Capstone Project Report

Report 4 – Software Design Document

Table of Contents

I. Software Design Document 4

1. System Design. 4

1.1 System Architecture. 4

            1.1.1 Overall System Architecture……………………………………………………………………………………………….4

1.2 Package Diagram.. 4

2. Database Design. 4

3. Detailed Design. 5

3.1 <Feature/Function Name1>. 5

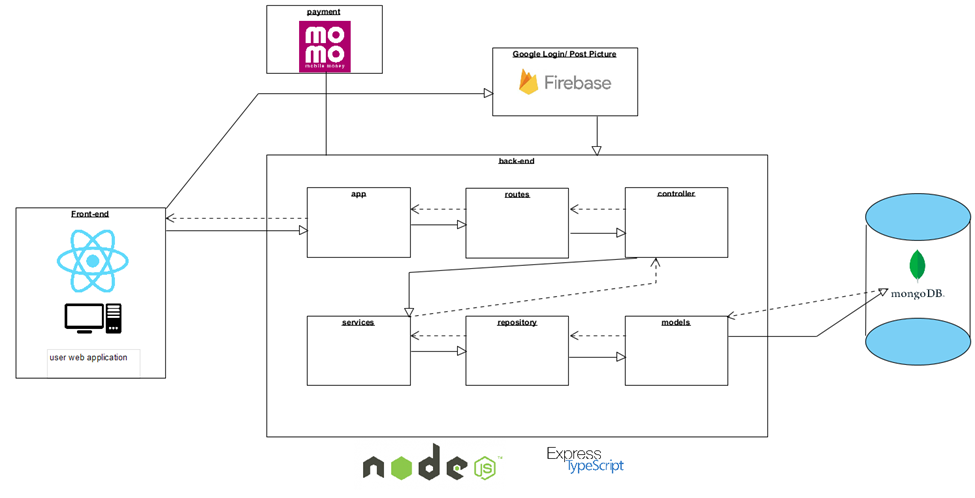
3.2 <Feature/Function Name2>. 6

# I. Software Design Document

## 1. System Design

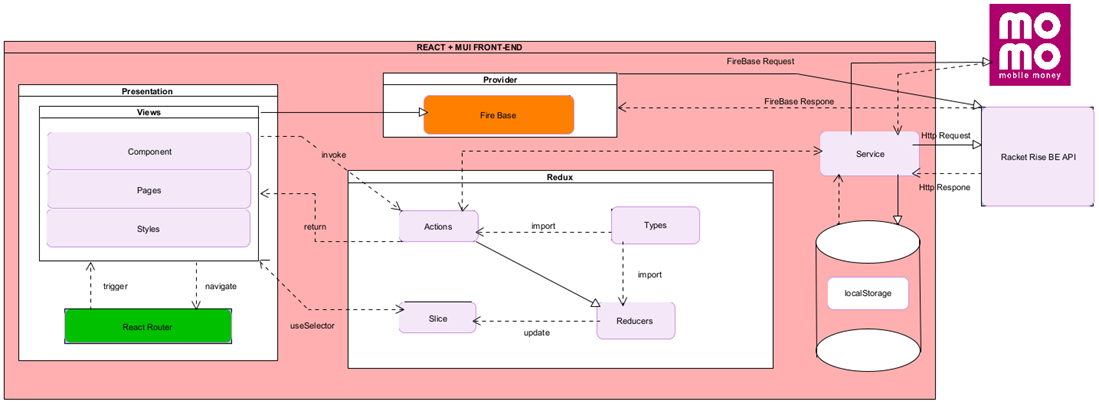
### 1.1 System Architecture

#### 1.1.1  Overall System Architecture

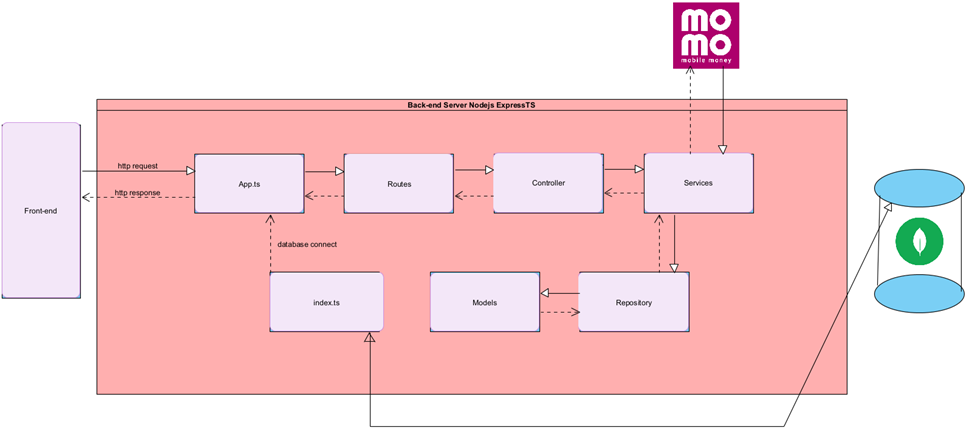


#### 1.1.2 Detail System Architecture

##### 1.1.2.1 User Web Application Sub-system (Front-end)



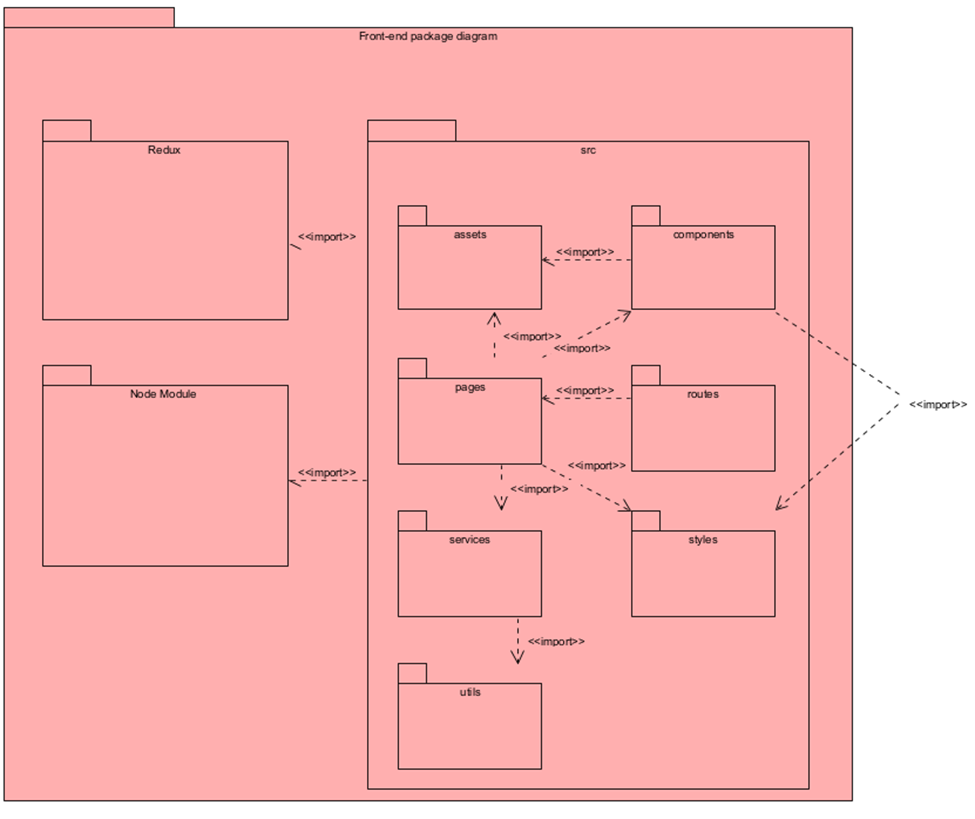
##### 1.1.2.2 Backend WebAPI Sub-system



### 1.2 Package Diagram

*[Provide the package diagram for each sub-system. The content of this section includes overall package diagram(s) and the explanation for each package (or namespace). Please see the sample and description table format below]*

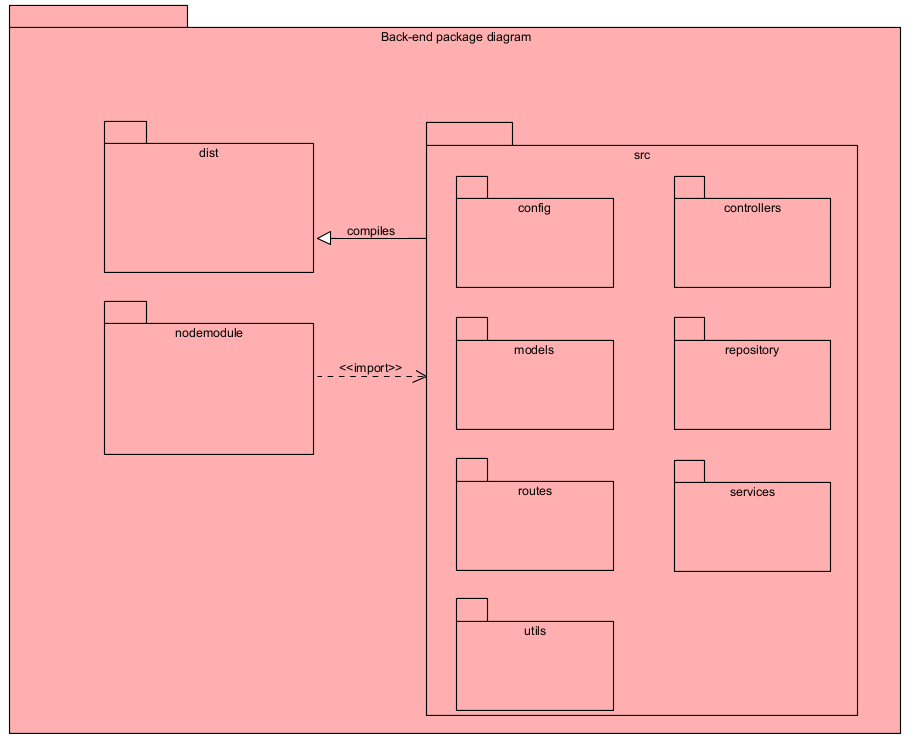
#### 1.2.1 Front-end Package Diagram



#### 1.2.2 Package Descriptions

|  |  |  |
| --- | --- | --- |
| No | Package | Description |
| 01 | Redux | This package manages the application's state using Redux, a predictable state container for JavaScript apps. It includes actions, reducers, and middleware to handle state changes and side effects. |
| 02 | Node Module | This directory contains the external Node.js modules and dependencies that the project relies on. These modules are imported into the source code to provide additional functionality and features. |
| 03 | src | The main source directory that houses the core of the application's codebase. It includes all the sub-packages and modules necessary for the application to function. |
| 04 | src/assets | This folder contains static assets like images, fonts, and other media files used in the application. |
| 05 | src/components | This package includes reusable UI components. Each component represents a piece of the user interface, such as buttons, header, footer, input fields, and other interactive elements. |
| 06 | src/pages | This package contains page-level components that represent different views or screens in the application. Each page is usually tied to a route and may contain multiple smaller components. |
| 07 | src/routes | This package manages the routing configuration of the application. It defines the various routes and their corresponding components or pages, enabling navigation within the app. |
| 08 | src/services | This package includes service modules responsible for handling API calls, data fetching, and other asynchronous operations. It abstracts the logic for interacting with external services. |
| 09 | src/styles | This package contains global and component-specific styles, including CSS, SCSS, or styled-components. It manages the overall look and feel of the application. |
| 10 | src/utils | This package provides utility functions and helper methods used throughout the application. These utilities perform common tasks such as data formatting, calculations, and other reusable logic. |

#### 1.3.1 Backend Package Diagram

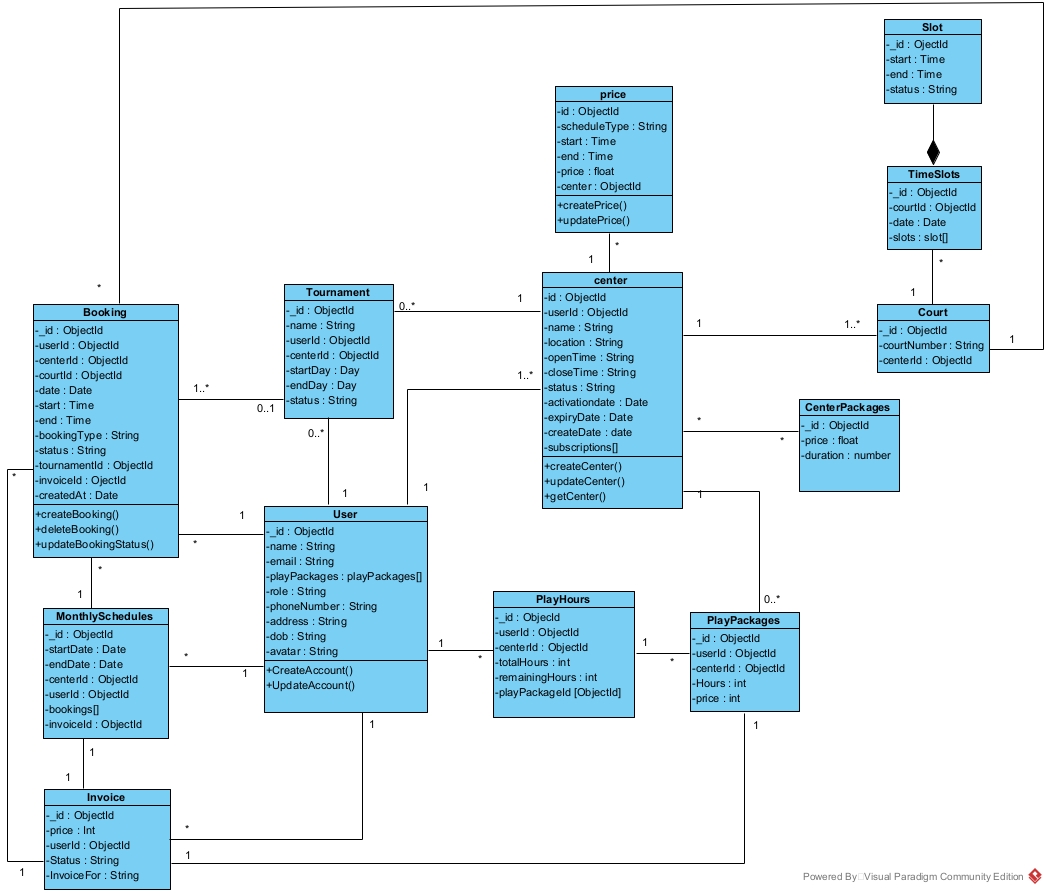


#### 1.3.2 Package Descriptions

|  |  |  |
| --- | --- | --- |
| No | Package | Description |
| 01 | dist | This directory typically contains the compiled output of the project. It is the distribution directory where the build process compiles the source code and places the final executable files or packages. |
| 02 | Node Module | This directory contains the external Node.js modules and dependencies that the project relies on. These modules are imported into the source code to provide additional functionality and features. |
| 03 | src | This is the main source directory containing the application’s source code. It includes several subdirectories, each serving a different purpose within the application. |
| 04 | config | This directory holds configuration files and settings for the application. It includes environment variables, configuration constants, and other settings that the application needs to run correctly. |
| 05 | controllers | This directory contains controller files, which manage the application's request handling logic. Controllers process incoming HTTP requests, interact with the models, and send responses back to the client. |
| 06 | models | This directory contains model files, which define the data structures and schemas used by the application. Models represent the core data entities and their relationships, and are often used to interact with the database. |
| 07 | repository | This directory contains route files, which define the application's routing logic. Routes map incoming HTTP requests to specific controller actions based on the request URL and method. |
| 08 | services | This directory holds service files, which contain business logic and operations that are reusable across different parts of the application. Services are used by controllers to perform complex operations and calculations. |
| 09 | utils | This directory contains utility files, which provide helper functions and utilities that are used throughout the application. These utilities can include common functions, data manipulation methods, and other reusable code snippets. |

## 2. Database Design

### 2.1Database diagram

**

### 2.2Table Descriptions

|  |  |  |
| --- | --- | --- |
| No | Table | Description |
| *01* | *Booking Collection* | *Description: This table records information about court bookings.*  *Fields:*  *\_id: ObjectId*  *userId: ObjectId (foreign key to User)*  *centerId: ObjectId (foreign key to Center)*  *courtId: ObjectId (foreign key to Court)*  *date: Date*  *start: Time*  *end: Time*  *bookingType: String*  *status: String*  *tournamentId: ObjectId (foreign key to Tournament)*  *invoiceId: ObjectId (foreign key to Invoice)*  *createdAt: Date*  *Methods:*  *createBooking()*  *deleteBooking()*  *updateBookingStatus()* |
| *02* | *MonthlySchedules Collection* | *Description: This table records the monthly schedules for users.*  *Fields:*  *\_id: ObjectId*  *startDate: Date*  *endDate: Date*  *centerId: ObjectId (foreign key to Center)*  *userId: ObjectId (foreign key to User)*  *bookingId: ObjectId (foreign key to Booking)*  *invoiceId: ObjectId (foreign key to Invoice)* |
| *03* | *Invoice Collection* | *Description: This table stores invoice information for transactions.*  *Fields:*  *\_id: ObjectId*  *status: String*  *price: Int*  *createdAt: Date*  *invoiceFor: String* |
| *04* | *Tournament Collection* | *Description: This table records information about tournaments.*  *Fields:*  *\_id: ObjectId*  *name: String*  *userId: ObjectId (foreign key to User)*  *centerId: ObjectId (foreign key to Center)*  *startDay: Date*  *endDay: Date*  *status: String* |
| *05* | *User Collection* | *Description: This table stores user information.*  *Fields:*  *\_id: ObjectId*  *name: String*  *email: String*  *playPackages: playPackages[] (foreign key to PlayPackages)*  *role: String*  *phoneNumber: String*  *address: String*  *dob: String*  *avatar: String*  *Methods:*  *CreateAccount()*  *UpdateAccount()* |
| *06* | *Center Collection* | *Description: This table stores information about sports centers.*  *Fields:*  *\_id: ObjectId*  *userId: ObjectId (foreign key to User)*  *name: String*  *location: String*  *openTime: String*  *closeTime: String*  *status: String*  *activationDate: Date*  *expiryDate: Date*  *createDate: Date*  *subscription[]: String*  *Methods:*  *createCenter()*  *updateCenter()*  *getCenter()* |
| *07* | *Court Collection* | *Description: This table records information about courts.*  *Fields:*  *\_id: ObjectId*  *centerId: ObjectId (foreign key to Center)*  *courtNumber: String* |
| *08* | *PlayPackages Collection* | *Description: This table stores information about play packages purchased by users.*  *Fields:*  *\_id: ObjectId*  *userId: ObjectId (foreign key to User)*  *centerId: ObjectId (foreign key to Center)*  *totalHours: Int*  *remainingHours: Int*  *purchasedAt: Date* |
| *09* | *CenterPackages Collection* | *Description: This table stores information about packages available at centers.*  *Fields:*  *\_id: ObjectId*  *price: Float*  *duration: Number* |
| *10* | *Price Collection* | *Description: This table records pricing information for different schedules.*  *Fields:*  *\_id: ObjectId*  *scheduleType: String*  *start: Time*  *end: Time*  *price: Float*  *center: ObjectId (foreign key to Center)*  *Methods:*  *createPrice()*  *updatePrice()* |
| *11* | *TimeSlots Collection* | *Description: This table records available time slots for courts.*  *Fields:*  *\_id: ObjectId*  *courtId: ObjectId (foreign key to Court)*  *date: Date*  *slots: slot[]* |
| *12* | *Slot Collection* | *Description: This table represents individual time slots.*  *Fields:*  *\_id: ObjectId*  *start: Time*  *end: Time*  *status: String* |

## 3. Detailed Design

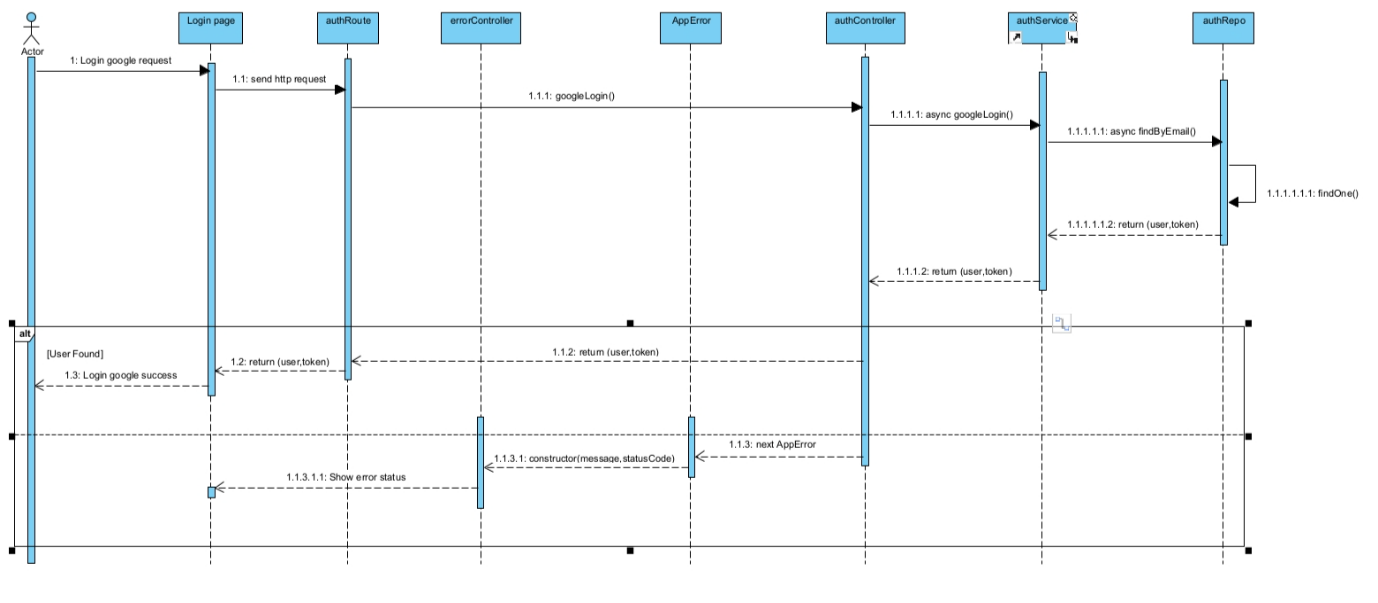
### 3.1 *SignUp*

#### 3.1.1 SignUp Sequence

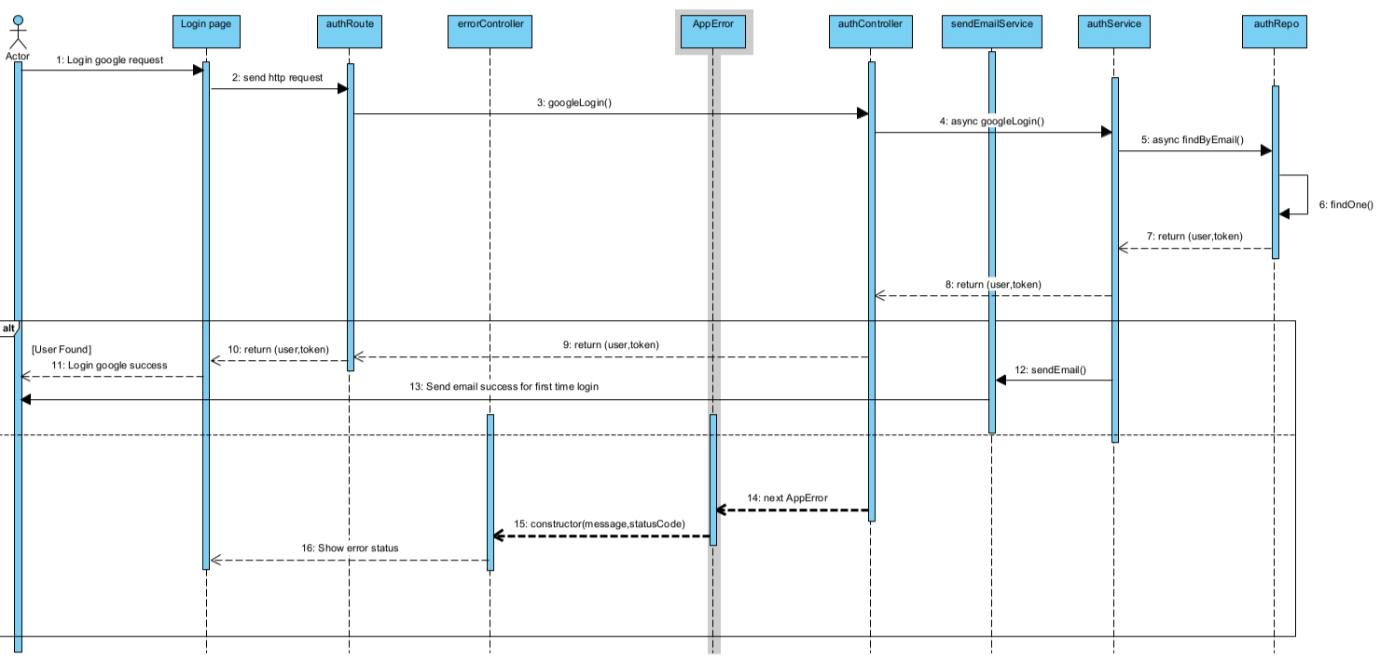
#### 

### 3.2 Login google

#### 3.2.1 Login google sequence

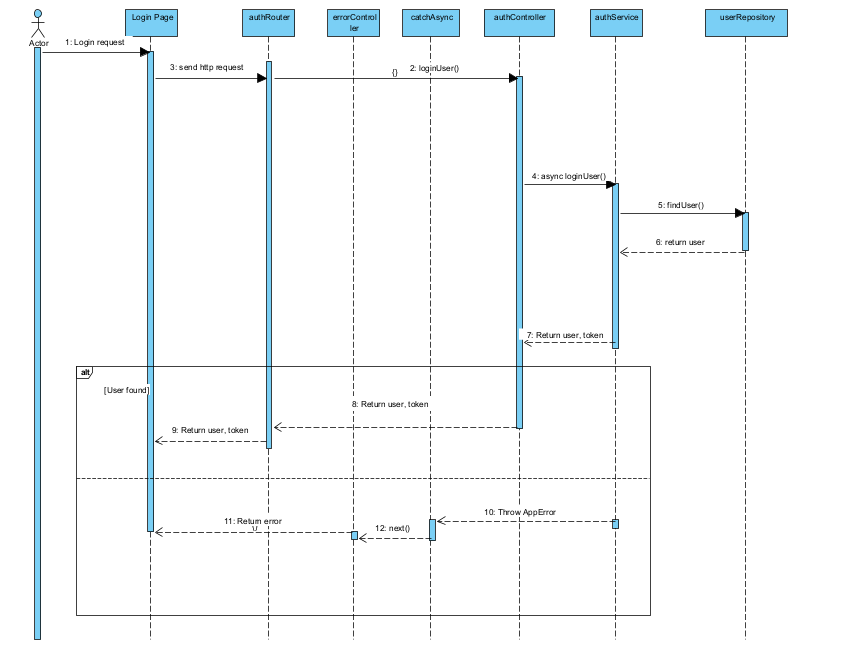


3.2.2 Login google have send mail sequence



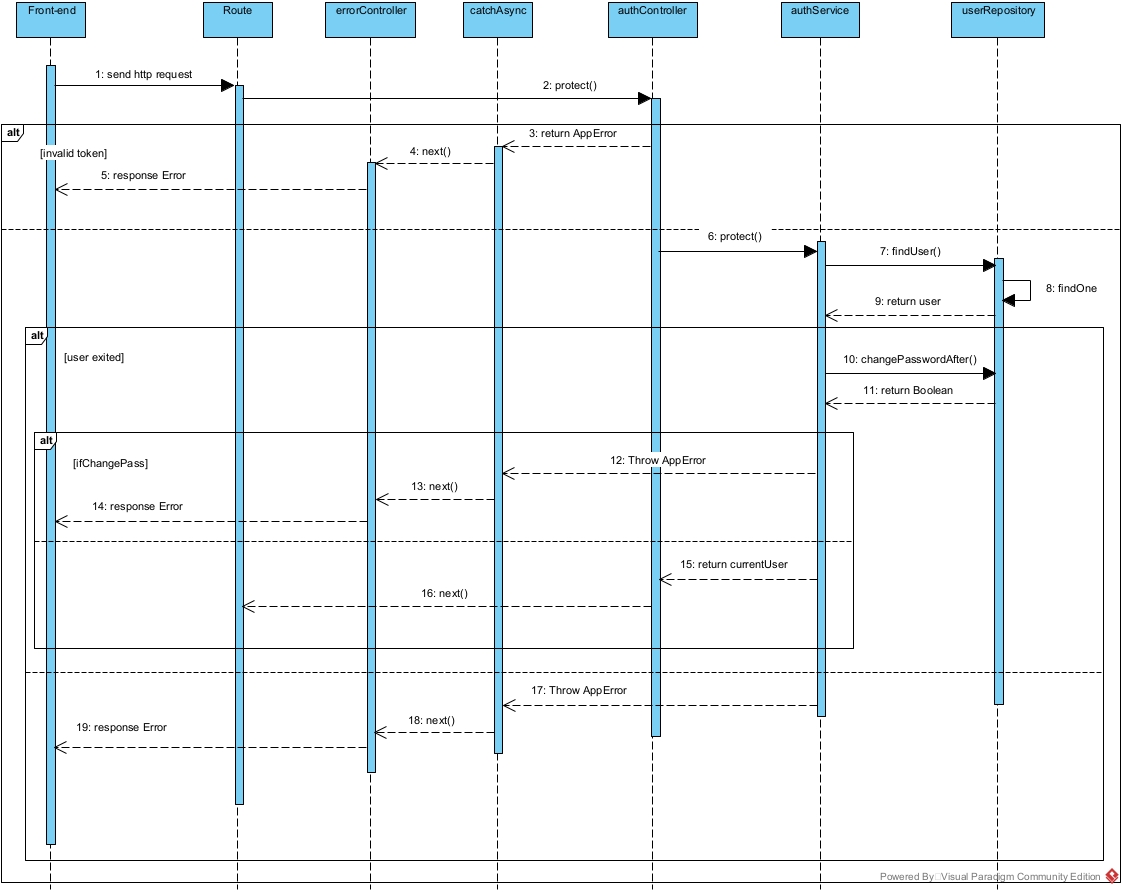
### 3.3 Login

#### 3.3.1 Login sequence

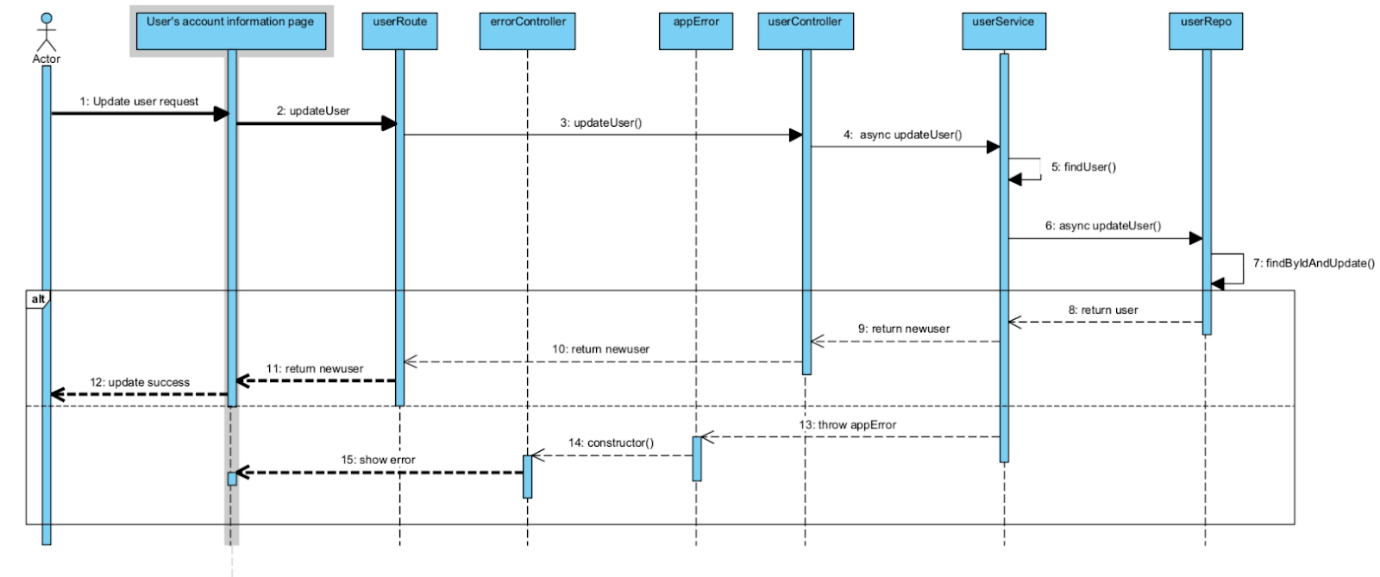


### 3.4 Authenticate

#### 3.4.1 Authenticate sequence



3.5 Update

3.5.1 Update user sequence  


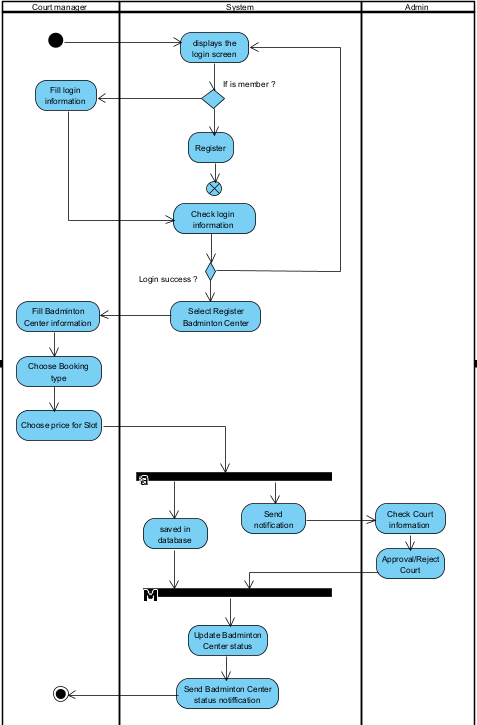
### 3.5 View Detail

#### 3.5.1 View Detail

### 3.6 Center Management

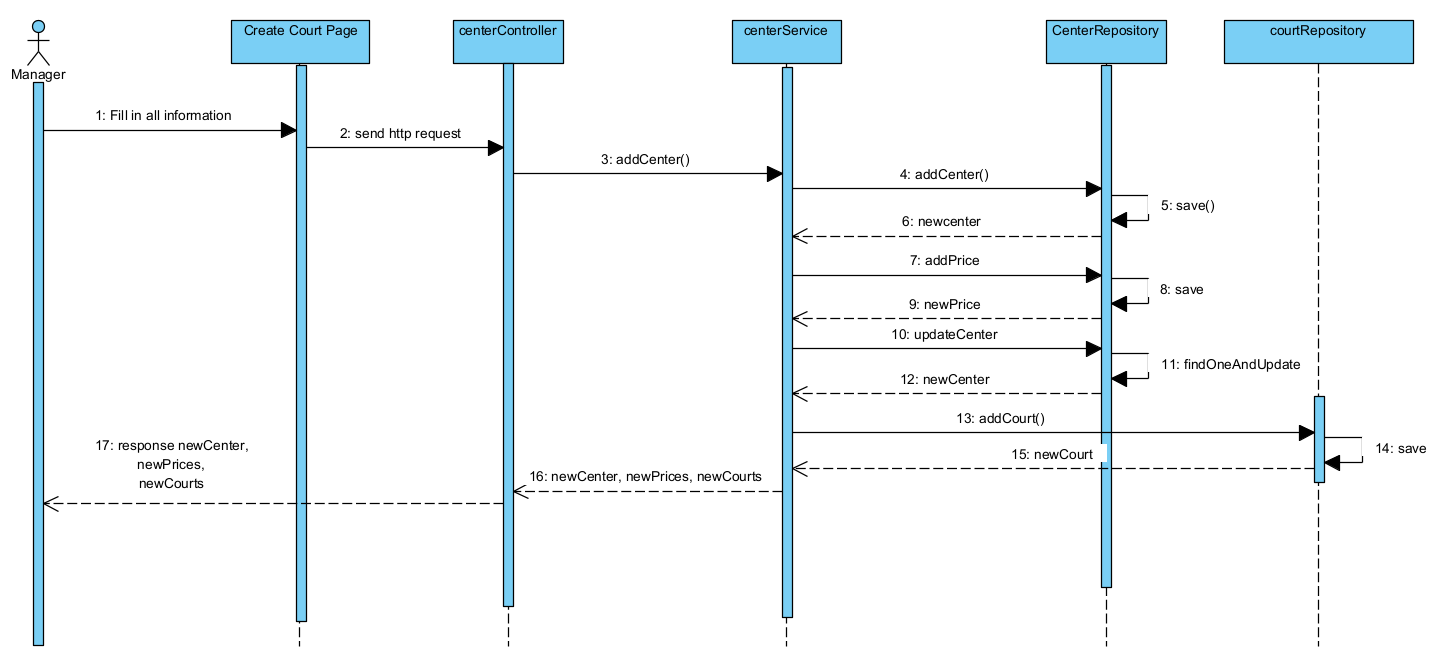
#### 3.6.1 Activity Diagram

##### 3.6.1.1 Create Center Activity Diagram



#### 3.6.2 Sequence

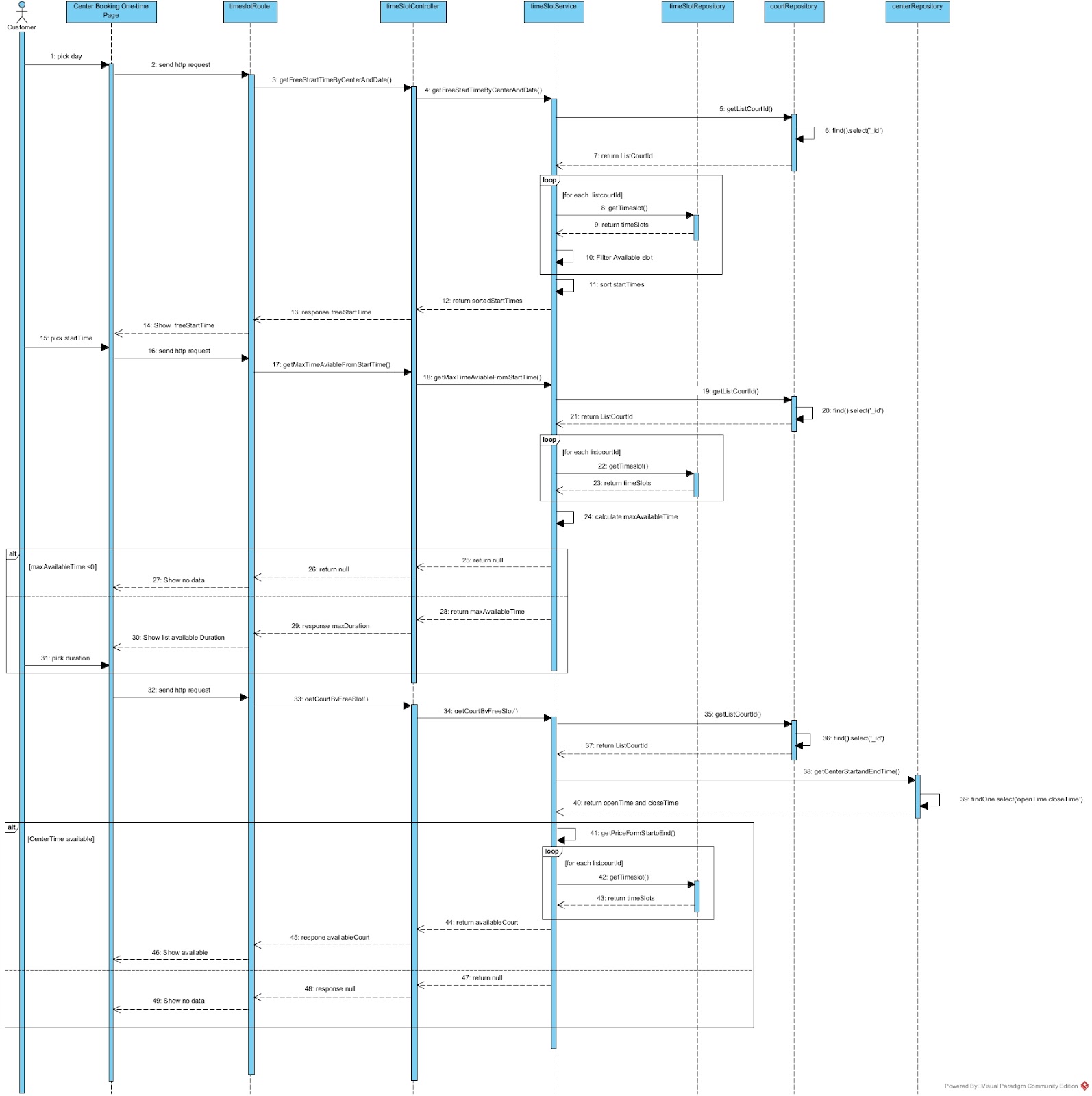
##### 3.6.2.1 Create Center Sequence



##### 3.6.2.2 Update Center Infor Sequence

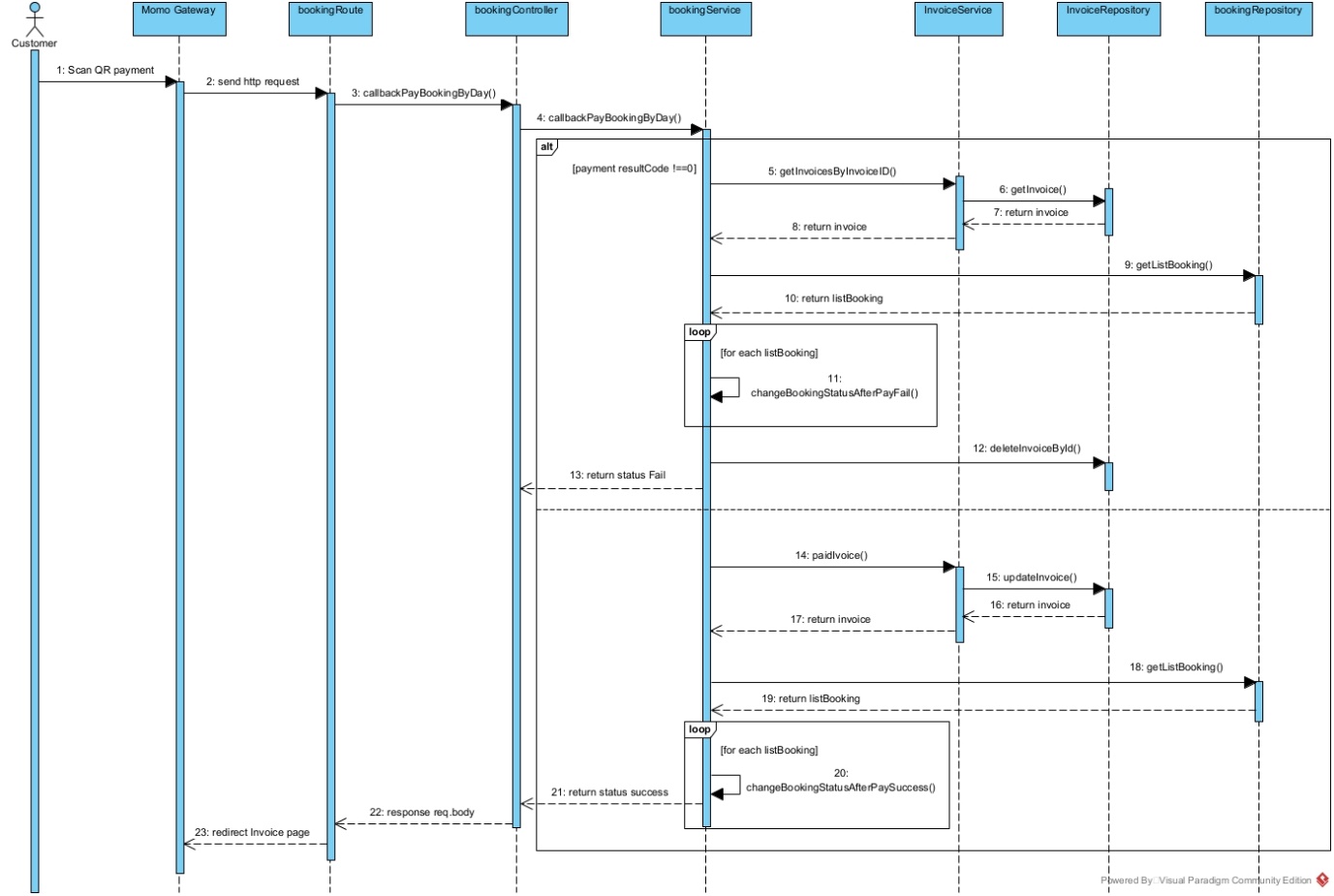
### 3.7 Book by date

#### 3.7.1 Pick available court



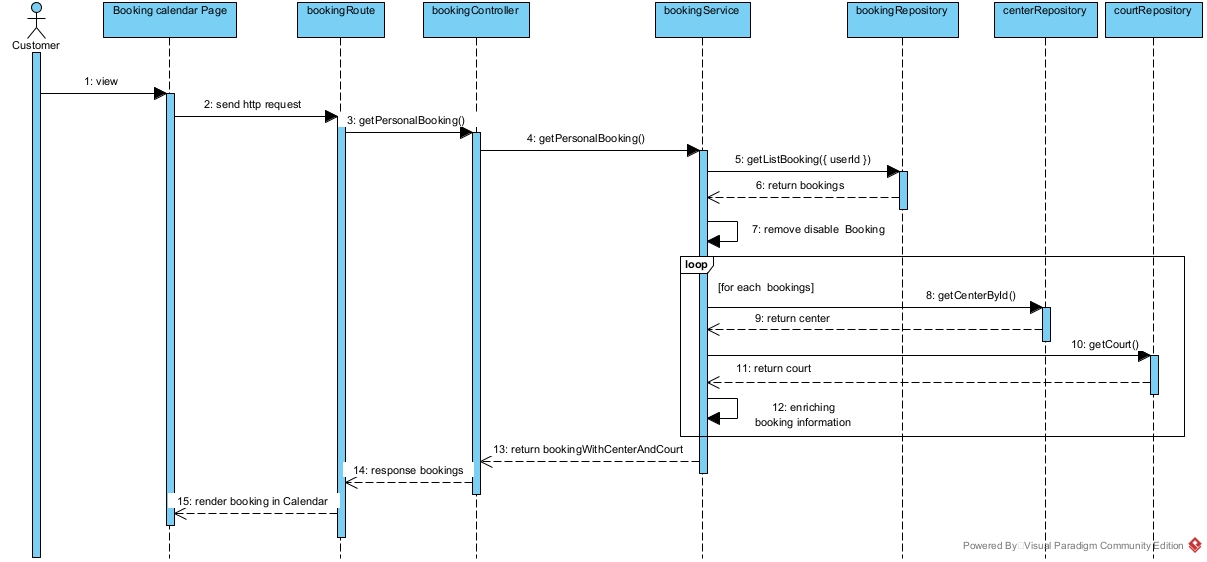
#### 3.7.2 Create booking

3.7.3 payment processing



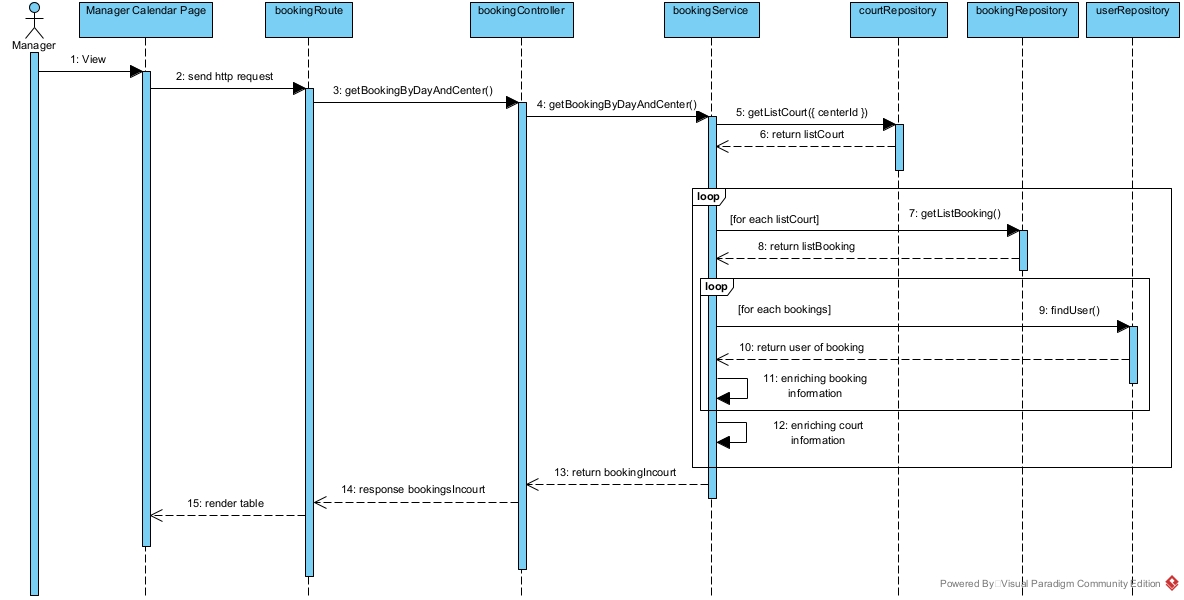
### 3.8 Personal Booking

#### 3.8.1 View Personal Booking in schedule

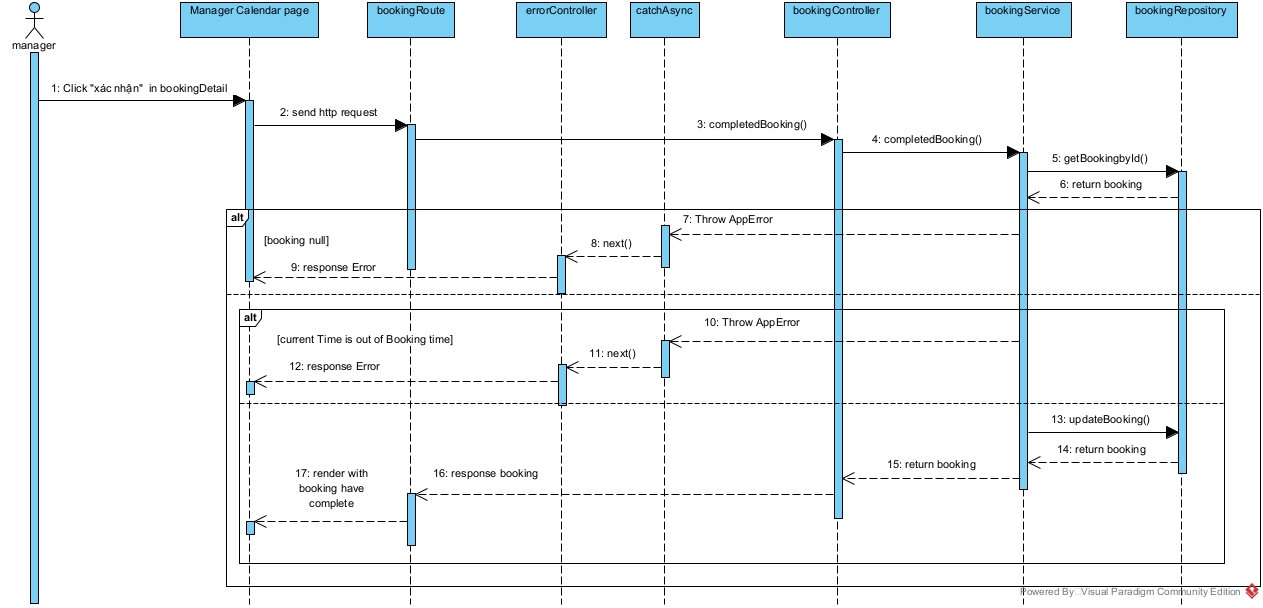


### 3.9 Manage Booking

#### 3.9.1 View Customer Booking in Table



3.9.2 Complete Booking for customer



### 3.10 Information User

#### 3.10.1 View information

