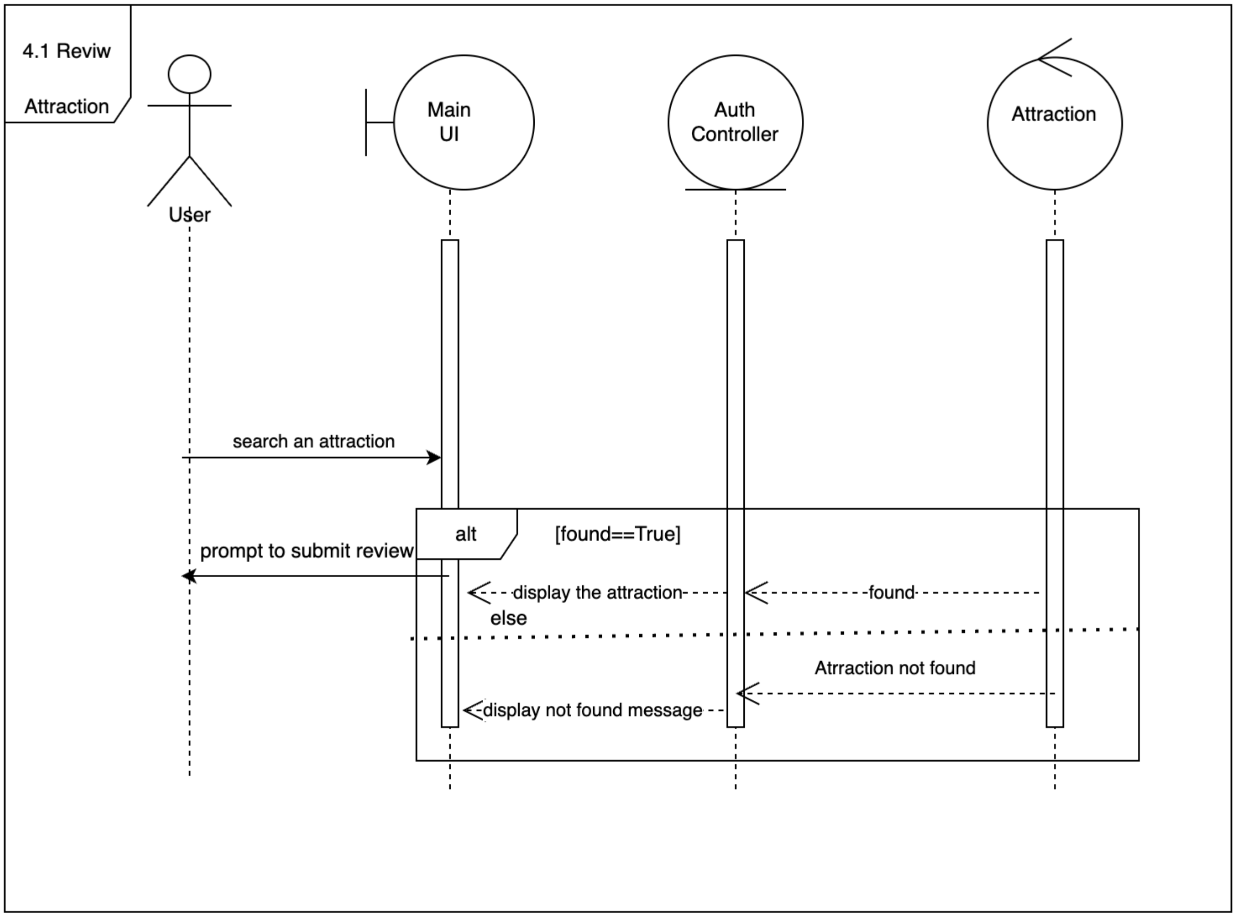

sd
SearchAttraction
User
Main UI
Search Attraction Controller
Attraction Database
1.enterSearch
2.sendSearchRequest
3.fetchAttractions
4.returnAttractions
5.displayAttraction
opt
6. applyFilters
7. processFilters
8.updateSearchResults
9.displayUpdatedSearchResults
10. enterSearch
11. navigateToDetails

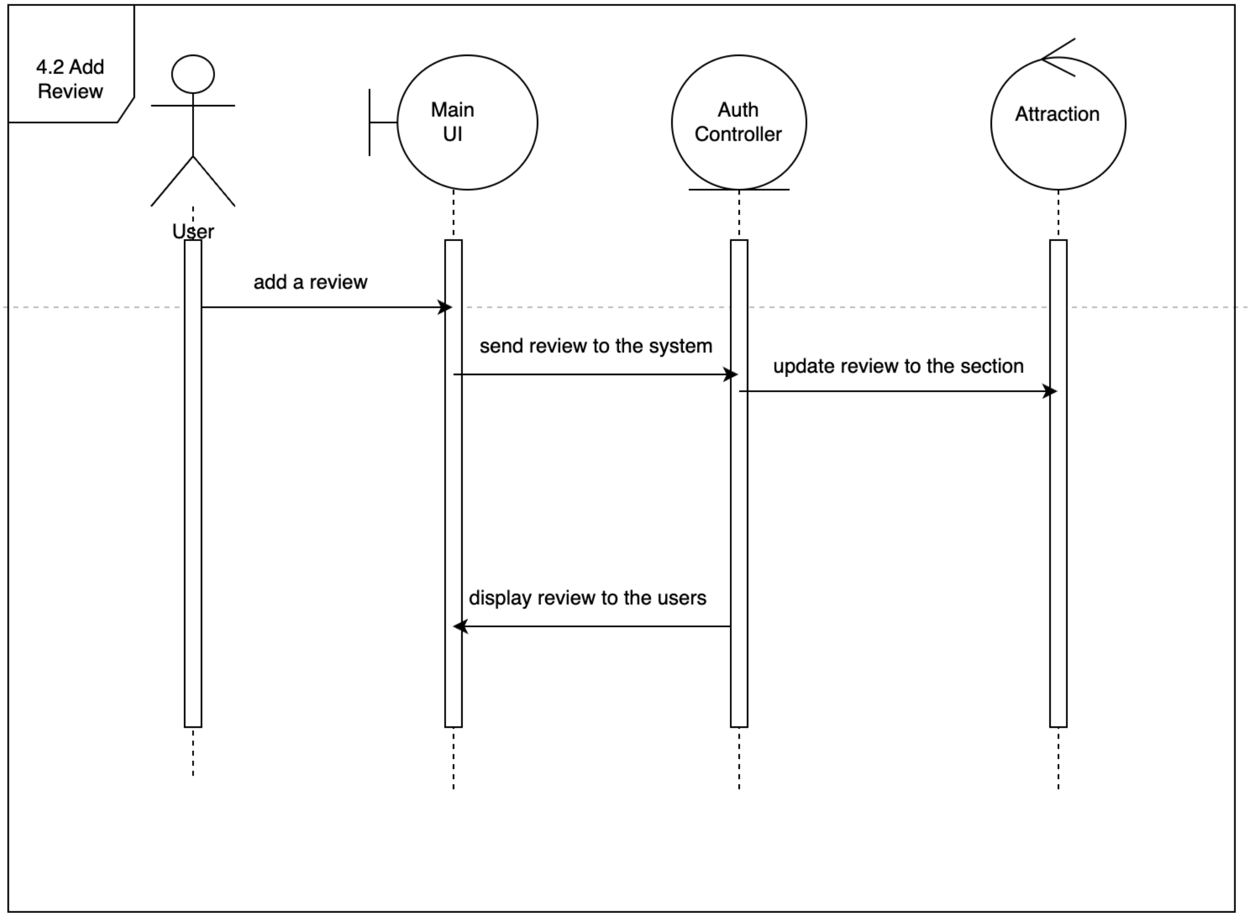

sd ViewDetails
User
Main UI
System Controller
Attraction Database
1.userSelectAttraction
2.requestAttractionDetails
3.fetchAttractionDetails
4.returnAttractionDetils
5. displayAttractionDetails
opt
6.clickBackButton
7.userRequestBack
8. redirectUserToSearchListings

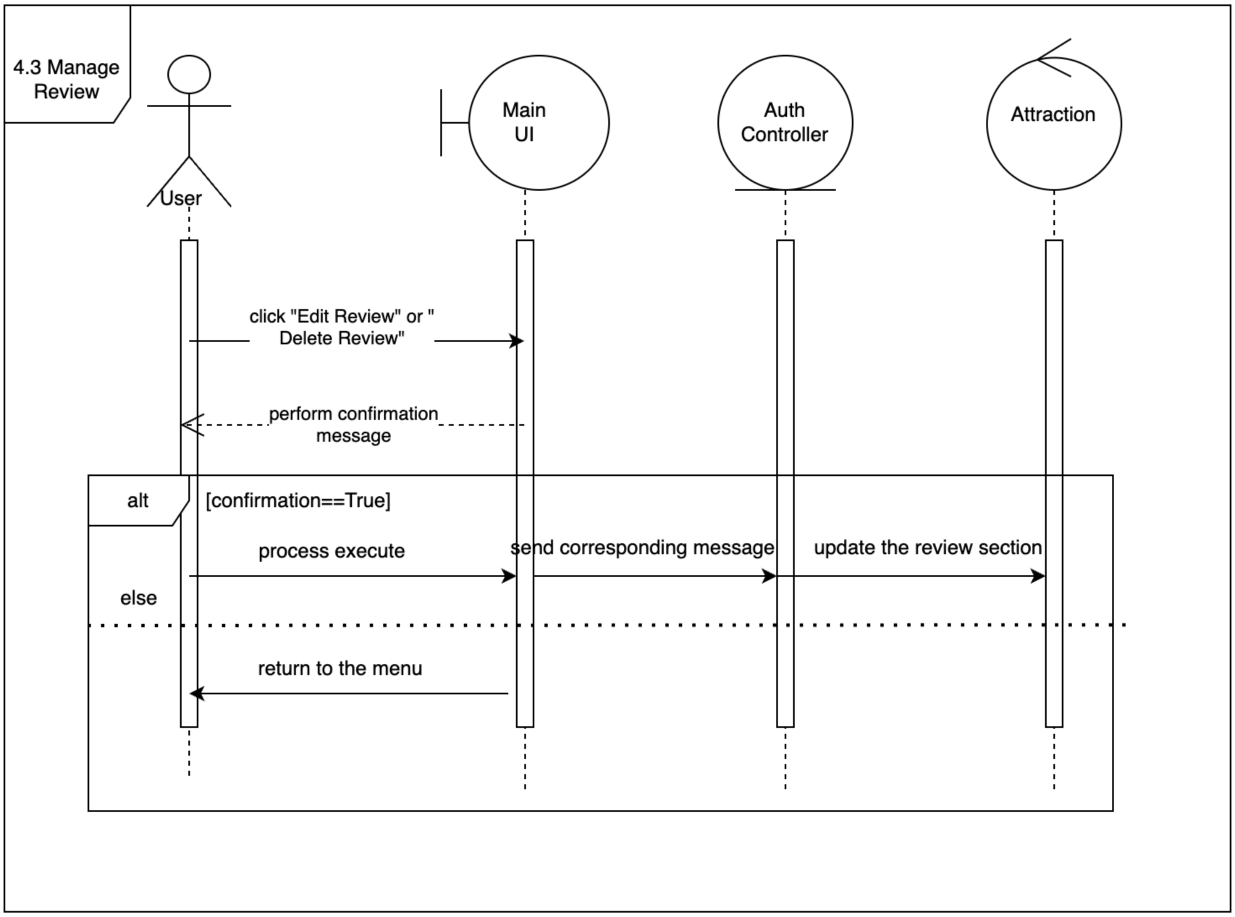

sd BookAttraction
User
Main UI
System Controller
Attraction Database
1.userSelectAttraction
2. requestAttractionDetails
3.fetchAttractionDetails
4.returnAttractionDetails
5.sendAttractionDetails
6. systemDisplayDetails
7. userIndicateTickets
8.confirmDetails
9.sendConfirmationPrompt
10. userConfirmBooking
11. processBooking
12. reserveSlots
13.confirmReservation
14.redirectUserToPayment

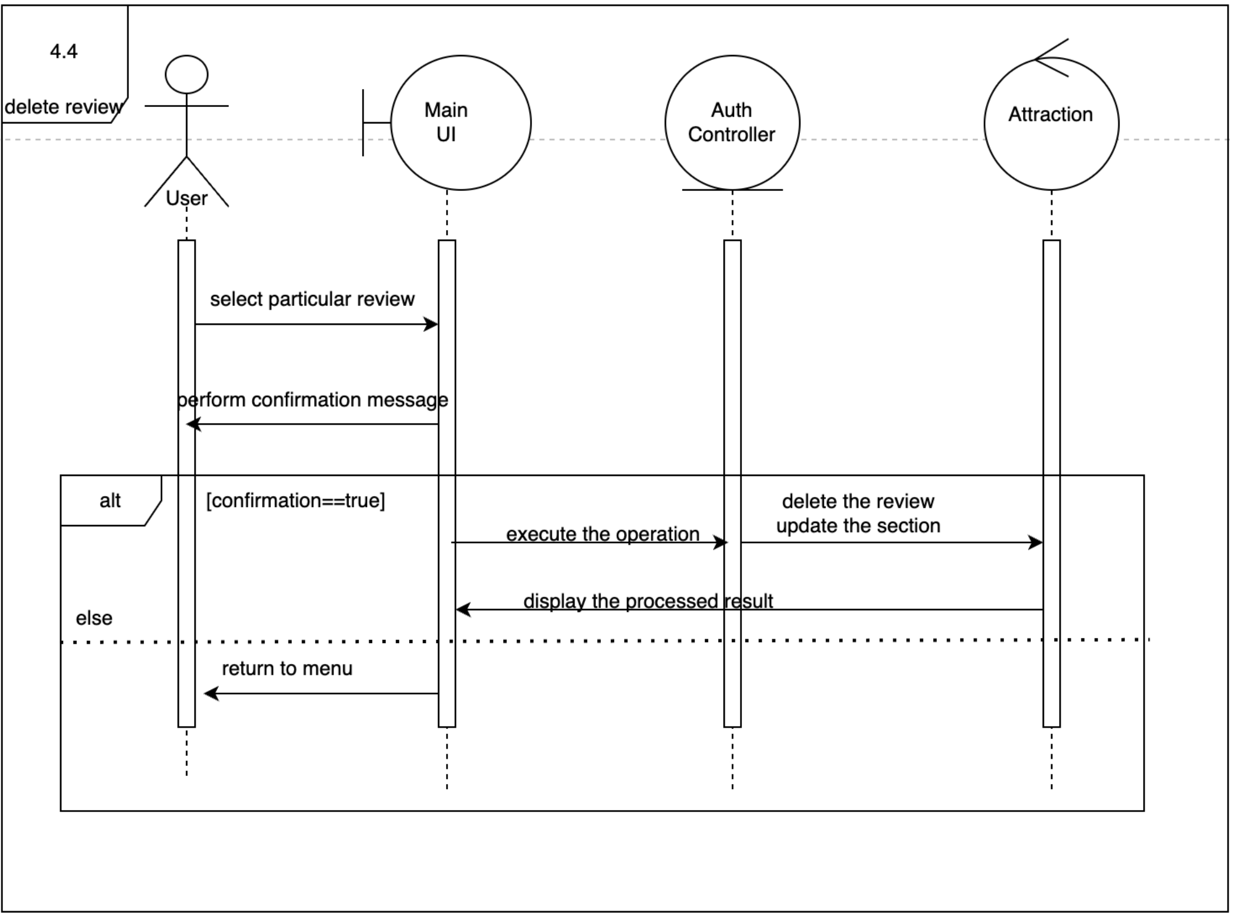

3.1 Payment
main UI
Auth Controller
Payment Gateway
initiate payment for booking
verify user authentic
alt
[verification]==True
authentic confirmed
display payment method options
else
payment denied
select payment method
redirect to payment gateway
requeste a confirmation
confirm
alt
[confirmation]==true
display payment success
else
return to main UI

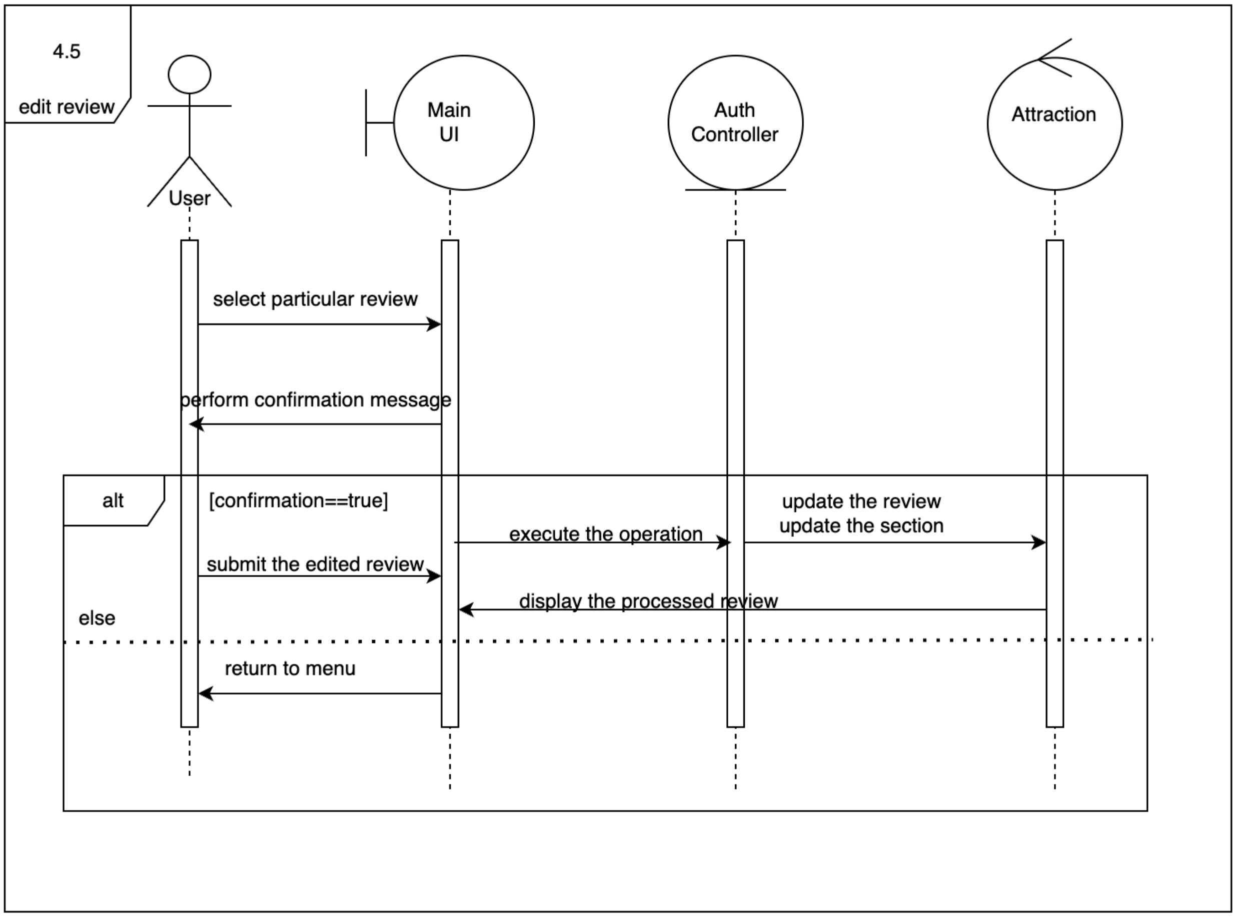

3.2 SelectPaymentType
main UI
Auth Controller
Payment Gateway
select payment method
verify authentic
authenUseconfirmed
prompt for payment detail
enter card detail/scan QR code
request payment confirmation
confirm payment
process payment
return payment detail
alt
[payment success]==True
display confirmation page
send email
else
display transaction page
prompt for retry

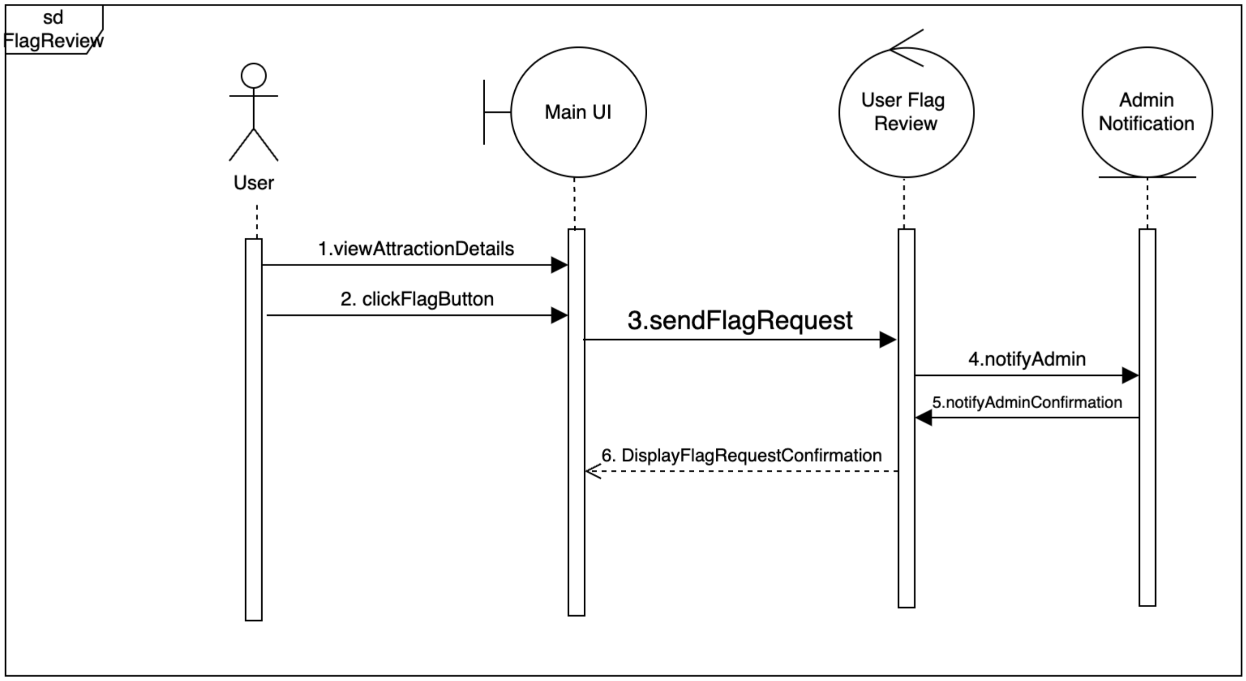

4.1 Reviw
Attraction
User
Main UI
Auth Controller
Attraction
search an attraction
alt
[found==True]
prompt to submit review
display the attraction
found
else
Atrraction not found
display not found message

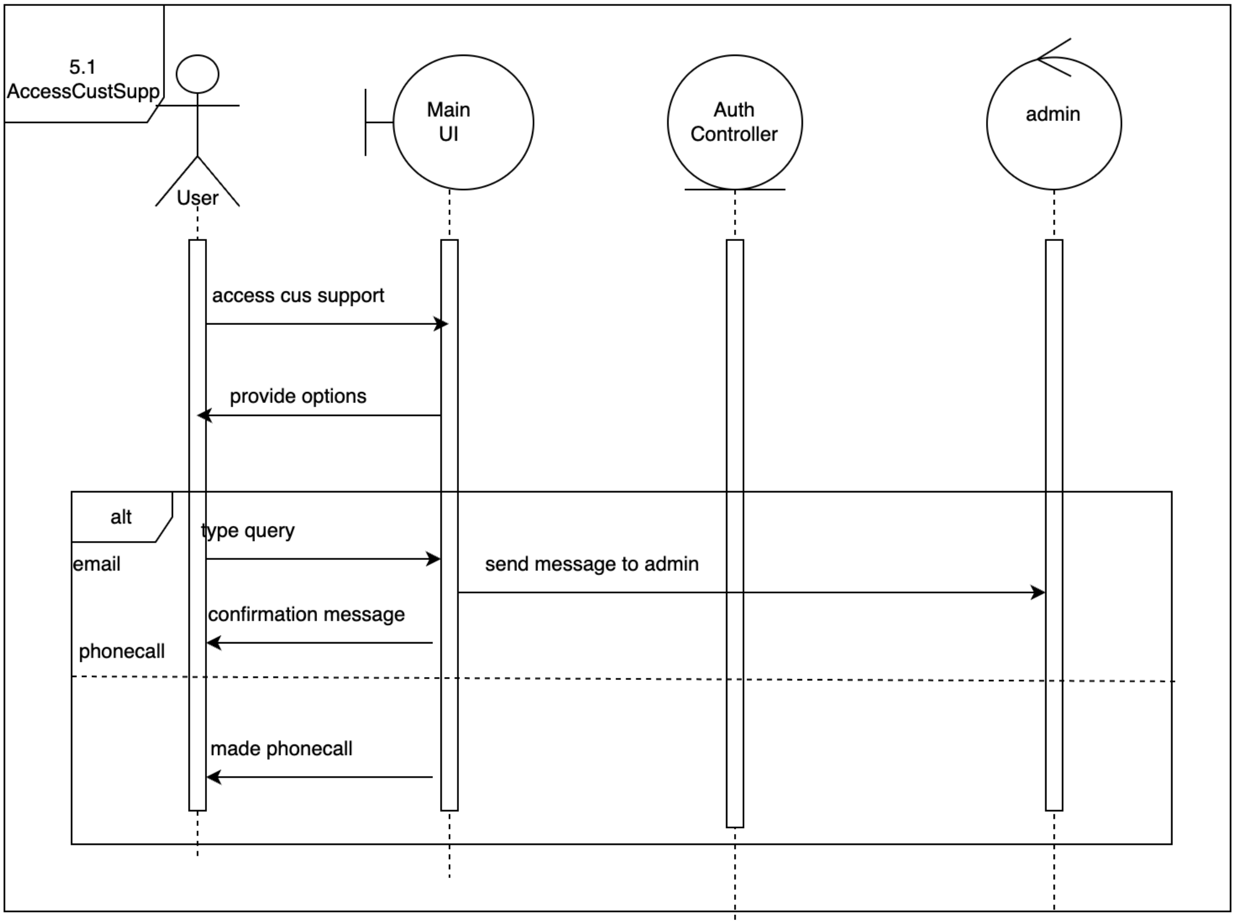

4.2 Add Review
User
Main UI
Auth Controller
Attraction
add a review
send review to the system
update review to the section
display review to the users

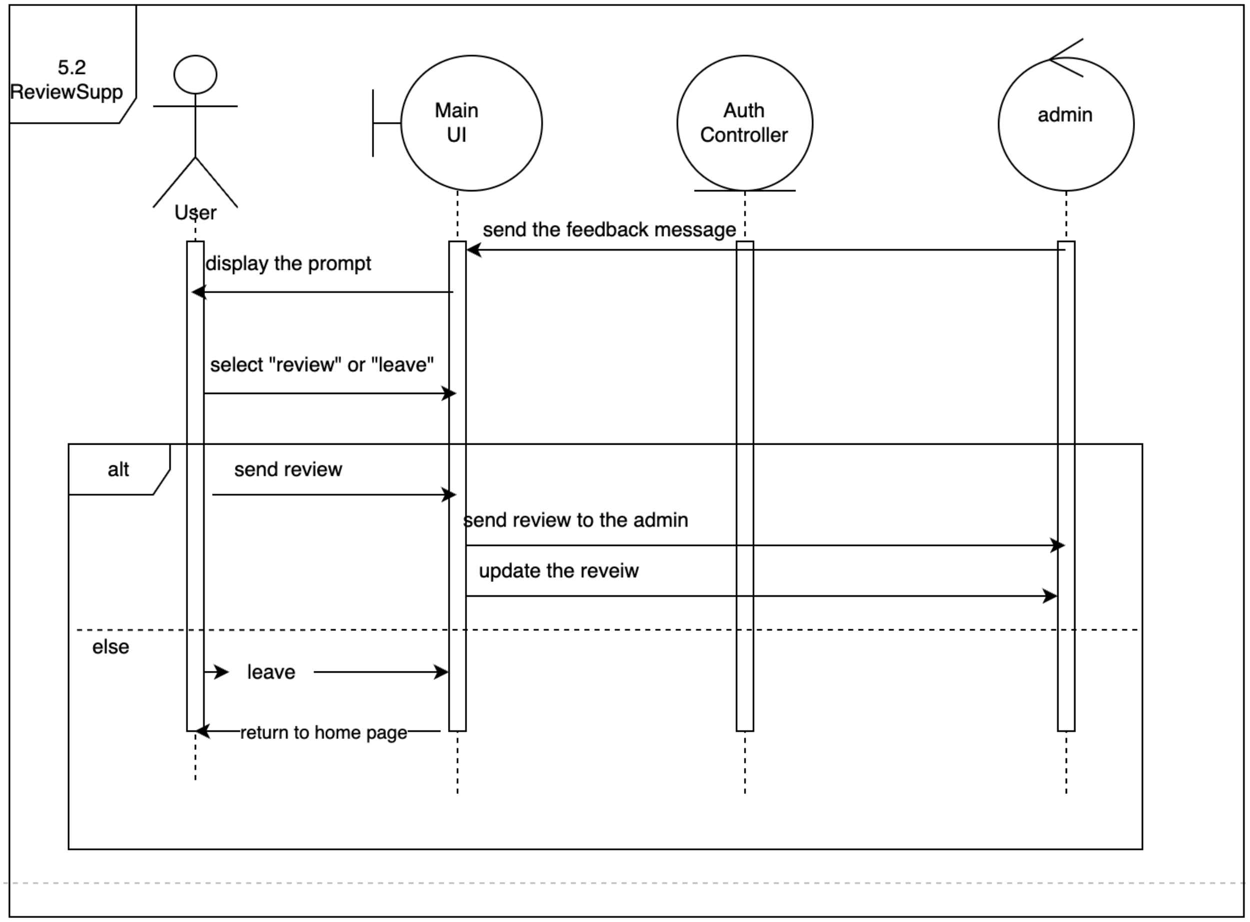

4.3 Manage Review
User
Main UI
Auth Controller
Attraction
click "Edit Review" or " Delete Review"
perform confirmation message
alt
[confirmation==True]
process execute
send corresponding message
update the review section
else
return to the menu

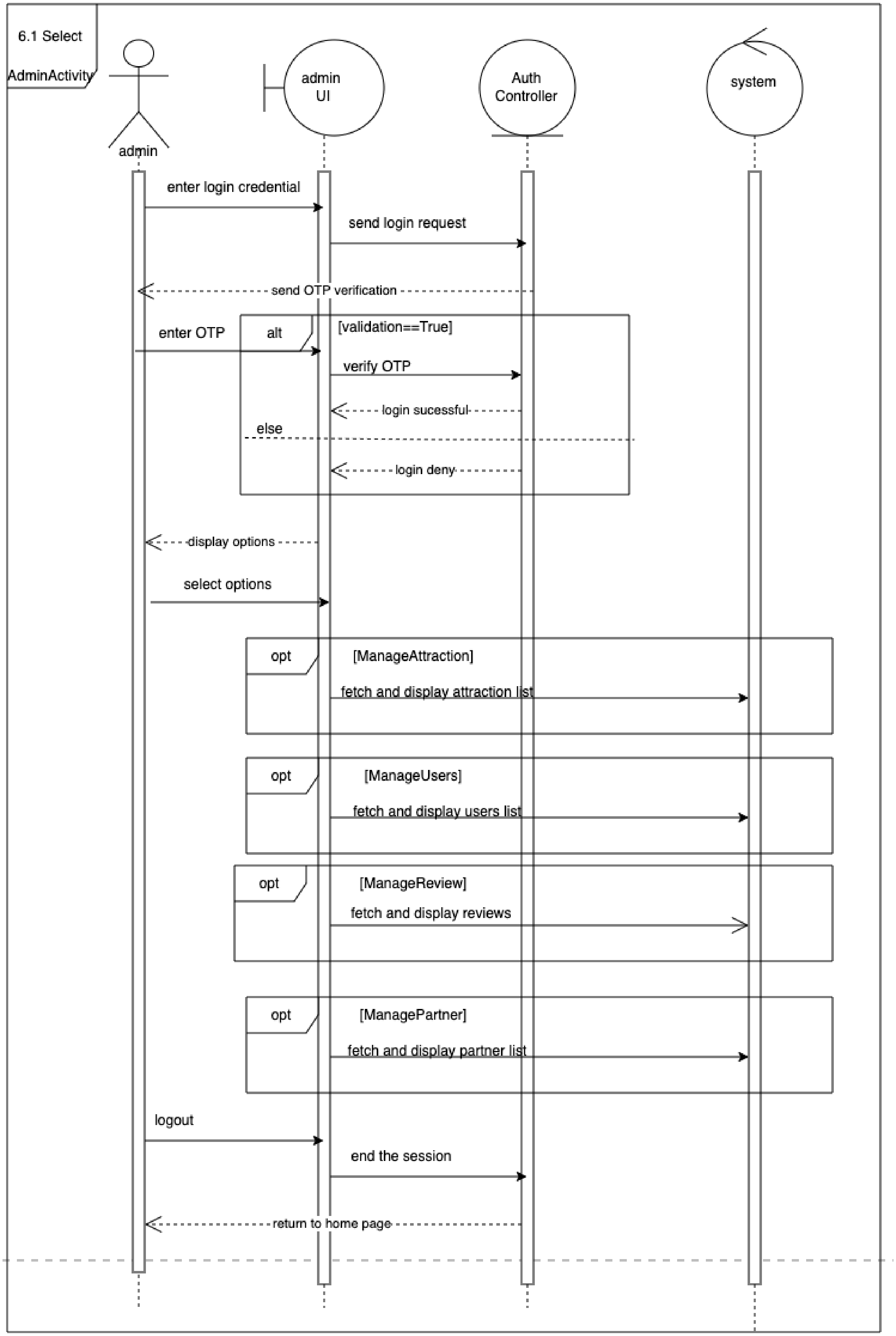

4.4
delete review
User
Main UI
Auth Controller
Attraction
select particular review
perform confirmation message
alt
[confirmation==true]
delete the review
update the section
execute the operation
display the processed result
else
return to menu

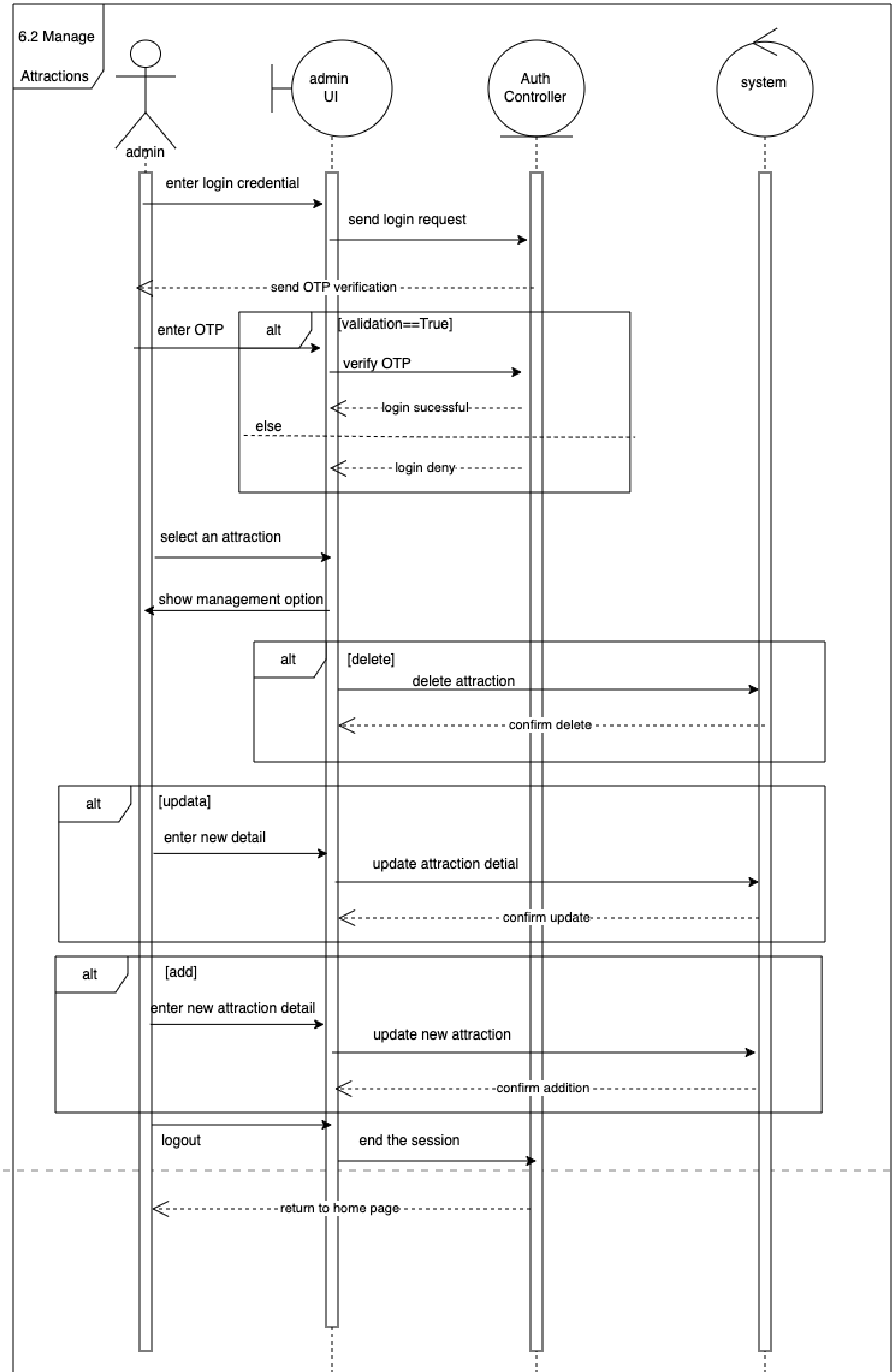

4.5
edit review
User
Main UI
Auth Controller
Attraction
select particular review
perform confirmation message
alt
[confirmation==true]
execute the operation
update the review
update the section
submit the edited review
display the processed review
else
return to menu

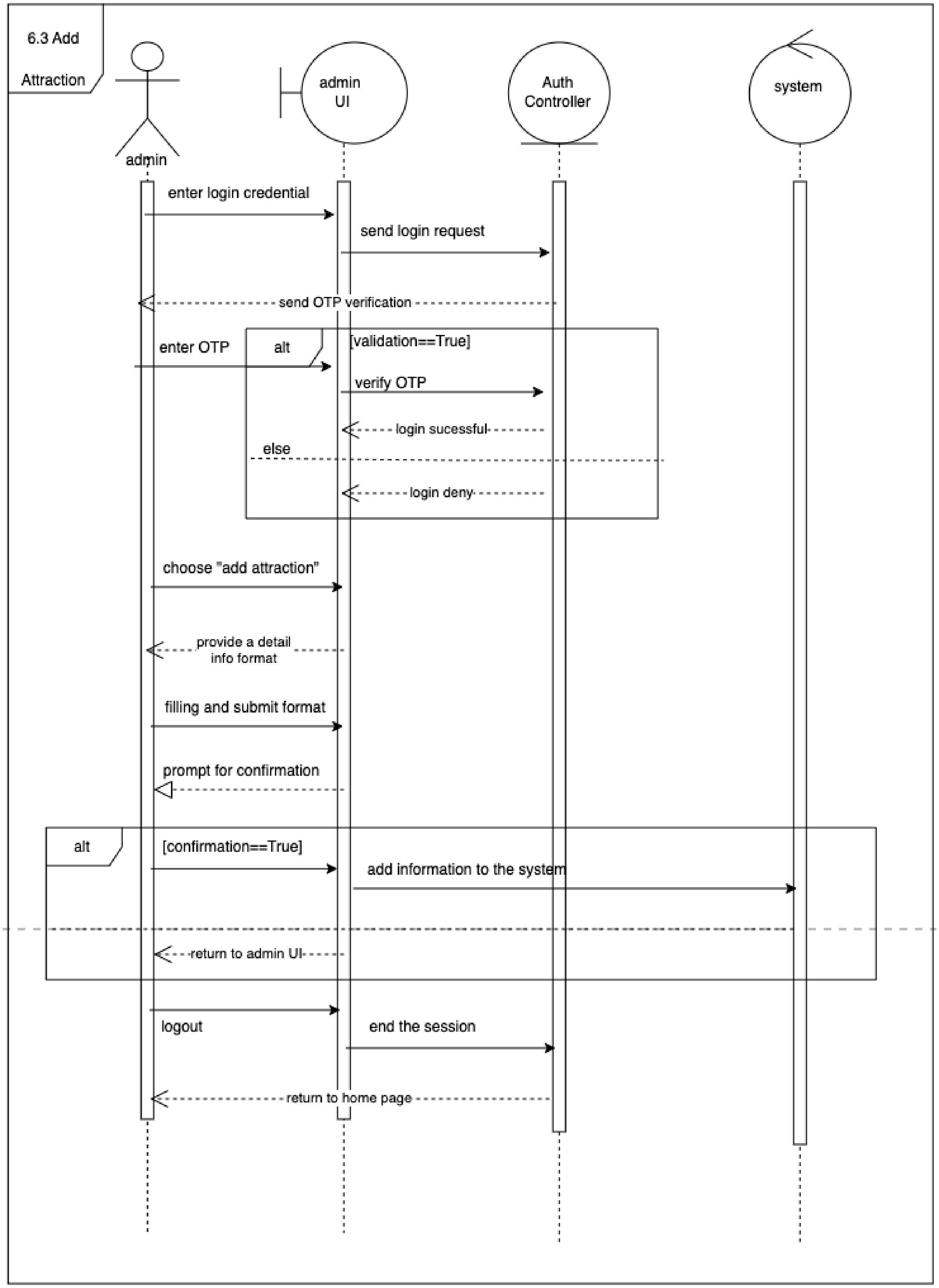

sd FlagReview
User
Main UI
User Flag Review
Admin Notification
1.viewAttractionDetails
2. clickFlagButton
3.sendFlagRequest
4.notifyAdmin
5.notifyAdminConfirmation
6. DisplayFlagRequestConfirmation

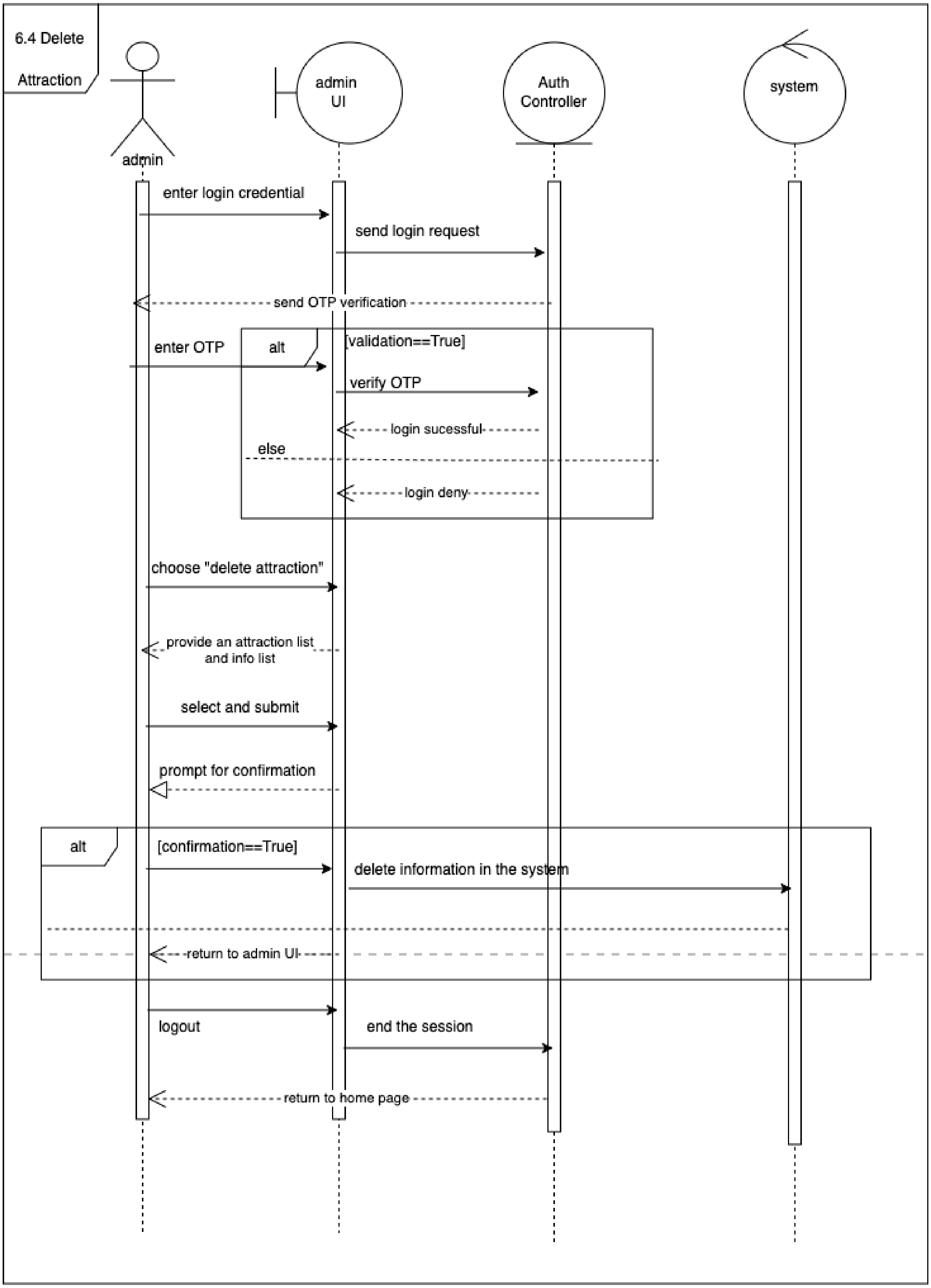

5.1 AccessCustSupp
User
Main UI
Auth Controller
admin
access cus support
provide options
alt
type query
email
send message to admin
confirmation message
phonecall
made phonecall

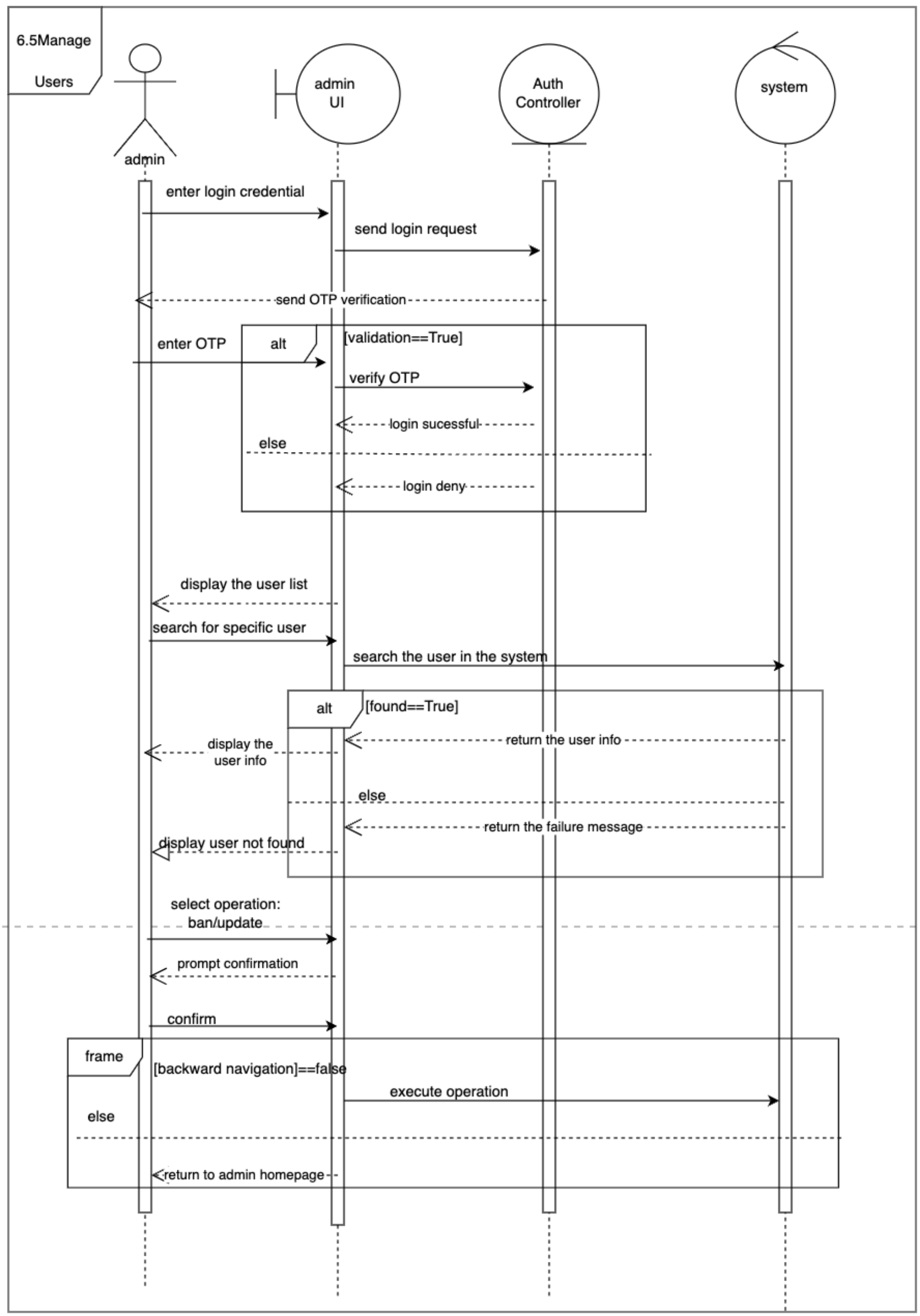

5.2
ReviewSupp
User
Main
UI
Auth
Controller
admin
send the feedback message
display the prompt
select "review" or "leave"
alt
send review
send review to the admin
update the reveiw
else
leave
return to home page

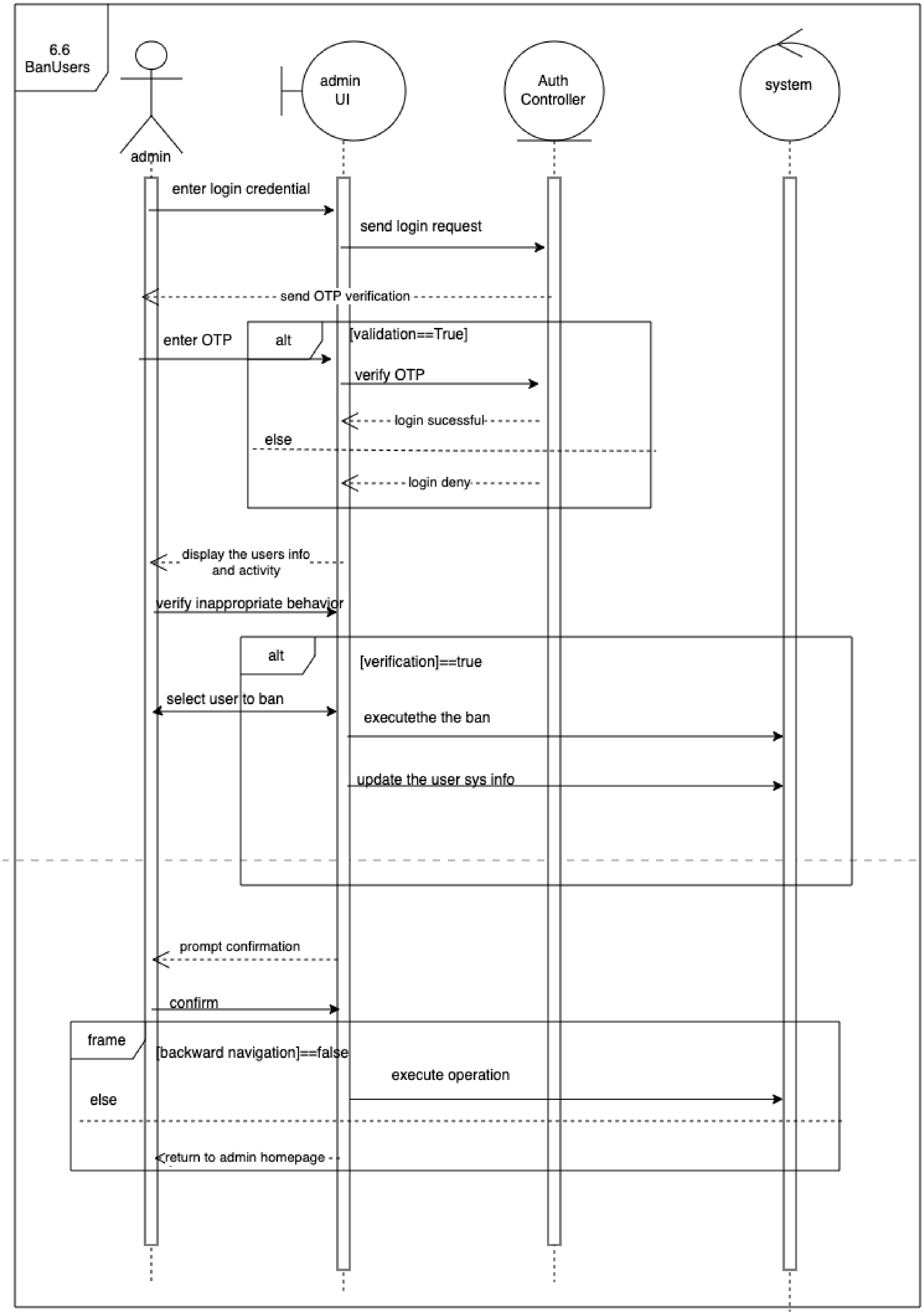

6.1 Select AdminActivity
admin
admin UI
Auth Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
display options
select options
opt
[ManageAttraction]
fetch and display attraction list
opt
[ManageUsers]
fetch and display users list
opt
[ManageReview]
fetch and display reviews
opt
[ManagePartner]
fetch and display partner list
logout
end the session
return to home page

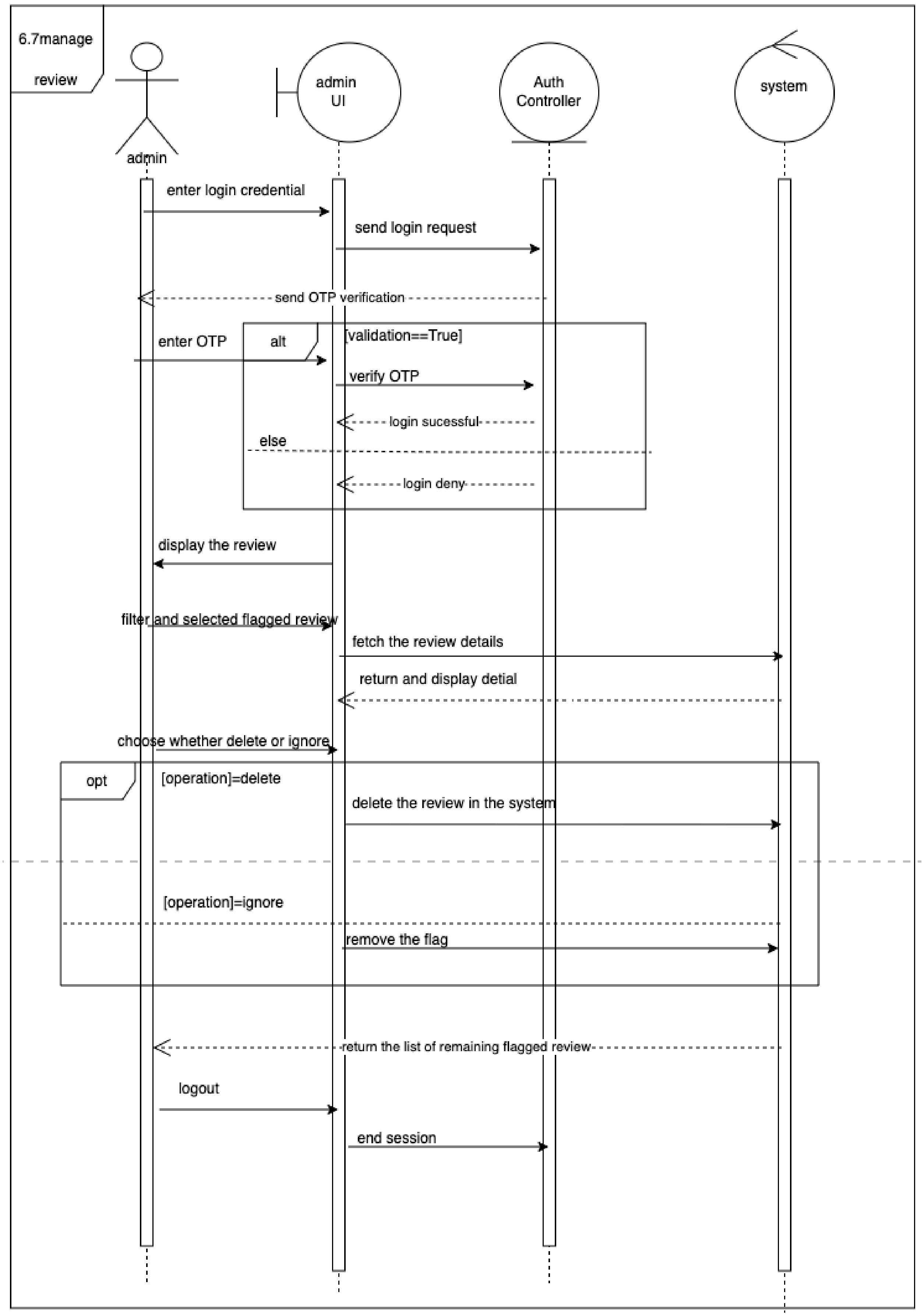

6.2 Manage Attractions
admin
admin UI
Auth Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
select an attraction
show management option
alt
[delete]
delete attraction
confirm delete
alt
[updata]
enter new detail
update attraction detial
confirm update
alt
[add]
enter new attraction detail
update new attraction
confirm addition
logout
end the session
return to home page


6.3 Add
Attraction
admin
admin UI
Auth Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
choose "add attraction"
provide a detail info format
filling and submit format
prompt for confirmation
alt
[confirmation==True]
add information to the system
return to admin UI
logout
end the session
return to home page

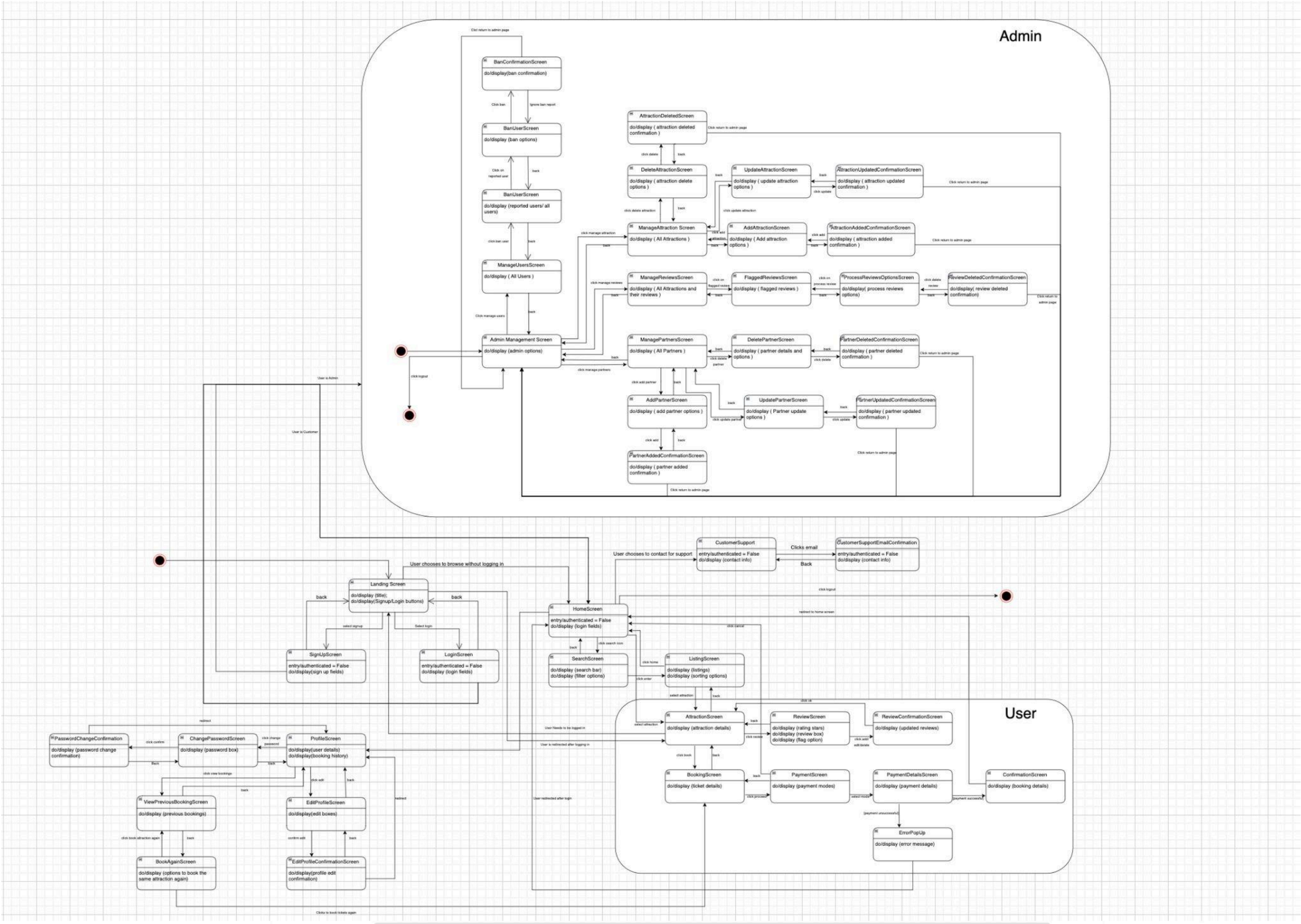

6.4 Delete Attraction
admin
admin UI
Auth Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
choose "delete attraction"
provide an attraction list and info list
select and submit
prompt for confirmation
alt
[confirmation==True]
delete information in the system
return to admin UI
logout
end the session
return to home page


6.5Manage
Users
admin
admin
UI
Auth
Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
display the user list
search for specific user
search the user in the system
alt
[found==True]
return the user info
display the
user info
else
return the failure message
display user not found
select operation:
ban/update
prompt confirmation
confirm
frame
[backward navigation]==false
execute operation
else
return to admin homepage

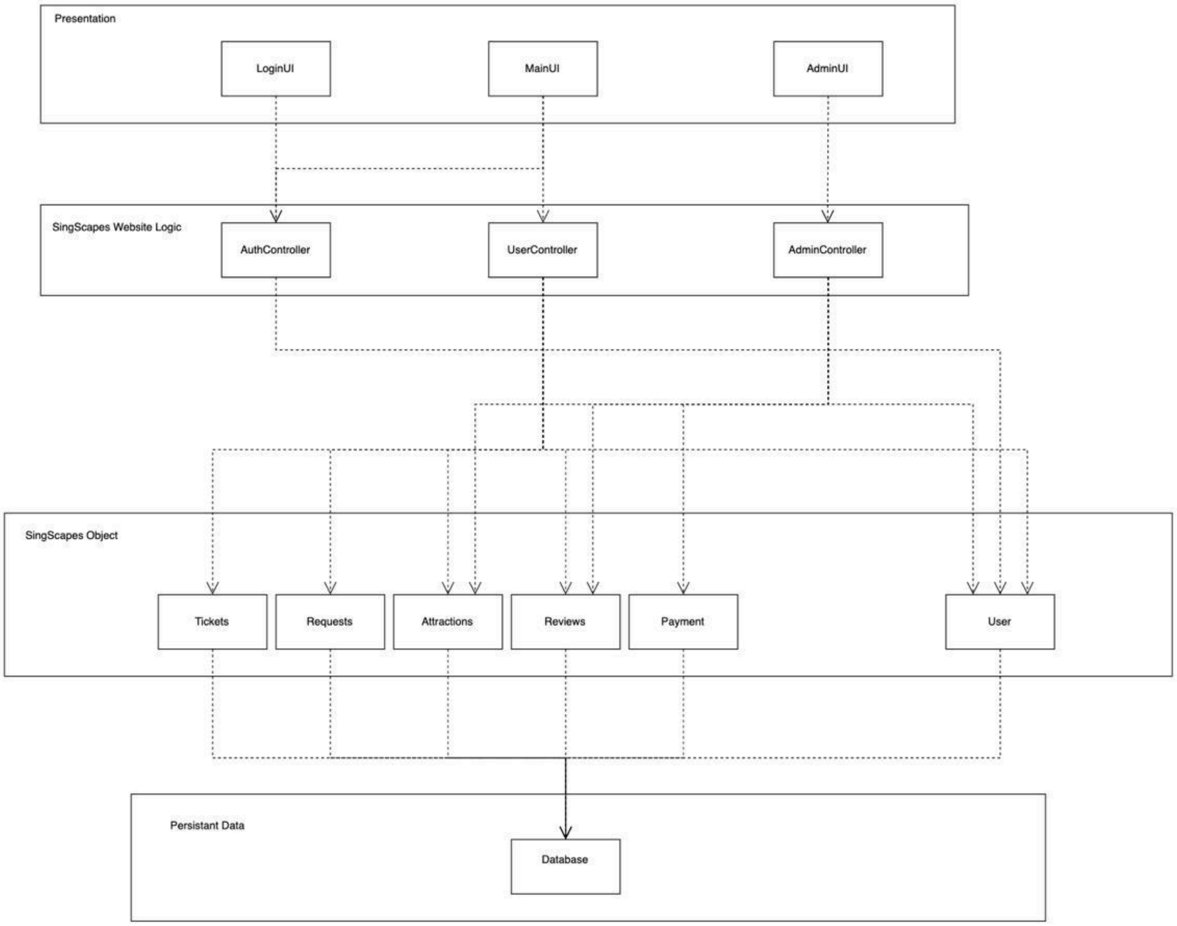

6.6
BanUsers
admin
admin
UI
Auth
Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
display the users info
and activity
verify inappropriate behavior
alt
[verification]==true
select user to ban
executethe the ban
update the user sys info
prompt confirmation
confirm
frame
[backward navigation]==false
execute operation
else
return to admin homepage


6.7manage review
admin
admin UI
Auth Controller
system
enter login credential
send login request
send OTP verification
alt
[validation==True]
enter OTP
verify OTP
login sucessful
else
login deny
display the review
filter and selected flagged review
fetch the review details
return and display detial
choose whether delete or ignore
opt
[operation]=delete
delete the review in the system
[operation]=ignore
remove the flag
return the list of remaining flagged review
logout
end session

## c. Dialog Map

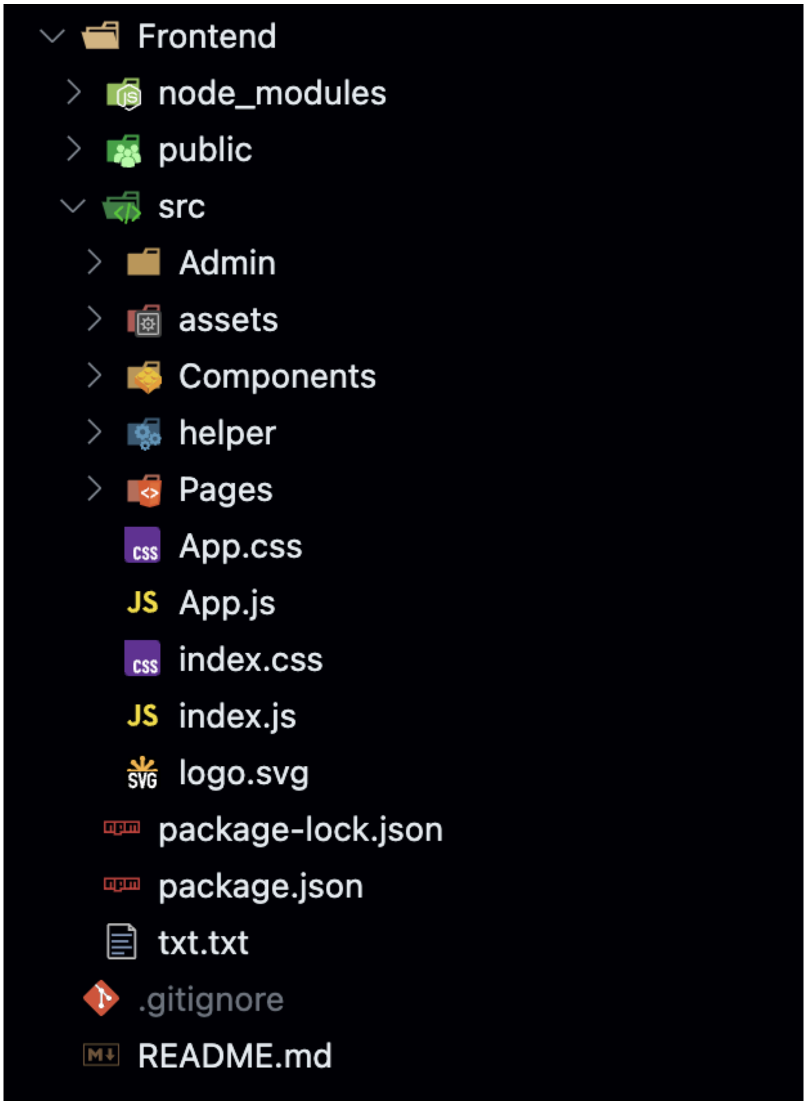

Admin
Click return to admin page
BanConfirmationScreen
do/display(ban confirmation)
Click ban
Ignore ban report
BanUserScreen
do/display (ban options)
Click on reported user
back
BanUserScreen
do/display (reported users/ all users)
click ban user
back
ManageUsersScreen
do/display ( All Users )
Click manage users
back
Admin Management Screen
do/display (admin options)
click logout
User is Admin
AttractionDeletedScreen
do/display ( attraction deleted confirmation )
Click return to admin page
click delete
back
DeleteAttractionScreen
do/display ( attraction delete options )
click delete attraction
back
UpdateAttractionScreen
do/display ( update attraction options )
click update attraction
AttractionUpdatedConfirmationScreen
do/display ( attraction updated confirmation )
click update
Click return to admin page
click manage attraction
ManageAttraction Screen
do/display ( All Attractions )
click add attraction
AddAttractionScreen
do/display ( Add attraction options )
click add
AttractionAddedConfirmationScreen
do/display ( attraction added confirmation )
Click return to admin page
click manage reviews
ManageReviewsScreen
do/display ( All Attractions and their reviews )
click on flagged review
FlaggedReviewsScreen
do/display ( flagged reviews )
click on process review
ProcessReviewsOptionsScreen
do/display( process reviews options)
click delete review
ReviewDeletedConfirmationScreen
do/display( review deleted confirmation)
Click return to admin page
click manage partners
ManagePartnersScreen
do/display ( All Partners )
click delete partner
DeletePartnerScreen
do/display ( partner details and options )
click delete
PartnerDeletedConfirmationScreen
do/display ( partner deleted confirmation )
Click return to admin page
click add partner
AddPartnerScreen
do/display ( add partner options )
click update partner
UpdatePartnerScreen
do/display ( Partner update options )
click update
PartnerUpdatedConfirmationScreen
do/display ( partner updated confirmation )
Click return to admin page
click add
PartnerAddedConfirmationScreen
do/display ( partner added confirmation )
Click return to admin page
back
User is Customer
CustomerSupport
entry/authenticated = False
do/display (contact info)
Clicks email
Back
CustomerSupportEmailConfirmation
entry/authenticated = False
do/display (contact info)
User chooses to contact for support
User chooses to browse without logging in
Landing Screen
do/display (title);
do/display(Signup/Login buttons)
back
select signup
Select login
SignUpScreen
entry/authenticated = False
do/display(sign up fields)
LoginScreen
entry/authenticated = False
do/display (login fields)
HomeScreen
entry/authenticated = False
do/display (login fields)
click logout
redirect to home screen
click cancel
click search icon
SearchScreen
do/display (search bar)
do/display (filter options)
click enter
click home
ListingScreen
do/display (listings)
do/display (sorting options)
select attraction
User
click ok
AttractionScreen
do/display (attraction details)
click review
ReviewScreen
do/display (rating stars)
do/display (review box)
do/display (flag option)
click add/ edit/delete
ReviewConfirmationScreen
do/display (updated reviews)
User Needs to be logged in
User is redirected after logging in
click book
BookingScreen
do/display (ticket details)
click proceed
PaymentScreen
do/display (payment modes)
select mode
PaymentDetailsScreen
do/display (payment details)
[payment successful]
ConfirmationScreen
do/display (booking details)
[payment unsuccessful]
ErrorPopUp
do/display (error message)
User redirected after login
redirect
PasswordChangeConfirmation
do/display (password change confirmation)
click confirm
Back
ChangePasswordScreen
do/display (password box)
click change password
ProfileScreen
do/display(user details)
do/display(booking history)
click view bookings
click edit
ViewPreviousBookingScreen
do/display (previous bookings)
EditProfileScreen
do/display(edit boxes)
click book attraction again
confirm edit
BookAgainScreen
do/display (options to book the same attraction again)
EditProfileConfirmationScreen
do/display(profile edit confirmation)
Clicks to book tickets again

## III. System Architecture:



# IV. Application Skeleton

## a. Frontend



```
Frontend
  node_modules
  public
  src
    Admin
    assets
    Components
    helper
    Pages
    App.css
    App.js
    index.css
    index.js
    logo.svg
  package-lock.json
  package.json
  txt.txt
.gitignore
README.md
```

Built with React.js framework

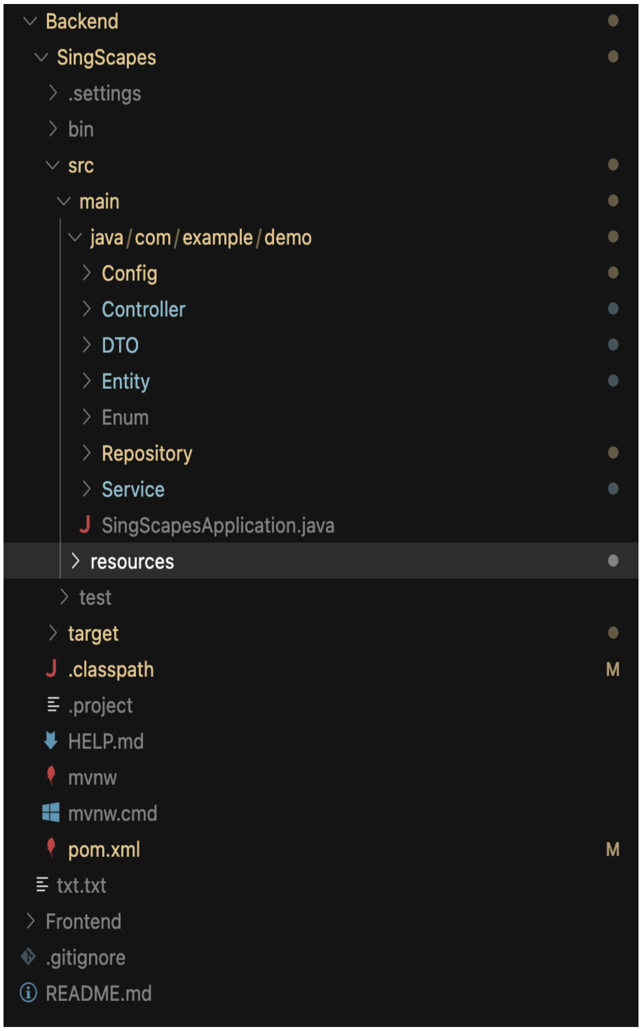
Frontend consists of the User Interfaces (Pages), which are structured into different UI.

Admin UI is placed separately

Assets contains images and animations used in the web app.

Components consists of reusable modules that are commonly used throughout the web app.

## b. Backend



```
Backend
  SingScapes
    .settings
    bin
    src
      main
        java/com/example/demo
          Config
          Controller
          DTO
          Entity
          Enum
          Repository
          Service
          SingScapesApplication.java
        resources
      test
    target
    .classpath                                  M
    .project
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml                                     M
  txt.txt
Frontend
.gitignore
README.md
```

**Built with Spring Boot framework**
**Database used: PostgreSQL**

**Config:** Contains configuration files for application-wide settings, such as database configurations, application properties, and other necessary setup files.

**Controller:** Routes incoming HTTP requests and forwards them to the corresponding service methods. Also, it forms REST API endpoints for interaction with the frontend.

**DTO (Data Transfer Objects):** Enables transfer of structured data across layers without showing internal entity structures.

**Entity:** Defines the data models for the tables in the database. Entities are mapped using JPA annotations.

**Enum:** Contains enumerations that standardize certain sets of values, enhancing data consistency.

**Repository:** Interfaces that expose database operations through Spring Data JPA, including Create, Read, Update, and Delete (CRUD) operations.

**Service:** The service layer comprises the business logic of the application, managing communication among controllers and repositories to maintain integrity in data and accurate processing.

**resources/:** Contains configuration files, such as application properties.

**target/:** Holds compiled Java classes and built application files.

**HELP.md & README.md:** Project setup and use documentation.

# V. Appendix

## a. Design Patterns Used

Identifying and Storing Persistent Data:

Relational Database:

- Attraction: **attractionid (uuid)**, name (varchar), type (varchar), location (varchar), postal (varchar), description (varchar), rating (float)

- Profile: **profileid (uuid)**, userid (=userid), email (varchar), phone_no (varchar), flagged (bool), is_admin (bool), created_at (datetime), updated_at (datetime), deleted_at (datetime)

- Booking: **bookingid (uuid)**, userid (=userid), status (varchar), tickets (jsonb), paymentid (=paymentid), created_at (datetime), updated_at (datetime), deleted_at (datetime)

- Payment: **paymentid (uuid)**, amount (float), mode (varchar)

- Review: reviewid (uuid), userid (=userid), text (varchar), flagged (bool), created_at (datetime), updated_at (datetime), deleted_at (datetime)

- Ticket: **ticketid (uuid)**, date (datetime), price (float), type (varchar), attractionid (=attractionid)

Access Control:

| Actors | User | Attraction | Booking |
|---|---|---|---|
| User | getUser() | getAllAttractions() | createBooking()<br>getBooking() |
| Admin | getUser()<br>flagUser()<br>getAllFlaggedUsers() | getAllAttractions()<br>addAttraction()<br>deleteAttraction()<br>updateAttraction() | createBooking()<br>getBooking()<br>updateBooking() |

| Actors | Payment | Review | Ticket |
|---|---|---|---|
| User | validateCardDetails() | getReviews()<br>submitReview()<br>editReview()<br>deleteReview() | getAllTicketTypes()<br>getTicketPrice() |
| Admin | validateCardDetails() | getReviews()<br>submitReview()<br>editReview()<br>deleteReview()<br>flagReview()<br>getAllFlaggedReviews() | getAllTicketTypes()<br>getTicketPrice()<br>addTicketPrice()<br>updateTicket() |

b. Tech Stack:

Frontend: React.js
Backend: Springboot (Java)
Database: PostgreSQL