Project Part 2

1. **Team**: Dixit Patel
   Swathi Upadhyaya
   Yuvraj Arora

2. **Title**: Rec-Center Management System

3. **Project Summary:** The aim of this project is to build a robust Web Application for a Rec-Center Management System. The system can be used by customers to avail the services of the Rec-Center like scheduling an appointment with a trainer, reserving a Sports Court/ Gym Equipment/ Parking Lot/ Renting a Locker/ Participating in an Event/ Subscribing for Memberships and Sports Clubs. The website will be accessible to the users, Sports administrator, the trainer and the maintenance staff. The systems has two user profiles – Student Profile and the Guest Profile.

4. **Project Requirements:**

User Requirements		
ID	**Description**	**Priority**
US-01	As a customer, I want to request appointments with trainers of different sports in Rec-Center Management System.	High
US-02	As a customer, I want to reserve Sports Court in the Rec-Center Management System so that it is available whenever I want to use it.	High
US-03	As a customer, I want to cancel a reserved Sports Court in the Rec-Center Management System if I am unable to make it at that schedule.	Medium
US-04	As a customer, I want to register for the events hosted in the Rec-Center Management System.	High
US-05	As a customer, I want to browse through the list of upcoming events in the Rec-Center Management System.	High

US-06	As a customer, I want to view details of any of the upcoming events in the browse list so that I can check the event details like, Schedule and Venue in the Rec-Center Management System.	High
US-07	As a customer, I want to view the list of upcoming events that I have registered for in the Rec-Center Management System to keep track of the events.	Medium
US-08	As a Trainer, I want to accept an appointment requested by the customer.	High
US-09	As a Trainer, I want to reject an appointment requested by the customer.	Medium
US-10	As a trainer, I should be able to view the appointments requested by the customers.	High
US-11	As an Administrator, I want to organize events in the Rec-Center Management System. I want to decide on event type, event commencement date and time, event finish date and time and the event fee for a customer to participate.	High
US-12	As an Administrator, I want to make updates in the organized event before it's commencement.	High
US-13	As an Administrator, I want to view the list of upcoming events.	Medium

Business Requirements		
ID	**Description**	**Priority**
BR-01	A Student Customer should create an account with the Rec-Center Management System using his University Email ID. This holds good for logging into the system as well. Ex. Alex.Parish@colorado.edu	High
BR-02	A customer can request for an appointment with only a single Trainer in a day.	High
BR-03	A customer can reserve only a single Sports Court in a day.	Medium
BR-04	A customer can reserve a court for only a single time slot.	Medium
BR-05	The password field should have a length of at least 8 characters.	High

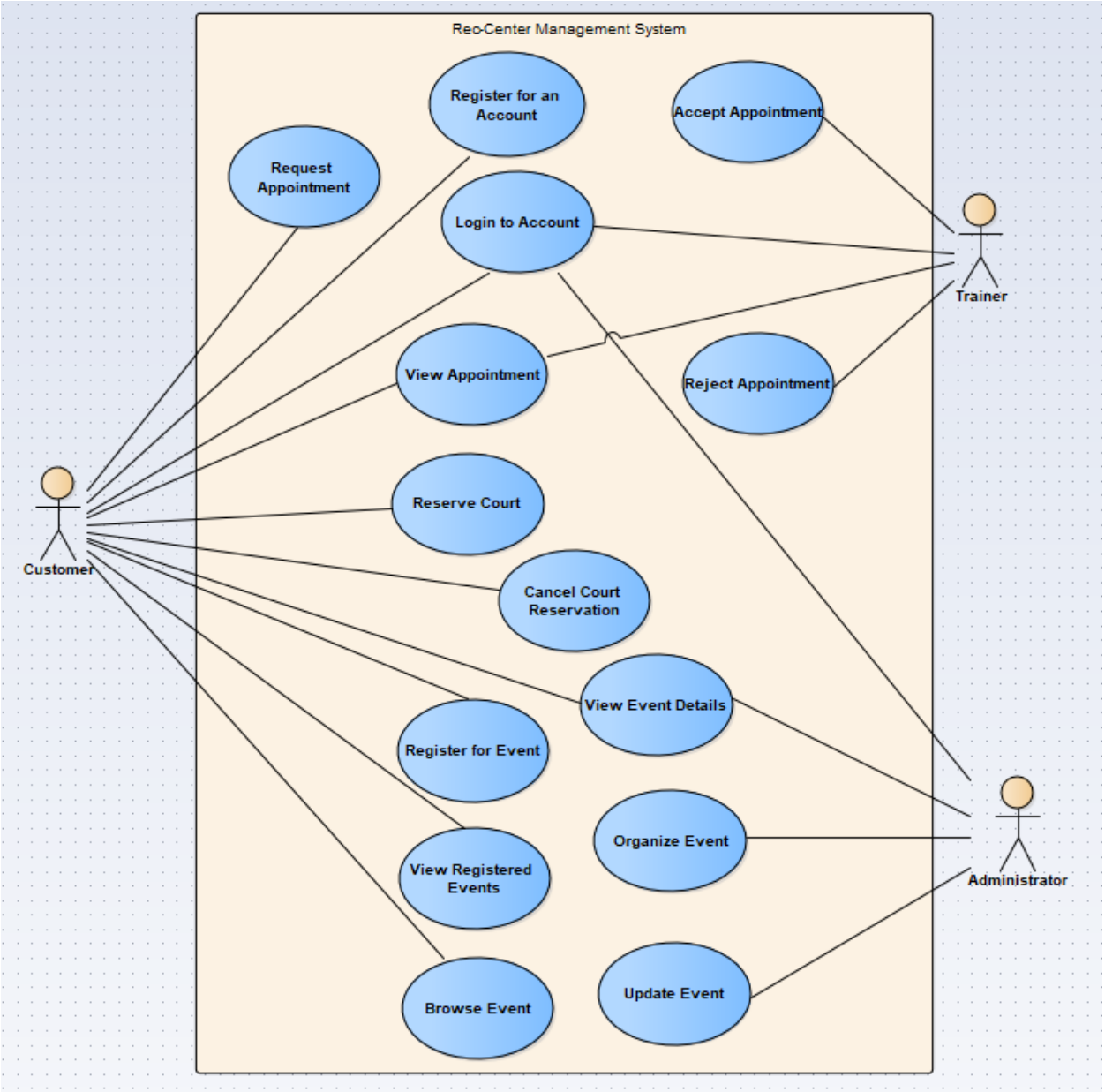
Functional Requirements		
ID	**Description**	**Priority**
FR-01	The time slot allocated to a trainer is blocked when a customer has already requested for appointment in that time slot.	High
FR-02	The time slot for a court is blocked when a customer has already reserved the court in that time slot.	Medium

FR-03	A customer needs to login to request for an appointment.	High
FR-04	A customer needs to login to reserve a court.	Medium

Non-Functional Requirements		
ID	**Description**	**Priority**
NFR-01	The UML Diagrams conform to Enterprise Architecture Standards. UML 2.0 version is used. (Maintainability)	High
NFR-02	The user data is written into the relational database during the transaction, so recovery from failure is easy. (Recovery)	High
NFR-03	User-friendly, Interactive and Simple GUI for users to navigate from one screen to the next. (Usability)	Medium
NFR-04	The system is available 24 hours. (Availability)	High
NFR-05	The information access levels are different for Customers, Staff and the Administrator. (Security)	High
NFR-06	The system takes care of synchronization when multiple users try to reserve Court/Trainer simultaneously. (Reliability)	Medium
NFR-07	User/Event Response Time is Minimum. (Performance)	Medium
NFR-08	Relational Database shall be used in the backend. Consistency and Integrity of data is guaranteed. (Integrity)	High

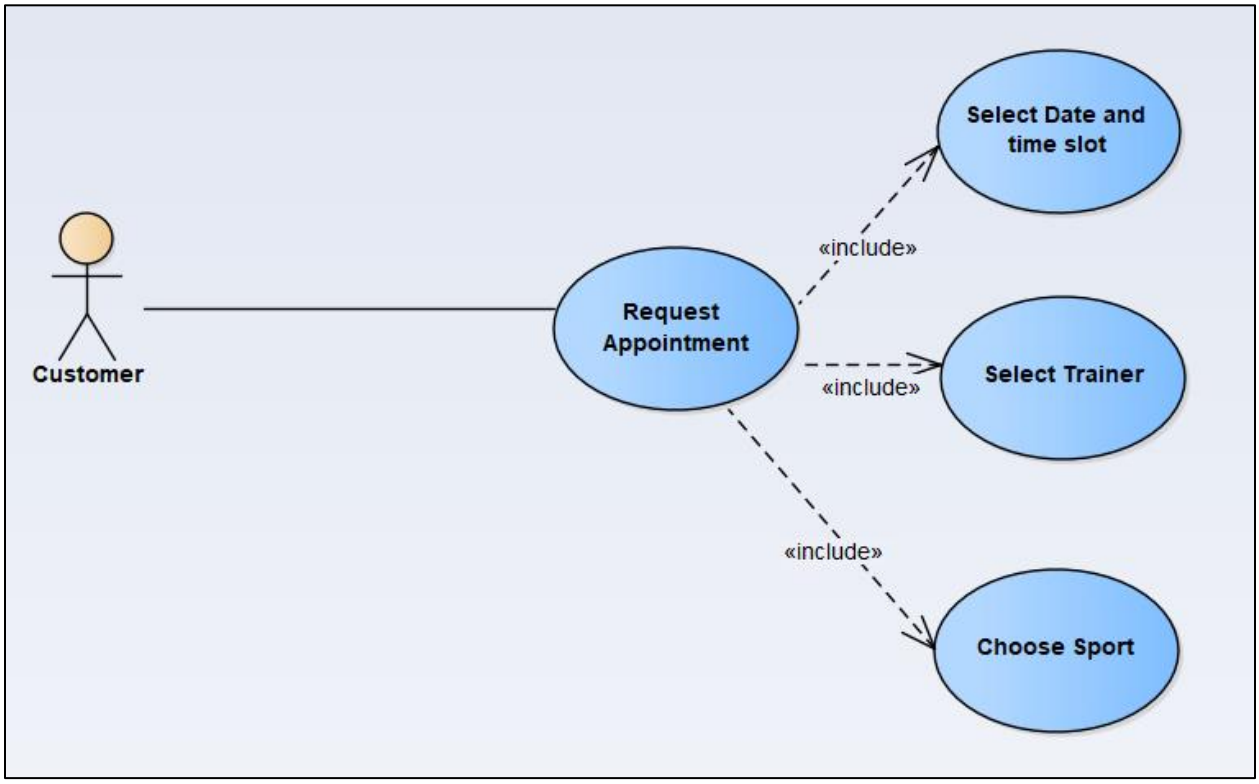
5. **Use Cases:**

a. Use Case Overview:

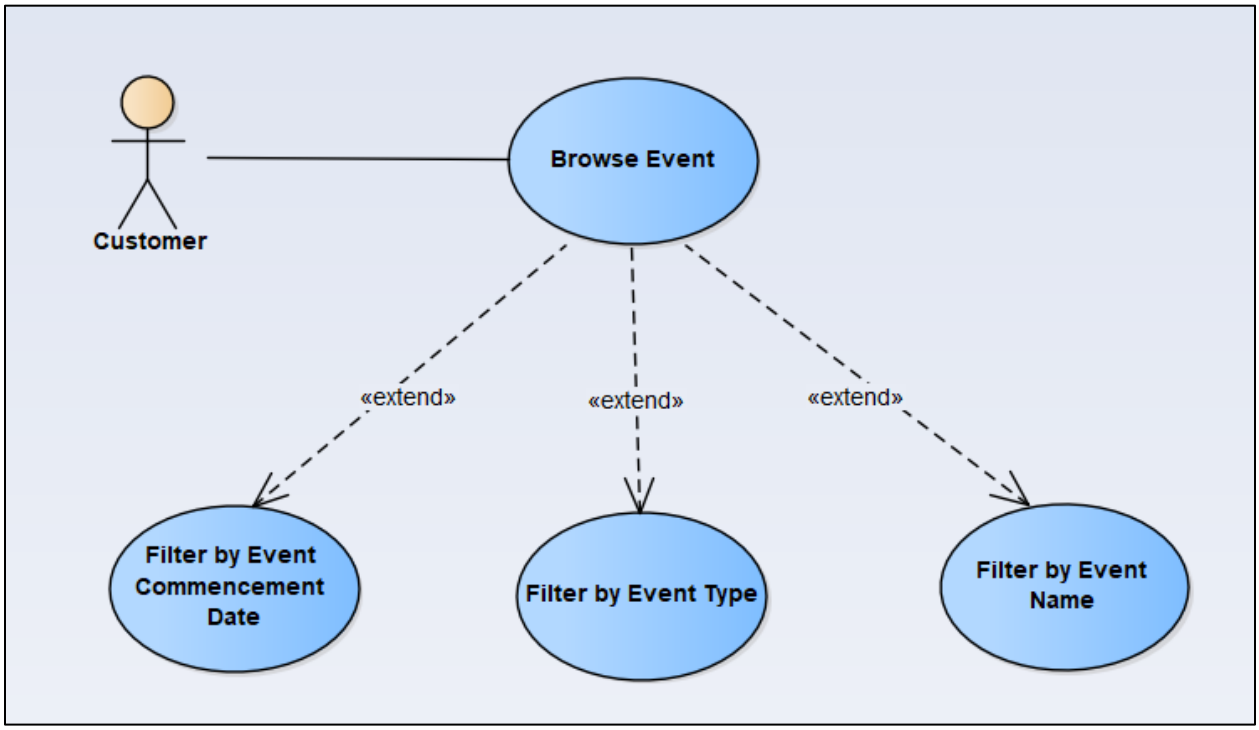


Sub-Diagrams:

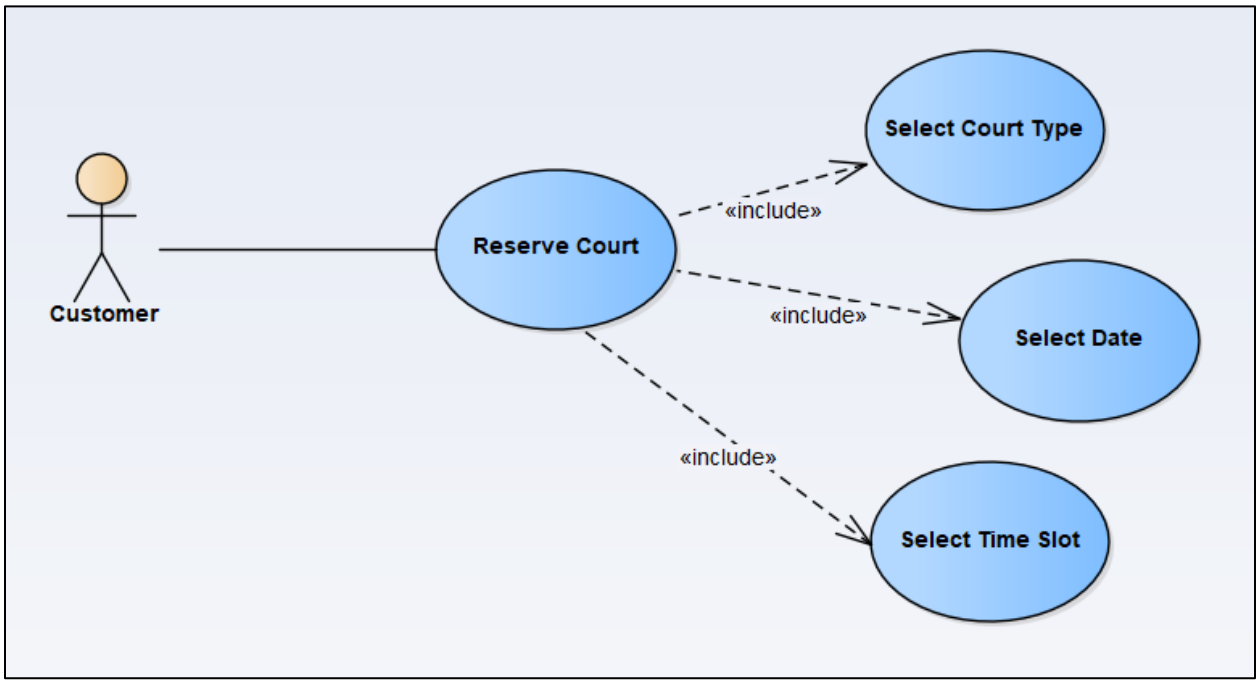
Request Appointment Sub-Diagram



Browse Event Sub-Diagram



Reserve Court Sub-Diagram



b. Use Case Document:

1. Customer should be able to reserve any sports court in the Rec-Center Management System.

Use Case ID:	UC-01
Use Case Name:	Reserve Court
Requirement ID:	US-02

Actor:	Customer		
Description:	As a customer, I want to reserve any of the Sports Court so that it is available when I want to use it in the Rec-Center Management System.		
Preconditions:	1. The web application is up and running 2. The customer is logged in to the system 3. The Guest Customer should be enrolled in any of the membership plans.		
Postconditions:	The Sports Court Reservation by the customer is successful.		
Priority:	High Priority		
Frequency of Use:	High		
Normal Course of Events:		User Action	System Response
	1.	User clicks on reserve court in the dashboard.	
	2.	User clicks on Make New Reservation button in the dashboard.	System prompts the courts available and the calendar for the user to select date.
	3.	User selects the required sports court.	
	4.	User selects the date from the Calendar.	System checks for the available slots for that date.
	5.		System displays the available slots and prompts user to select one of the slots.
	6.	User selects time slot.	
	7.	User clicks on confirm button.	System confirms the reservation and displays reservation successful message.
Alternative Courses:			
Exceptions:	7. Reservation failed because some other user selected the same time slot simultaneously. System displays the reservation failure message. 5. No time slot is available for the selected court and selected date. System displays the no slots available message.		

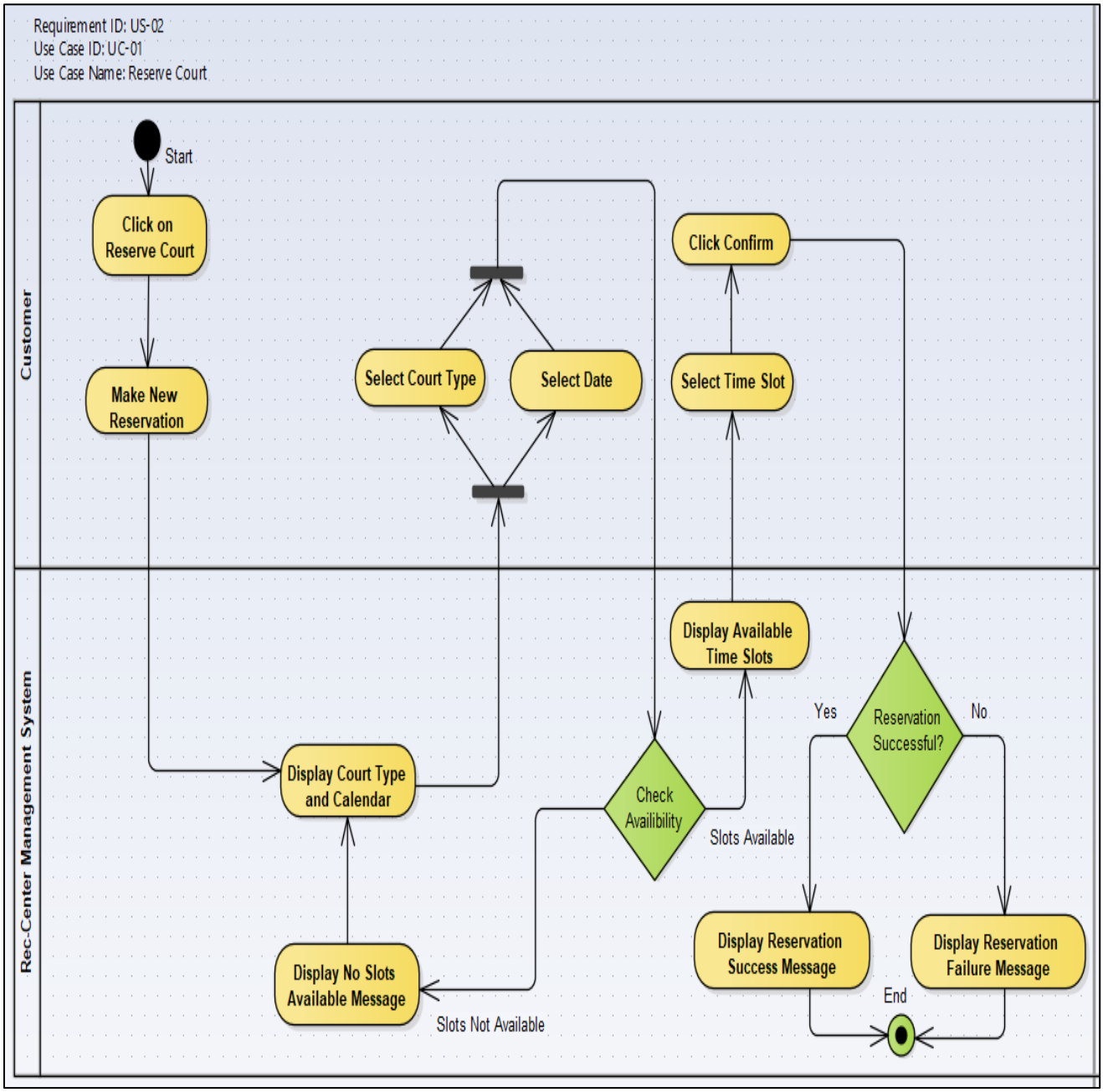
2. Customer should be able to participate in any of the events in the Rec-Center Management System.

Use Case ID:	UC-02
Use Case Name:	Register in a Rec-center event
Requirement ID:	US-04

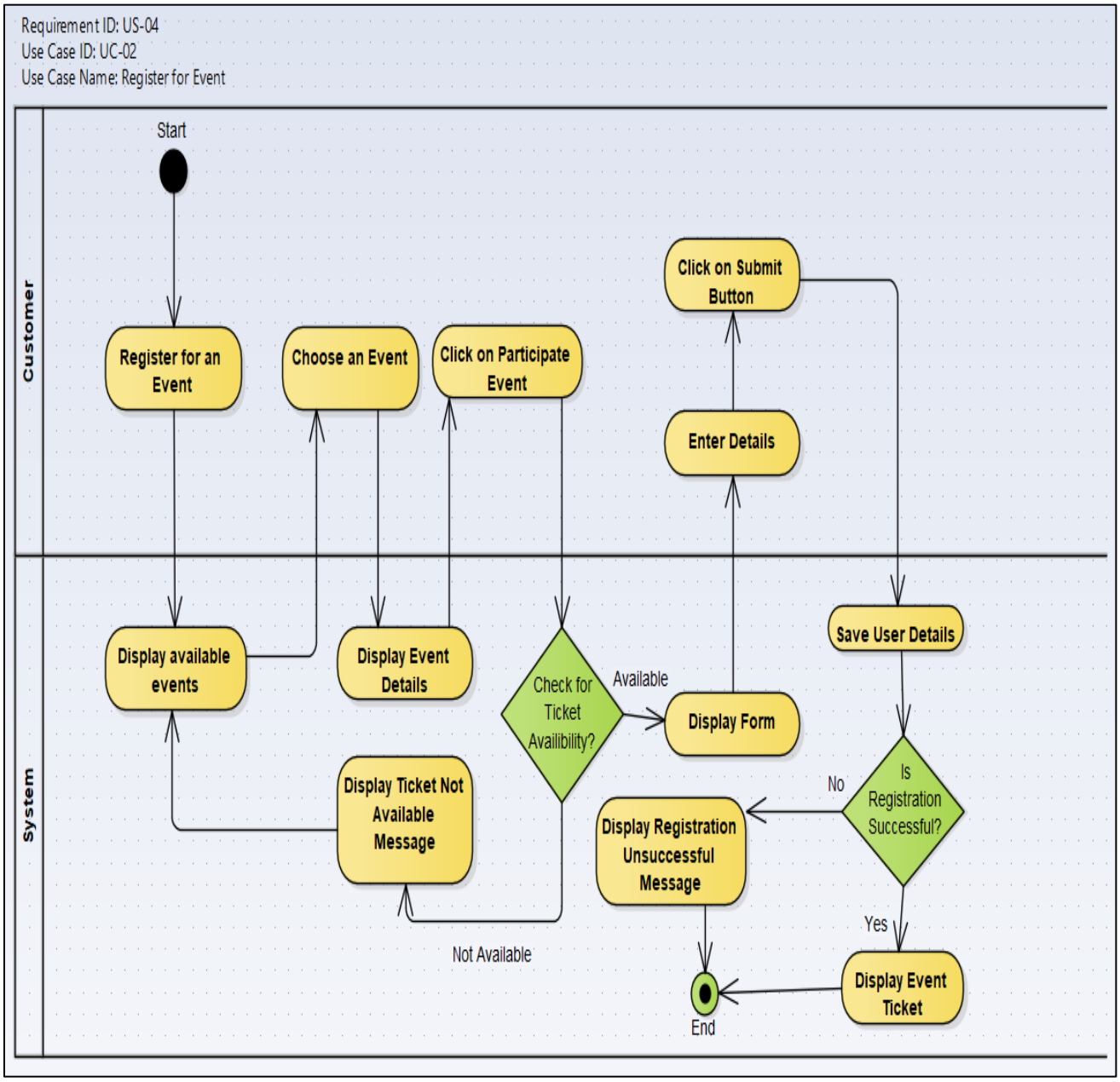
Actor:	Customer		
Description:	As a customer, I want to participate in any of the events which is organized by the Rec Center		
Preconditions:	4. The web application is up and running 5. The customer is logged in to the system 6. The Guest Customer should be enrolled in any of the membership plans.		
Postconditions:	The Sports Court Reservation by the customer is successful.		
Priority:	High Priority		
Frequency of Use:	High		
Normal Course of Events:		User Action	System Response
	1.	User clicks on Events button in the dashboard.	System displays Event List.
	2.	User clicks on a particular event in the dashboard.	System displays the details of that particular event.
	3.	User clicks the Participate button.	System checks for the Ticket Availability.
	4.		System displays a form for the user to fill in details.
	5.	User enters his details in the event form.	
	6.	User clicks on Submit Button.	Systems confirms user registration for the event.
Alternative Courses:			
Exceptions:	4. No tickets available for the user. System displays a No Tickets available message and redirects to display event list page. 6. System displays a error message if the registration is not successful.		

6. Activity Diagram :

The below activity diagram shows the Sports Court Reservation by the customer.

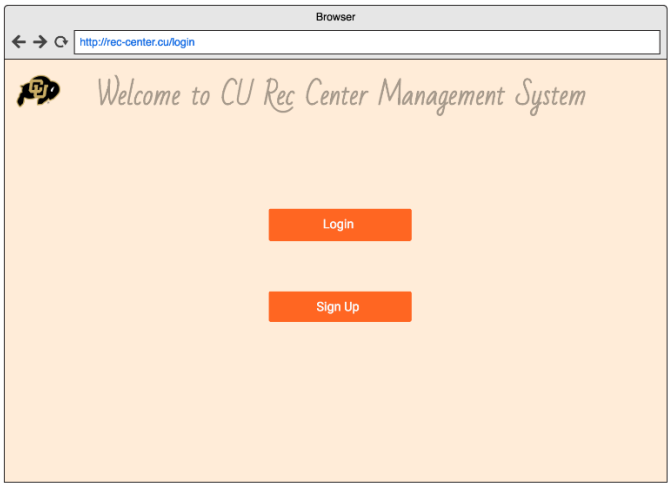


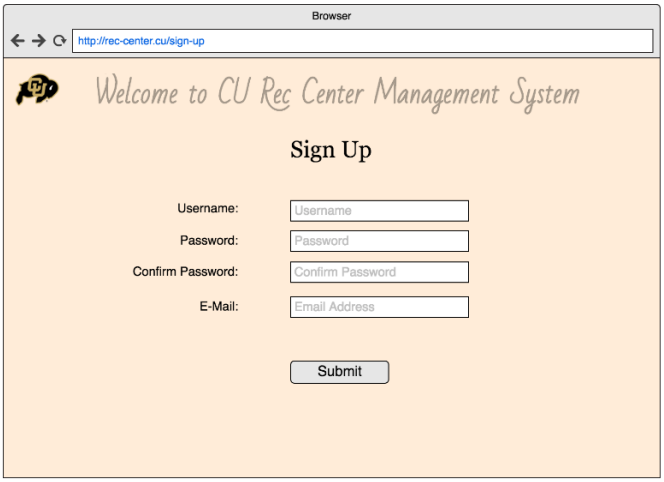
The below activity diagram shows the Event Registration for the customer.



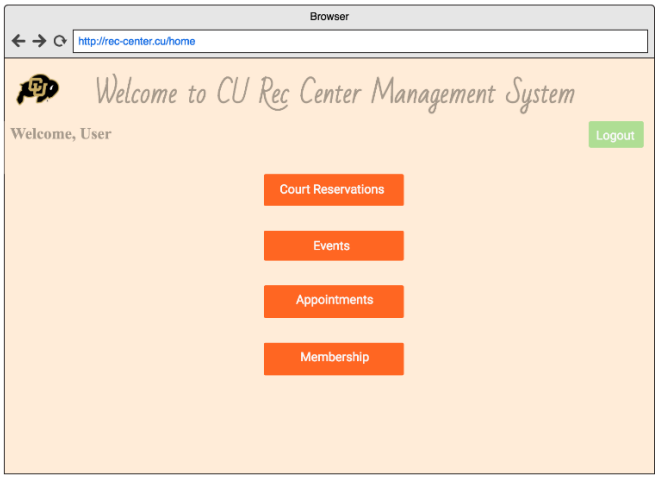
7. UI Mockup:

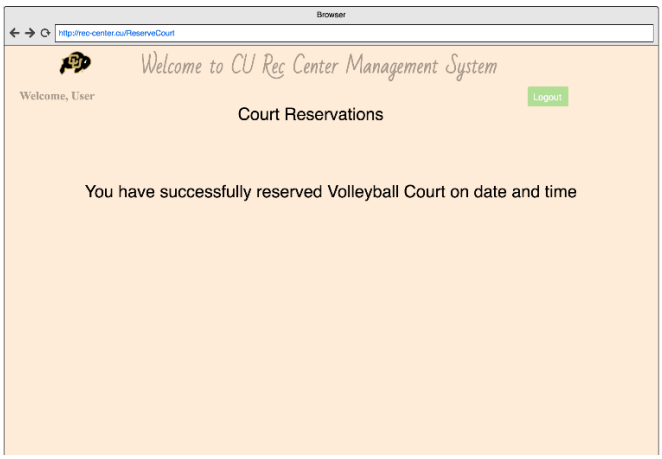
UI for signing up and logging into the system for a customer is shown below –



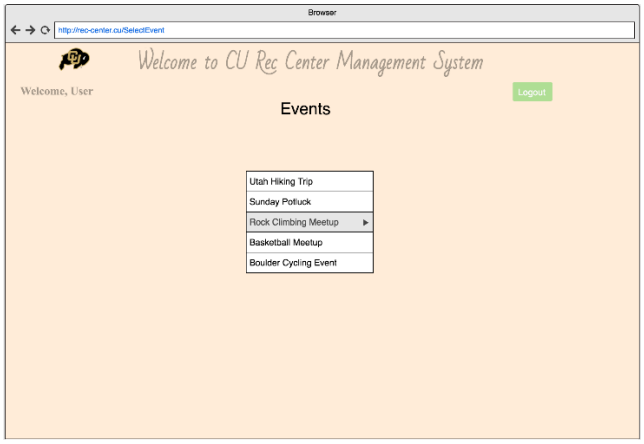


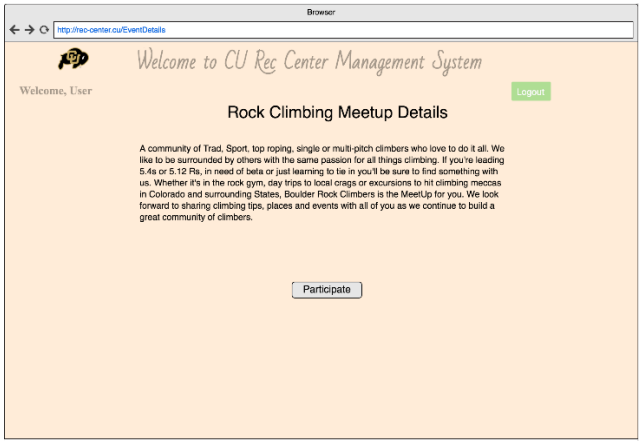
UI for Reservation of court is shown below –

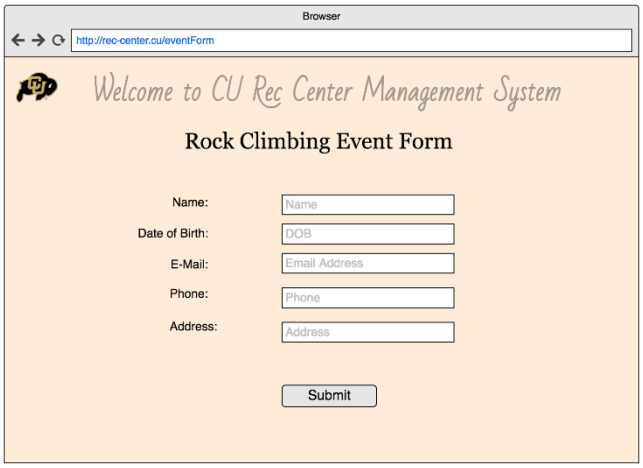


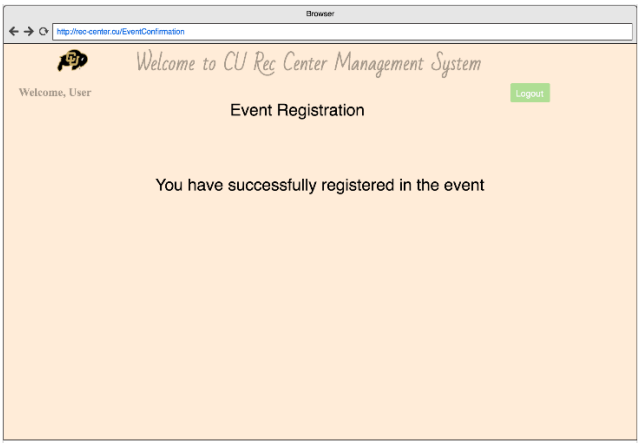


UI for registering to an event is shown below –





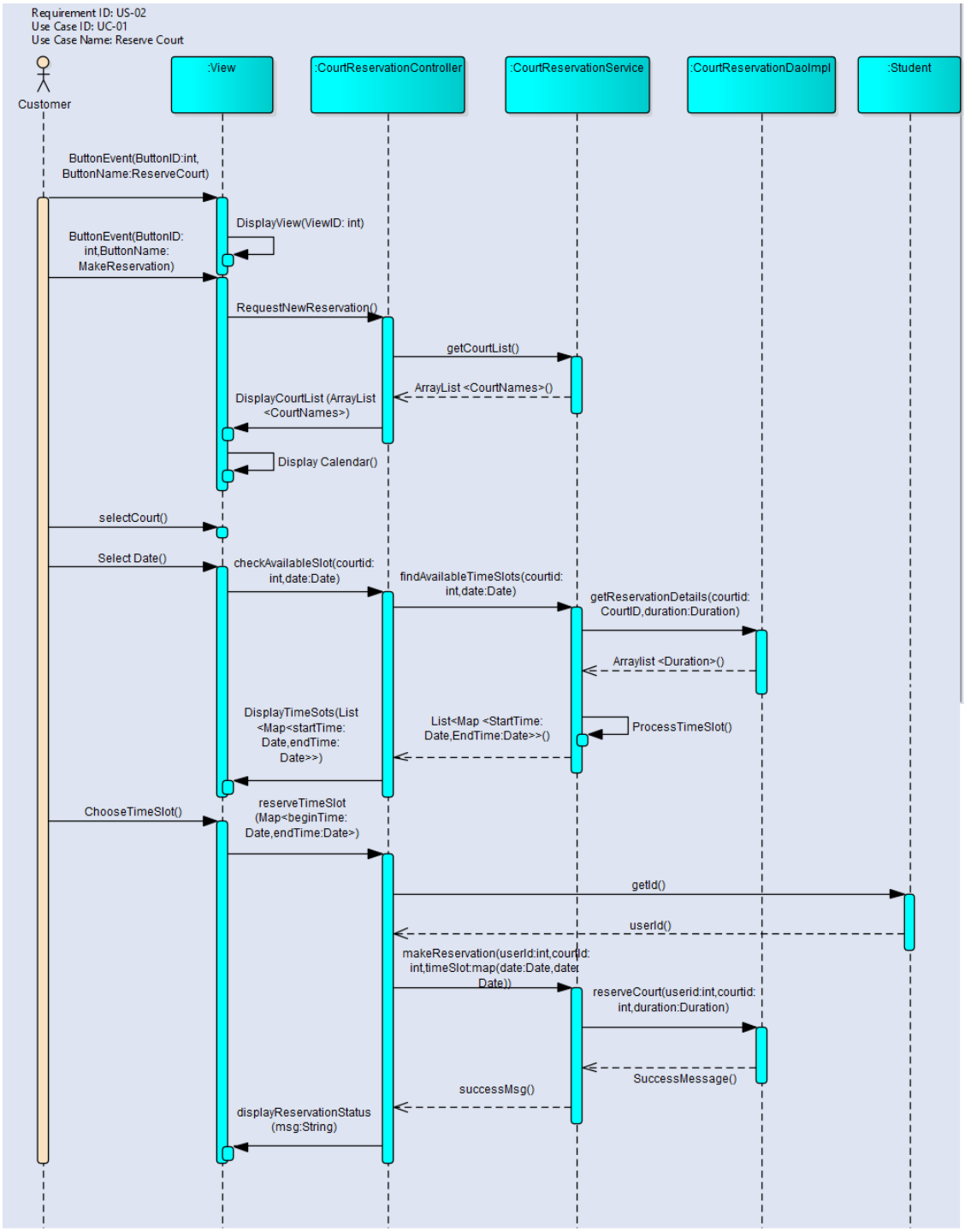




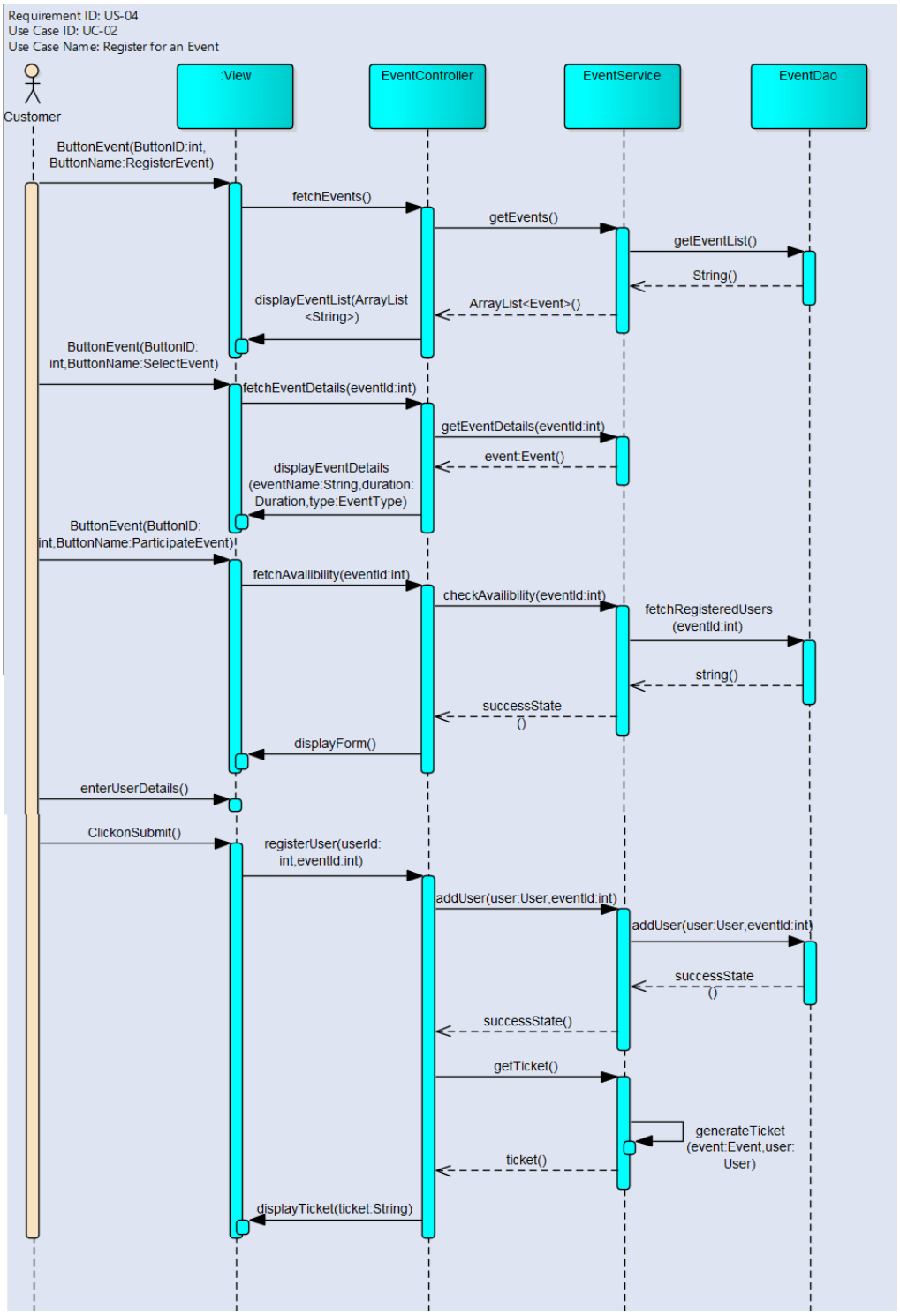
Data Storage : We propose to use MySQL Relational Database Systems for storing the data. User data, Court Data, Event Data shall be stored in the database.

8. Sequence Diagram:

The below sequence diagram shows the court reservation for the customer.



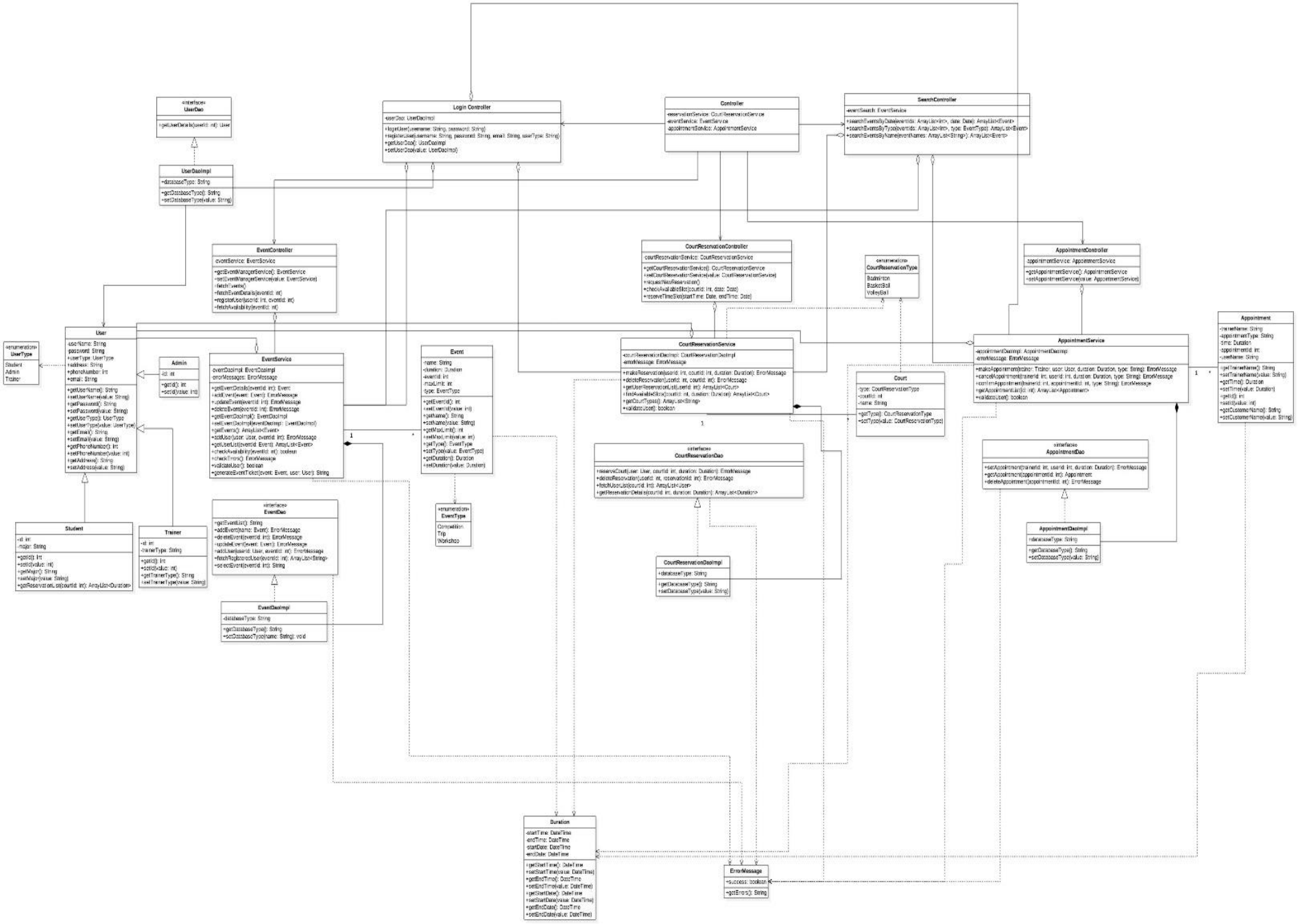
The below sequence diagram shows event registration for the customer.



9. Class Diagram:

The below Class Diagram shows the classes, their methods and attributes, and their visibility modifier in the Rec-Center Management System. It also depicts the relationships between the classes - Generalization, Association (Aggregation and Association) among classes with multiplicity and dependency relations.

Note: For better resolution, the class diagram has also been submitted as a separated image file in the github by name "**37_RecCenter_Management_System_Part2_ClassDiagram.jpg**".


«interface»
UserDao
+getUserDetails(userId: int): User
Login Controller
-userDao: UserDaoImpl
+loginUser(username: String, password: String)
+registerUser(username: String, password: String, email: String, userType: String)
+getUserDao(): UserDaoImpl
+setUserDao(value: UserDaoImpl)
Controller
-reservationService: CourtReservationService
-eventService: EventService
-appointmentService: AppointmentService
SearchController
-eventSearch: EventService
+searchEventsByDate(eventIds: ArrayList<Int>, date: Date): ArrayList<Event>
+searchEventsByType(eventIds: ArrayList<Int>, type: EventType): ArrayList<Event>
+searchEventsByName(eventNames: ArrayList<String>): ArrayList<Event>
UserDaoImpl
-databaseType: String
+getDatabaseType(): String
+setDatabaseType(value: String)
EventController
-eventService: EventService
+getEventManagerService(): EventService
+setEventManagerService(value: EventService)
-fetchEvents()
-fetchEventDetails(eventId: int)
+registerUser(userId: Int, eventId: Int)
-fetchAvailability(eventId: int)
CourtReservationController
-courtReservationService: CourtReservationService
+getCourtReservationService(): CourtReservationService
+setCourtReservationService(value: CourtReservationService)
+requestNewReservation()
+checkAvailableSlot(courtId: int, date: Date)
+reserveTimeSlot(startTime: Date, endTime: Date)
«enumeration»
CourtReservationType
Badminton
BasketBall
VolleyBall
AppointmentController
-appointmentService: AppointmentService
+getAppointmentService(): AppointmentService
+setAppointmentService(value: AppointmentService)
User
-userName: String
-password: String
+userType: UserType
+address: String
+phoneNumber: Int
+email: String
+getUserName(): String
+setUserName(value: String)
+getPassword(): String
+setPassword(value: String)
+getUserType(): UserType
+setUserType(value: UserType)
+getEmail(): String
+setEmail(value: String)
+getPhoneNumber(): Int
+setPhoneNumber(value: int)
+getAddress(): String
+setAddress(value: String)
«enumeration»
UserType
Student
Admin
Trainer
Admin
-id: int
+getId(): int
+setId(value: int)
EventService
-eventDaoImpl: EventDaoImpl
-errorMessages: ErrorMessage
+getEventDetails(eventId: int): Event
+addUserEvent(event: Event): ErrorMessage
+updateEvent(eventId: Int): ErrorMessage
+deleteEvent(eventId: Int): ErrorMessage
+getEventDaoImpl(): EventDaoImpl
+setEventDaoImpl(eventDaoImpl: EventDaoImpl)
+getEvents(): ArrayList<Event>
+addUser(user: User, eventId: Int): ErrorMessage
+getUserList(eventId: Event): ArrayList<User>
-checkAvailability(eventId: int): boolean
+checkErrors(): ErrorMessage
+validateUser(): boolean
+generateEventTicket(event: Event, user: User): String
Event
-name: String
-duration: Duration
-eventId: int
-maxLimit: int
-type: EventType
+getEventId(): int
+setEventId(value: int)
+getName(): String
+setName(value: String)
+getMaxLimit(): int
+setMaxLimit(value: int)
+getType(): EventType
+setType(value: EventType)
+getDuration(): Duration
+setDuration(value: Duration)
«enumeration»
EventType
Competition
Trip
Workshop
CourtReservationService
-courtReservationDaoImpl: CourtReservationDaoImpl
-errorMessage: ErrorMessage
+makeReservation(userId: Int, courtId: Int, duration: Duration(): ErrorMessage
+deleteReservation(userId: int, courtId: int): ErrorMessage
+getUserReservationList(userId: int): ArrayList<Court>
+findAvailableTime(courtId: int, duration: Duration): ArrayList<Court>
+getCourtTypes(): ArrayList<String>
+validateUser(): boolean
Court
-type: CourtReservationType
-courtId: int
-name: String
+getType(): CourtReservationType
+setType(value: CourtReservationType)
AppointmentService
-appointmentDaoImpl: AppointmentDaoImpl
-errorMessage: ErrorMessage
+makeAppointment(trainer: Trainer, user: User, duration: Duration, type: String): ErrorMessage
+cancelAppointment(trainerId: int, userId: int, duration: Duration, type: String): ErrorMessage
+confirmAppointment(trainerId: int, appointmentId: int, type: String): ErrorMessage
+getAppointmentList(id: int): ArrayList<Appointment>
+validateUser(): boolean
Appointment
-trainerName: String
-appointmentType: String
-time: Duration
-appointmentId: int
-userName: String
+getTrainerName(): String
+setTrainerName(value: String)
+getTime(): Duration
+setTime(value: Duration)
+getId(): int
+setId(value: int)
+getCustomerName(): String
+setCustomerName(value: String)
1
*
Student
-id: int
-major: String
+getId(): Int
+setId(value: int)
+getMajor(): String
+setMajor(value: String)
+getReservationList(courtId: int): ArrayList<Duration>
Trainer
-id: int
-trainerType: String
+getId(): Int
+setId(value: int)
+getTrainerType(): String
+setTrainerType(value: String)
«interface»
EventDao
+getEventList(): String
+addEvent(name: Event): ErrorMessage
+deleteEvent(eventId: Int): ErrorMessage
+updateEvent(event: Event): ErrorMessage
+addUser(userId: User, eventId: int): ErrorMessage
+fetchRegisteredUser(eventId: int): ArrayList<String>
+secretEvent(eventId: Int): String
EventDaoImpl
-databaseType: String
+getDatabaseType(): String
+setDatabaseType(name: String): void
«interface»
CourtReservationDao
+reserveCourt(user: User, courtId: int, duration: Duration(): ErrorMessage
+deleteReservation(userId: int, reservationId: int): ErrorMessage
+fetchUserList(courtId: Int): ArrayList<User>
+getReservationDetails(courtId: int, duration: Duration(): Array..st<Duration>
CourtReservationDaoImpl
+databaseType: String
+getDatabaseType(): String
+setDatabaseType(value: String)
«interface»
AppointmentDao
+setAppointment(trainerId: int, userId: Int, duration: Duration(): ErrorMessage
+getAppointment(appointmentId: Int): Appointment
+deleteAppointment(appointmentId: int): ErrorMessage
AppointmentDaoImpl
+databaseType: String
+getDatabaseType(): String
+setDatabaseType(value: String)
Duration
-startTime: DateTime
-endTime: DateTime
-startDate: DateTime
-endDate: DateTime
+getStartTime(): DateTime
+setStartTime(value: DateTime)
+getEndTime(): DateTime
+setEndTime(value: DateTime)
+getStartDate(): DateTime
+setStartDate(value: DateTime)
+getEndDate(): DateTime
+setEndDate(value: DateTime)
ErrorMessage
+success: boolean
+getErrors(): String