

CSC343 :Systems Analysis & Design

Project final report.

Salym

(submitted date: 11\25\2024)

Group ()

Name	Contribution
Reem Alsuhaime (Leader)	REQ: 3 , 4 , 5 , 11 , 12 , 19 , 21. US Casual Description: 5-8 US diagram & traceability matrix & UC-4 description SSD , SD UC-4 Class diagram & State diagram booking
Tansnim Mohammed Kamal	REQ: 1 , 2 , 13 , 14 , 22 , 23 . US Casual Description: 9-12 US diagram & traceability matrix & UC-1 & UC-5 descriptions SD UC-5 Class diagram & State diagram Advertisement
Lama Almoghamis	REQ: 6 , 7 , 8 , 15 , 17 . US diagram & UC-7 description SD UC-1 Method description & Architectural diagram
Manar baamir	REQ: 9 , 10 , 16, 18 , 20. Stakeholders & Actors and goals US Casual Description: 1-4 US diagram & SD UC-7 Method description & Architectural diagram

Table of content:

1.	Customer Statement of Requirements (CSR).....	3
1.1.	Problem Statement	3
2.	System Requirements	4
2.1.	Enumerated Functional Requirements:	4
2.2.	Enumerated Nonfunctional Requirements	5
2.3.	On-Screen Appearance Requirements	5
3.	Functional Requirements Specification.....	7
3.1.	Stakeholders:	7
3.2.	Actors and Goals:	8
3.3.	Use Cases	9
I.	Casual Description.....	9
II.	Use Case Diagram :	10
III.	Fully-Dressed Description:.....	10
.4	Interaction Diagrams	16
4.1.	System Sequence Diagrams	16
4.2.	Sequence Diagrams	17
5.	System Architecture and System Design.....	19
5.1	System structural diagram	19
5.2	Architectural diagram	25
I.	Identifying Subsystems.....	25
II.	Suitable System Organization:	26
III.	Identifying Subsystems for Modular Decomposition	26
5.3	System behavioral diagram	27
6.	References :	27

1. Customer Statement of Requirements (CSR)

1.1. Problem Statement

As a student at King Saud University, I find myself struggling to make the most out of the fantastic sports facilities available on campus. The university's Sports Club is well-equipped with modern amenities for various activities such as tennis, basketball, and more. However, despite these excellent offerings, the club remains underutilized by students, particularly those who are not enrolled in sports-related programs like the College of Sports.

For someone like me, who has a full academic schedule outside of sports, one of the biggest hurdles is scheduling. It's difficult to determine when the sports facilities are available for free practice, as the timetable isn't readily accessible or easy to navigate. The existing system often prioritizes students involved in official sports classes, leaving the rest of us unsure about when we can use the facilities.

In addition to this, another significant problem is team formation. I love playing team sports, but often find it hard to connect with other students who share the same passion and have a similar availability. Forming a complete team for sports like basketball or soccer can be a challenge, and this lack of organization discourages spontaneous participation.

As a beginner in some sports, I also face the issue of lack of guidance. There are no easy ways to connect with experienced athletes or trainers who can guide me through the basics. I would love to participate more actively if I had access to mentors who could help me develop my skills.

I believe a dedicated application could significantly improve the situation by making the Sports Club more accessible and engaging for students like me. This platform could provide an organized schedule for all the sports facilities, allowing students to easily view available time slots for practice sessions. This would help us plan our visits without conflicting with official sports classes.

Furthermore, having virtual communities on the platform where students can connect based on their shared sports interests would be immensely helpful. This would allow us to form teams, organize matches, and even set training schedules together. It would create a stronger sense of community among students who are passionate about the same sports but may not have met otherwise.

For beginners like me, having an option where experienced students can volunteer as trainers would encourage us to participate more. The platform could allow skilled athletes to offer mentorship to students wanting to learn, fostering an inclusive and supportive sports environment.

In short, with better scheduling, team formation, beginner support, the Sports Club can truly become a hub for all students to engage in physical activities and promote a healthier, more connected campus life

2. System Requirements

(Note: Priority out of 5 points, where 5 is the highest priority and 1 is the lowest priority.)

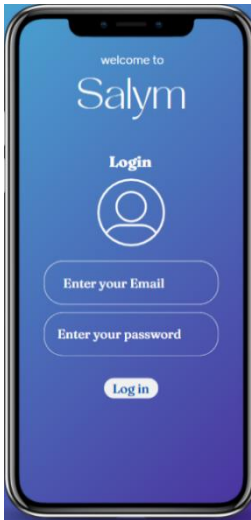
2.1. Enumerated Functional Requirements:




ID	Priority	Requirement
REQ-1	5	The system shall allow users to log in using their university email and the same password used in university System (so it has same rules as university edugate password).
REQ-2	4	The system shall automatically be able to upload the times available for free practice for each facility from edugate.
REQ-3	5	The system should allow the user to view and book facilities at available times.
REQ-4	2	The administrators shall be able to add sessions to the uploaded Schedule to the system through a form-based interface (only in Exceptional cases).
REQ-5	2	The administrators shall be able to delete sessions to the uploaded Schedule to the system through a form-based interface (only in Exceptional cases).
REQ-6	5	The system shall provide virtual communities as a chat page for each sport for all students. (Note: chat only with text)
REQ-7	3	The student should be able to create a training advertisement in the advertisements (page or section) of the system, the advertisement contains (Title, description, date, time, trainer name and experience as text only).
REQ-8	3	A student should be able to view (other students training advertisements, availability, and bookings).
REQ-9	1	system should have a notification system to alert students about new training sessions. By a push notification service to alert users about new training sessions as text, the notification will be sent once a week every Saturday (text e.g. Checkout a new training session of this week!)
REQ-10	2	Students should receive reminders for scheduled practices and events to ensure participation. by push notification service for scheduled practices and events 1 day and 1 hour before it starts, (text e.g. 1 hour left before “event name” starts!)


2.2. Enumerated Nonfunctional Requirements

REQ-11	5	system shall support iOS and Android mobile platforms.
REQ-12	4	the user interface shall be friendly to use.
REQ-13	4	System should be available on all school days (5 days a week) which represents (99.6% available) and have downtime (not more than 2 hours per month.) to ensure users can access it when needed.
REQ-14	3	The system maintenance periods should be only on holidays (weekends and official holidays).
REQ-15	4	The application should be as lightweight as possible, reducing storage consumption. // therefor the size must not be more than 25MB. And it should clean up old data exceeding the retention period (one year) to ensure efficiency.
REQ-16	2	system should have user support services to address user queries and issues. By either FAQ or user support services to address user queries and issues by sending the problem to the support email written on the home page and supporter will respond in their work hours.
REQ-17	1	The system should be able to integrate with existing university systems and databases for seamless data exchange and compatibility with other university tools and resources. by integrating with the university's edugate to enable seamless authentication using the same username and password as the university system. Additionally, student schedules should be automatically synced from the edugate to the sports club application to ensure accurate and up-to-date information, with the option for manual adjustments if necessary.
REQ-18	3	system should be able to handle a growing user base and increasing data volume by be able to handle 5000 users at the same time.

2.3. On-Screen Appearance Requirements

REQ-19	5	<p>GUI must have a landing page (register and log in).</p> 
--------	---	------------------------------------------------------------------------------------------------------------------------------------------------

REQ-20	5	<p>GUI must have a page of the user info, and all services provided by the application.</p>  <p>The first screen shows a user profile with a 'user name' label and a circular profile icon. The second screen shows a list of six options labeled 'OPTION 1' through 'OPTION 6'.</p>
REQ-21	5	<p>GUI must have a page of the facilities and their schedule.</p>  <p>The first screen shows a 'schedules' page with six buttons labeled 'facility 1' through 'facility 6'. The second screen shows a 'facility 1 schedule' page with a grid displaying times from 8:00 to 2:00 for days SUN, MON, TUE, WED, and THU.</p>
REQ-22	5	<p>GUI must have a page of sports' virtual communities.</p>  <p>The screen shows a 'virtual communities' page with three buttons labeled 'sport 1', 'sport 2', and 'sport 3'. Below the buttons is a 'sport 2 chat' section with a text input field and a send button.</p>

REQ-23	3	<p>GUI must have a page for creating or seeing advertisements.</p> 
--------	---	------------------------------------------------------------------------------------------------------------------------------------------------------

3. Functional Requirements Specification

3.1. Stakeholders:

Students:

The primary beneficiaries of the application, particularly those who are not majoring in sports. Our goal is to enhance their engagement in sporting activities .

Administrator (Sports Club Staff):

To manage facility schedules, add or delete sessions in exceptional cases, and oversee facility usage.

Sports Club Management:

Responsible for overseeing and developing the club's facilities. They will benefit from the data provided by the application regarding facility usage and student needs.

Volunteers:

Student volunteers who can offer support and coaching to beginner students. The application will facilitate communication and organization of training sessions.

University Administration:

Interested in increasing physical activity levels among students and promoting overall health, as well as utilizing student data for research related to health.

3.2. Actors and Goals:

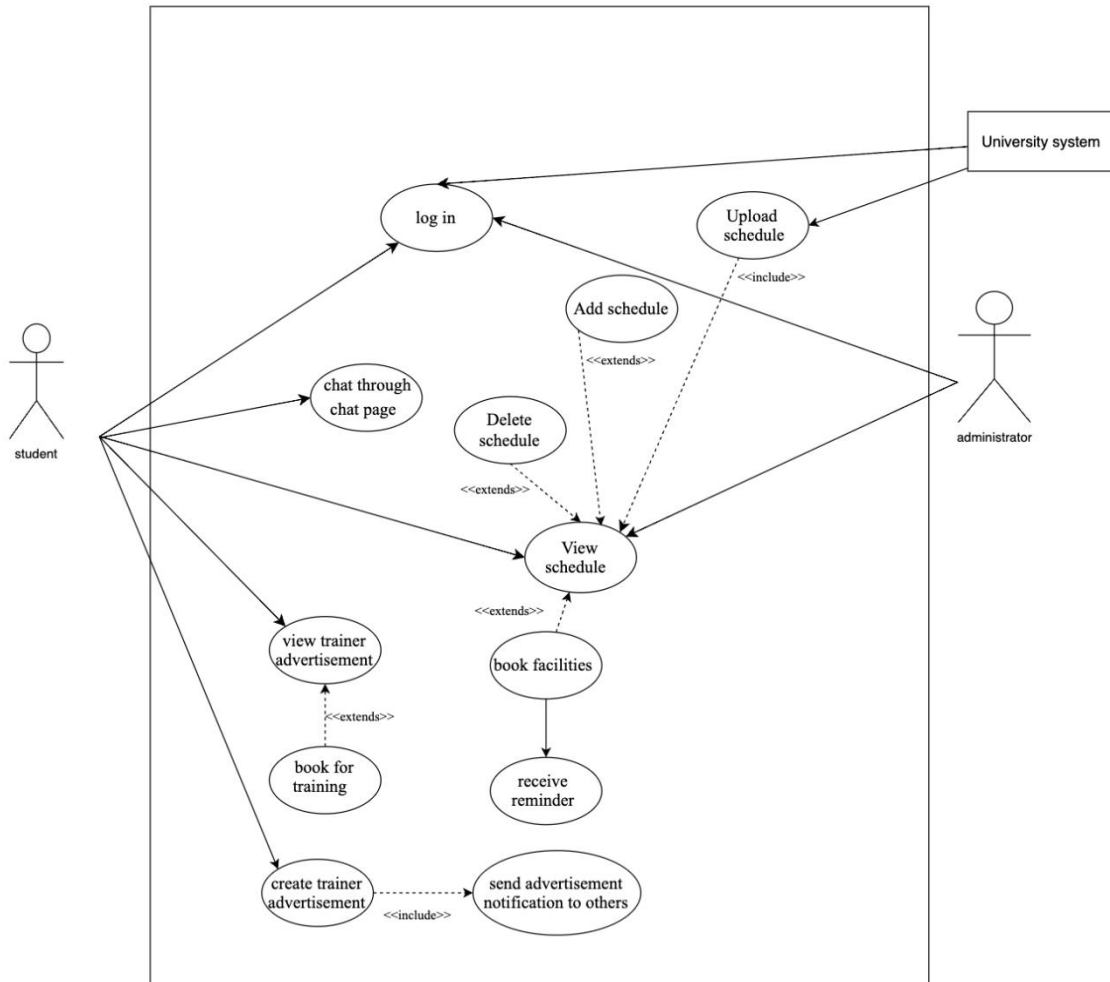
Primary/ Secondary Actors	Actor	Goals
Primary Actors	Student	<ul style="list-style-type: none"> • View and book sports facilities at available times. • Chat on chat pages for sports to connect with peers. • Form teams for team sports and organize matches. • Access training opportunities by connecting with volunteer trainers. • Receive reminders for booked sessions to ensure participation. • can volunteer and Advertise training sessions and availability to students. to help them develop new skills in various sports.
	Administrator	<ul style="list-style-type: none"> • Make exceptional changes to schedules by adding or deleting sessions as needed
Secondary Actors	University Systems	<ul style="list-style-type: none"> • Provide seamless integration with the Edugate system for authentication and data synchronization. • Offer a user-friendly and lightweight application accessible on iOS and Android. • Ensure high availability, minimal downtime, and efficient performance under peak load. • Support push notifications for updates and reminders.
	University Staff	<ul style="list-style-type: none"> • Increase student engagement in extracurricular activities. • Promote healthier lifestyles and physical activity among students. • Utilize data from the system for health research and service improvement.

3.3. Use Cases

I. Casual Description

Use Case	Action	Description
UC-1	Log in	A student can log in to the system using their university email and the same password used in the university's Edugate system, following the same password rules.
UC-2	Delete sessions	An administrator can delete sessions from the uploaded schedule using a form-based interface, but only in exceptional cases
UC-3	Add sessions	An administrator can add sessions to the uploaded schedule using a form-based interface, but only in exceptional cases
UC-4	View Schedule	A student can view available times for facilities and book them through the system.
UC-5	Chat through chat page	A student can chat and send text messages to other students through virtual groups dedicated to each sport.
UC-6	View trainer advertisement	A student can view sports training advertisements from different students through the advertisements page.
UC-7	Book facilities	student can view the schedules for each sports facility, including available time and reserved periods. and book a suitable available time through the system.
UC-8	Receive reminder	The student will receive a notification from the application if they book a free time slot, a reminders sent one day and one hour before the scheduled time.
UC-9	Book for training	The user can book one of the training advertisements displayed by the system, which allows him to benefit from the service provided by the trainers.
UC-10	Upload Schedule	The system uses the university system to automatically download the practice schedule that shows the free practice times so that the user can benefit from it to know the available times.
UC-11	Create a training advertisement	The trainer can create advertisements for the training courses that he offers through the system, and through them he can clarify the necessary information for the rest of the users, such as time, type of training, experiences, etc.
UC-12	Send notification system	After the trainee creates the ad, a notification will be sent to the rest of the users, but not immediately, but on Saturday to avoid inconvenience

II. Use Case Diagram :



III. Fully-Dressed Description:

traceability matrix:

	Priority Weight	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12
REQ-1	5	X											
REQ-2	5										X		
REQ-3	5				X			X					
REQ-4	2			X	X								
REQ-5	2		X		X								
REQ-6	5					X							
REQ-7	3											X	
REQ-8	3						X			X			
REQ-9	1								X				
REQ-10	2												X
Total Weight		5	2	2	9	5	3	5	1	3	4	3	2

Use Case ID:	UC-1
Use Case Name:	Log in
Actors:	Users (Student, Administrator) and university system
Description:	<p>User-Goal: authenticate users (students or administrators) and provide them access to the system.</p> <p>Sub-Functions:</p> <ul style="list-style-type: none"> • Displaying a secure login form for credential input. • Sending authentication requests to the university system (Edugate). • Handling and responding to authentication results (success or failure).
Stakeholders and Interests:	<p>Students: Want to access the system securely using their university credentials to book facilities, participate in chats, and view advertisements.</p> <p>Administrators: Need secure access to manage schedules and handle exceptional cases.</p> <p>University: Ensures data security and integrates with existing systems like Edugate for a seamless experience.</p>
Trigger:	<ul style="list-style-type: none"> • The user trigger: user opens the sports club mobile application on their device, then the application prompts the user to log in if no active session exists. • The administrator accesses the sports club system to manage schedules, the login screen is presented as the default entry point, requiring authentication.
Pre-conditions:	<ul style="list-style-type: none"> • The user must have a valid university email and password (managed via the Edugate system). • The system must be connected to the university's authentication system for seamless login.
Post-conditions:	<ul style="list-style-type: none"> • The user is logged into the system and can access their respective functionalities based on their role. • The system displays an appropriate error message, such as "Invalid username or password."
Normal Flow:	<ol style="list-style-type: none"> 1. The user opens the sports club application on their iOS or Android device. 2. The system displays the login page, prompting the user to enter their university email and password. 3. The user enters their credentials. 4. The system sends the credentials to the university's Edugate system for authentication. 5. The Edugate system verifies the credentials and returns a success response. 6. The system grants the user access to the application and redirects them to the homepage/dashboard based on their role (Student or Administrator).
Alternative Flows (Extensions):	<p>3a. Invalid Credentials:</p> <ol style="list-style-type: none"> 3a.1. The user enters an incorrect email or password. 3a.2. The Edugate system rejects the login request. 3a.3. The system displays an error message: "Invalid username or password. Please try again." <p>4a. Network/Authentication Failure:</p> <ol style="list-style-type: none"> 4a.1. The system cannot connect to the Edugate system. 4a.2. The system displays an error message: "Unable to connect to the authentication server. Please try again later."

Priority:	5
Technology and Data Variations List	<p>Input Methods: Touchscreen, single sign-on (SSO).</p> <p>Output Methods: Success/failure messages, redirect to dashboard, UI adapted for mobile or web platforms.</p> <p>Data Formats: JSON/XML authentication requests and responses. Email/password inputs follow strict formatting.</p> <p>Connectivity: Requires real-time online connectivity to Educate;</p>
Special Requirements:	<ul style="list-style-type: none"> • The system must enforce the same password rules as the university's Edugate system. • The login feature must comply with university security protocols, including secure transmission (e.g., HTTPS) of login credentials. • The system must support iOS and Android platforms for seamless user access.

Use Case ID:	UC-4
Use Case Name:	View schedule
Actors:	Users: Student, Administrator.
Description:	<p>User Goal: To enable users (students, trainers, or administrators) to view up-to-date schedules for facilities, sessions, and events, ensuring they can plan and participate in activities effectively.</p> <p>Sub-Functions:</p> <ul style="list-style-type: none"> • Fetch the schedule from the university's Edugate system or the local database. • Present schedules in a clear and intuitive format, such as a calendar, list, or table. • Handle scenarios where data retrieval fails, providing users with an appropriate error message (e.g., <i>"Unable to load schedules. Please try again later."</i>).
Stakeholders and Interests:	<p>Students: Want to view free practice times or booked slots for specific facilities.</p> <p>Administrators: Need visibility into facility schedules to manage exceptions.</p>
Trigger:	The user (student, trainer, or administrator) selects the "View Schedule" option in the application interface and selecting the facility required (either by clicking a button, selecting a menu option,).
Pre-conditions:	<p>The user must be logged into the system.</p> <p>Facility schedules must already be uploaded or synced from Edugate.</p>
Post-conditions:	<p>Success: The user views the updated facility schedule.</p> <p>Failure: The system displays an error (e.g., "Unable to fetch schedules").</p>
Normal Flow:	<ol style="list-style-type: none"> 1. The user navigates to the "View Schedule" section. 2. The system fetches the schedule from the database or Edugate. 3. The user selects a facility to view the schedule. 4. The system displays the schedule, highlighting available and booked slots.
Alternative Flows (Extensions):	<p>3a. No Schedules Available: The system shows a message: "No available schedules for this facility."</p> <p>4a. System Error: If fetching data fails, the system displays: "Unable to load schedules. Please try again later."</p>

Priority:	9
Technology and Data Variations List	Input Methods: Mobile (touch) Output Methods: Mobile (calendar, list), Web (interactive table, calendar). Data Formats: JSON/XML for schedule retrieval, date filters, and error handling. Display formats vary by platform. Connectivity: Online mode with Edugate or internal database, Offline mode with cached data.
Special Requirements:	<ul style="list-style-type: none"> • Integration with Edugate for automatic synchronization. • Real-time updates for schedule changes made by administrators.

Use Case ID:	UC-5
Use Case Name:	Chat through chat page
Actors:	User (Student)
Description:	<p>User Goal: Participate in real-time text conversations within virtual communities (chat rooms) dedicated to specific sports. Chat allows users to discuss schedules, events, or general topics related to sports in a collaborative and friendly environment, allowing users with common interests to organize teams and play together.</p> <p>Sub-Functions:</p> <ul style="list-style-type: none"> • Sending Messages: Users can type and send text-based messages to others in the same chat room. • Receiving Messages: Users receive, and view messages sent by others in real time, ensuring an active conversation flow. • Joining Specific Chat Rooms: Users can access chat rooms for specific sports, connecting with peers sharing similar interests. • Message History Access: Users can view the chat history to catch up on previous conversations within the retention period (up to 1 year).
Stakeholders and Interests:	<p>Students: Want to interact with peers in virtual communities (chat page) for specific sports.</p> <p>University/Sports Club: Facilitates communication among students</p>
Trigger:	The user selects a specific sport's chat page from the application interface.
Pre-conditions:	<ul style="list-style-type: none"> • The user must be logged into the system. • A chat room must exist for the selected sport.
Post-conditions:	<ul style="list-style-type: none"> • The user successfully sends and receives messages in the chat. • The system displays an error (e.g., <i>"Message failed to send."</i>).
Normal Flow:	<ol style="list-style-type: none"> 1. The user navigates to the chat section and selects a sport-specific chat room. The system displays the chat interface with previous messages. 2. The user types a message and sends it. 3. The message is displayed in the chat room for all participants to see. 4. The user receives messages from other participants in real time.
Alternative Flow (Extensions):	<p>3a.1. Message Too Long: the system displays: "Message exceeds the character limit."</p> <p>3a.2. Message Failed to Send: the system displays: "Unable to send message. Please try again."</p>

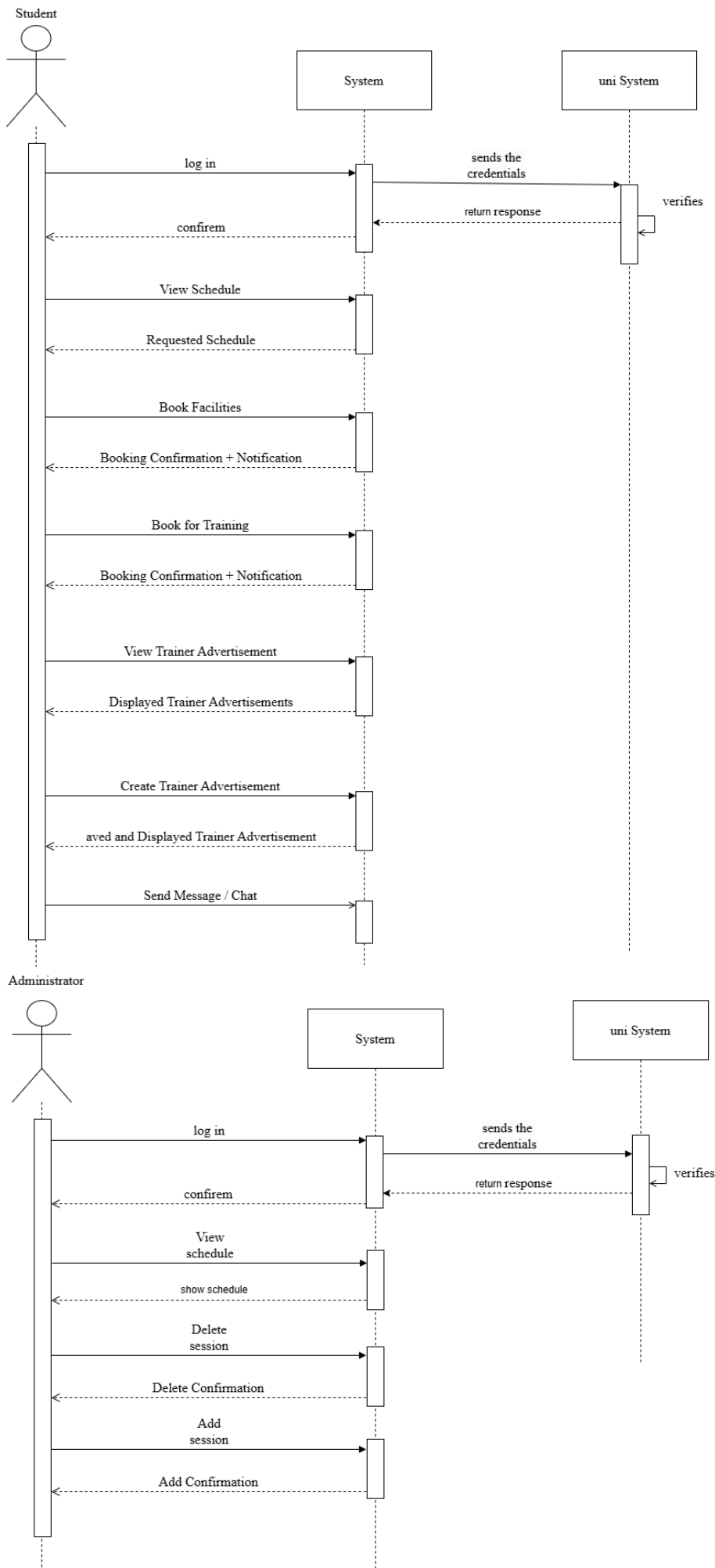
Priority:	5
Technology and Data Variations List	Input Methods: Mobile (touch). Output Methods: Mobile chat bubbles. Data Formats: JSON or relational data for message storage/transmission; Unicode for emojis. Connectivity: Online via WebSocket or HTTP(S); Offline with local caching for unsent messages.
Special Requirements:	<ul style="list-style-type: none"> • Text-only communication to reduce resource usage. • Real-time messaging support.

Use Case ID:	UC-7
Use Case Name:	Book facilities
Actors:	User (Student)
Description:	<p>User Goal: to allow students to reserve sports facilities for practice or events at available times. This ensures fair and organized access to facilities while enabling students to plan their activities conveniently.</p> <p>Sub-Functions:</p> <ul style="list-style-type: none"> • View Available Facilities and Times: users can browse a list of facilities (e.g., gym, swimming pool, courts) with their availability based on the schedule synced by Edugate. • Select Facility and Time Slot: users choose a specific facility and an available time slot that suits their schedule. • Receive Confirmation: the system sends a booking confirmation with the details (facility, date, time) for reference. • Modify or Cancel Booking (Optional): users may update or cancel their booking before the reserved time slot, depending on the system's cancellation policy. • Notifications: Users receive reminders for their booked facilities, such as 1 day and 1 hour before the reservation.
Stakeholders and Interests:	<p>Students: Want to secure facility slots for practice at their preferred times.</p> <p>University/Sports Club: Ensures efficient use of facilities and prevents overbooking.</p>
Trigger:	The user selects the 'Book Facilities' option from the application interface
Pre-conditions:	<ul style="list-style-type: none"> • The user must be logged in. • The schedule for the facility must already exist and show available slots.
Post-conditions:	<ul style="list-style-type: none"> • Success: The facility slot is booked, and the user receives confirmation. • Failure: The system displays an error (e.g., "Slot unavailable").
Normal Flow:	<ol style="list-style-type: none"> 1.The user selects a facility and views the schedule. 2.The user chooses an available time slot. 3.The system validates the booking (e.g., checks availability and user eligibility). 4.The system confirms the booking and updates the schedule. 5.The user receives a confirmation notification.
Alternative Flows (Extensions):	<p>3a. Slot Already Booked:</p> <p>The system displays: "Selected slot is no longer available. Please choose another slot."</p>

	<p>4a. System Error:</p> <p>If booking fails, the system displays: "Unable to process your booking. Please try again later."</p>
Priority:	5
Technology and Data Variations List	<p>Input Methods: Mobile (touch).</p> <p>Output Methods: Push notifications (mobile).</p> <p>Data Formats:</p> <p>Time slots: HH:MM (24-hour).</p> <p>Dates: YYYY-MM-DD.</p> <p>Notifications: Plain text.</p> <p>Connectivity:</p> <p>Integration with Edugate system for availability.</p> <p>Integration with university database for validation and booking storage.</p>
Special Requirements:	<p>Notifications for confirmed bookings.</p> <p>Concurrent booking prevention to avoid double-booking of slots.</p>

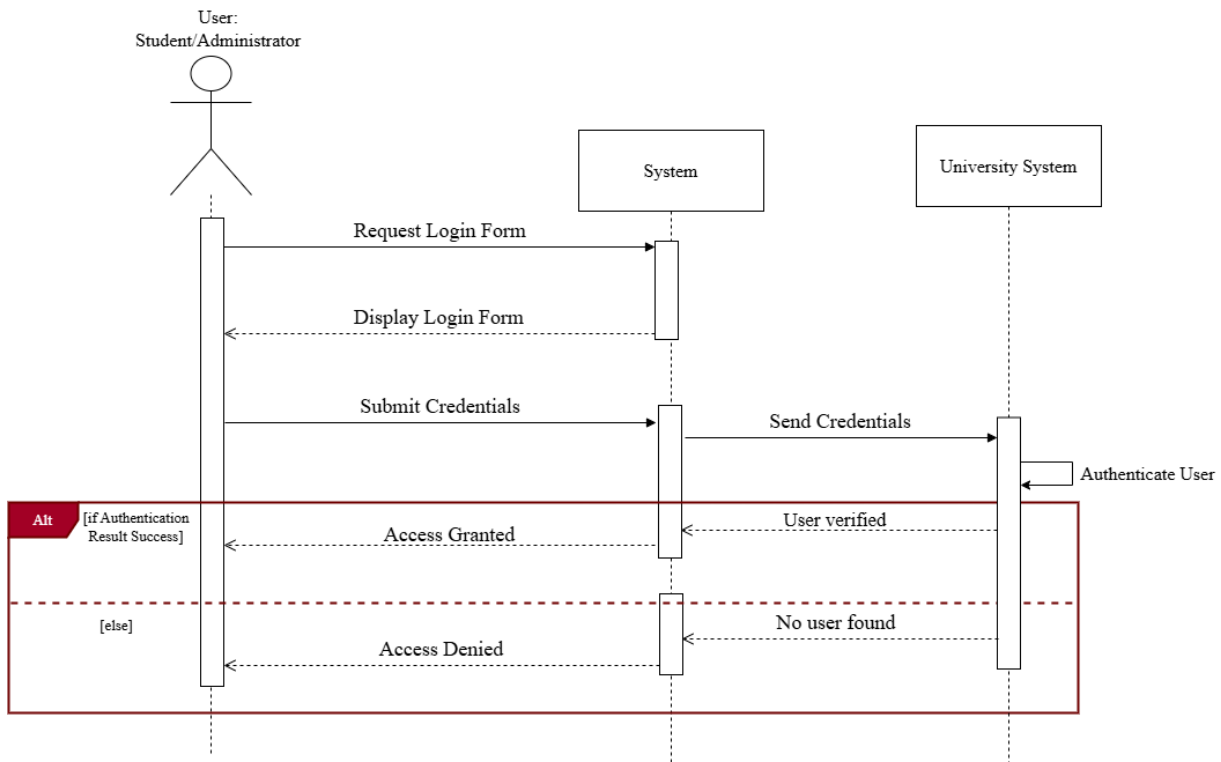
4. Interaction Diagrams

4.1. System Sequence Diagrams

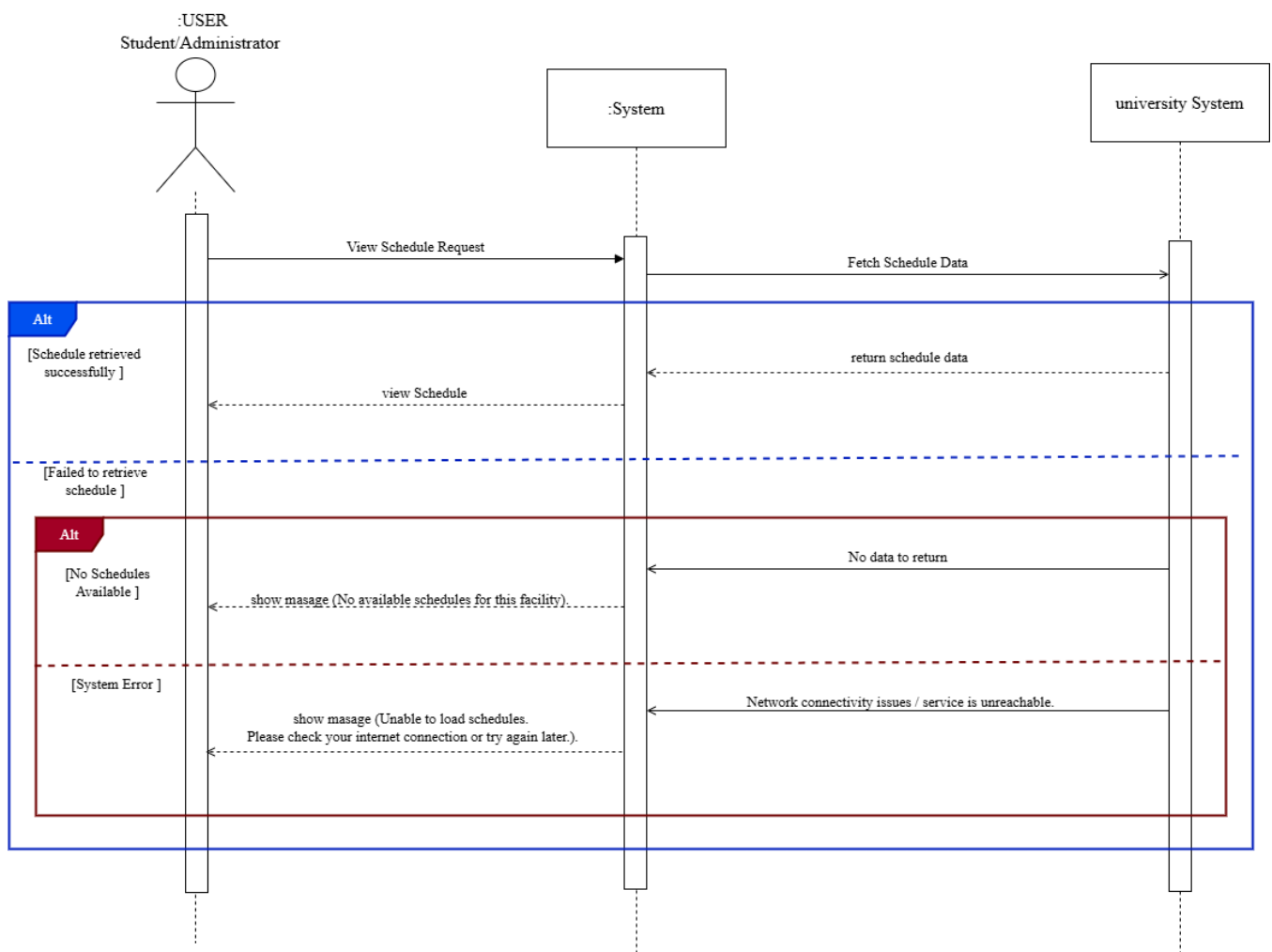


4.2. Sequence Diagrams

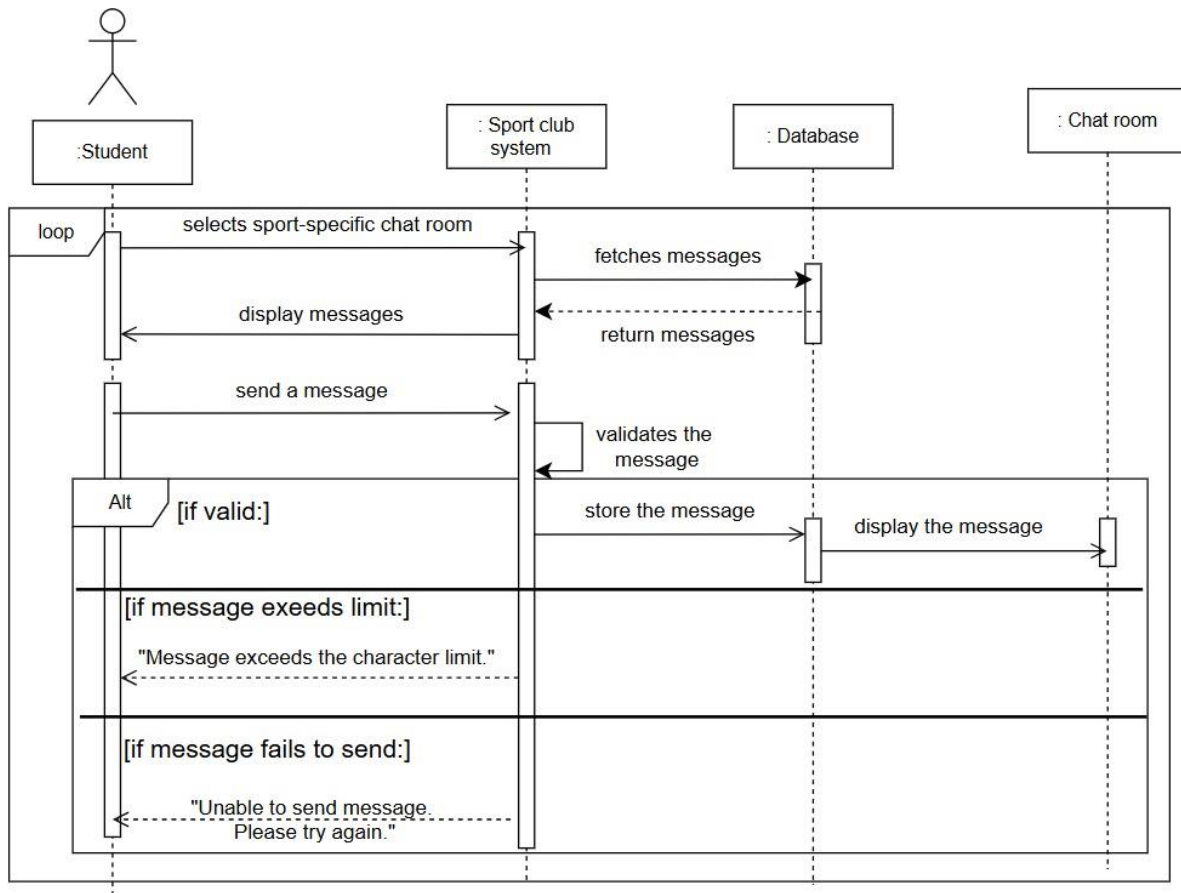
UC-1 <Log-in>:



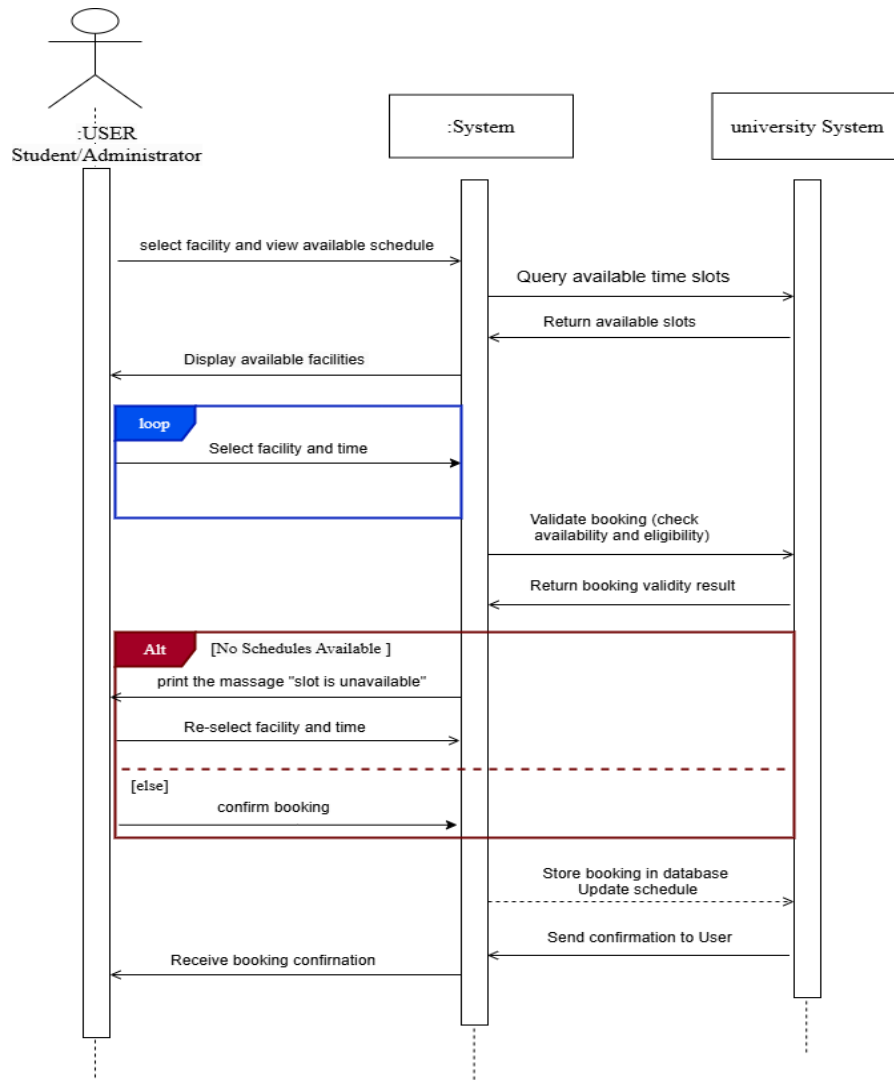
UC-4 <View schedule >:



UC-5 <chat through chat page>:

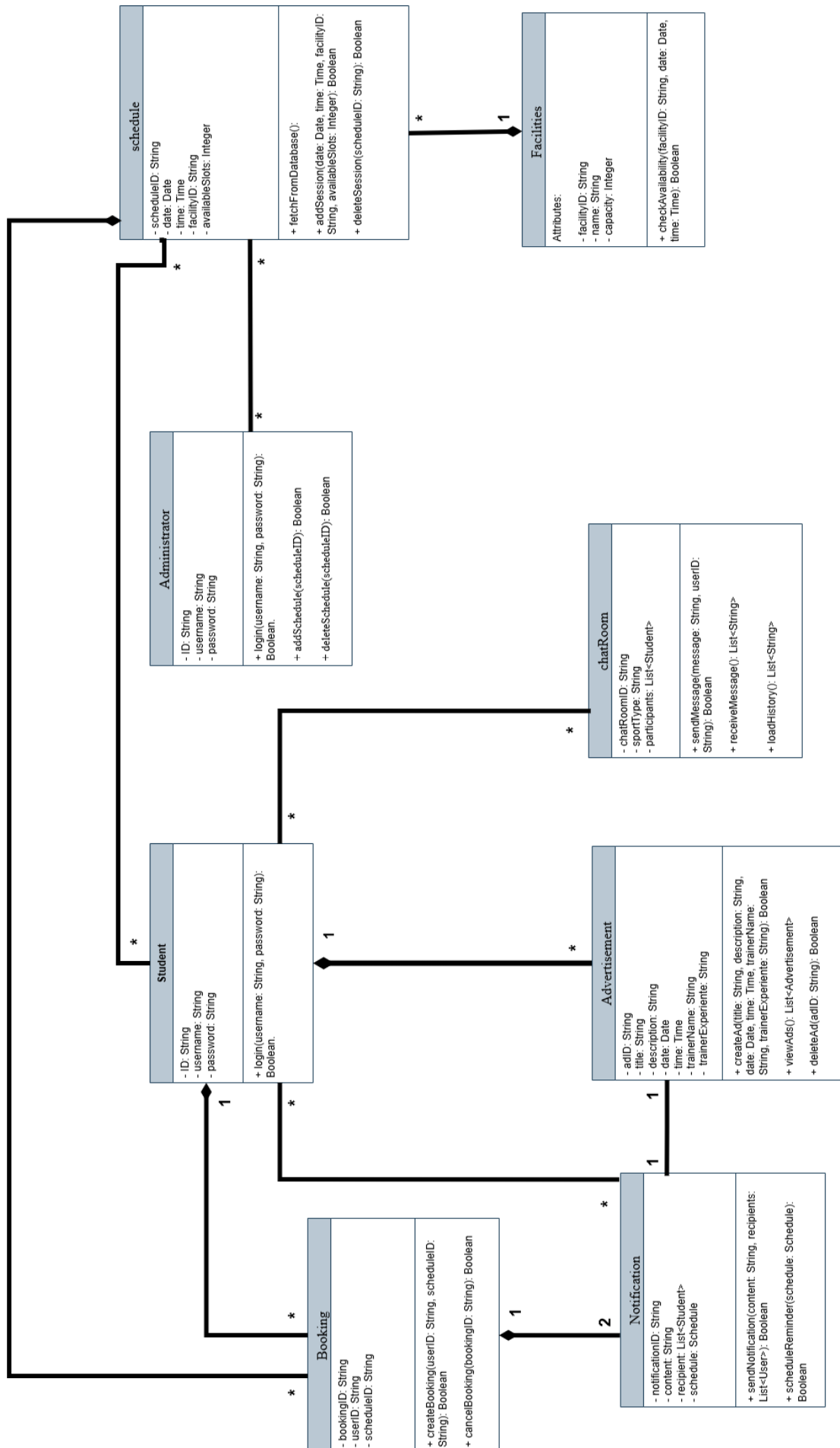


UC-7 < Book facilities >:



5. System Architecture and System Design

5.1 System structural diagram



Student:

Attribute

- *ID: String*

(A unique identifier assigned to each student. This ID is used to differentiate between users in the system.)

- *username: String*

(The university email of the student, used for logging into the system.)

- *password: String*

(The password associated with the university account, adhering to the same security rules as the university's edugate system.)

Methods

+ *login(username: String, password: String): Boolean.*

Description: This method allows a student to log into the system by validating their username and password against the credentials stored in the university's Edugate system.

Parameters:

- username: The student's university email address.

- password: The student's password.

Returns: A Boolean value indicating whether the login was successful (true) or failed (false).

Booking:

Attribute

- *bookingID: String*

(A unique identifier for each booking made within the system.)

- *userID: String*

(The ID of the student who made the booking, linking the booking to a specific user.)

- *scheduleID: String*

(The ID of the schedule entry for the facility being booked.)

Methods

+ *createBooking(userID: String, scheduleID: String): Boolean*

Description: This method allows a student to create a booking for a specific facility at designated time.

Parameters:

- userID: The ID of the student wishing to make the booking.

- scheduleID: The ID of the schedule entry corresponding to the facility and time.

Returns: A Boolean indicating whether the booking was successfully created (true) or not (false).

+ *cancelBooking(bookingID: String): Boolean*

Description: This method allows a student to cancel an existing booking using its unique booking ID.

Parameters:

- bookingID: The unique identifier of the booking to be canceled.

Returns: A Boolean indicating whether the cancellation was successful (true) or not (false).

Notification:

Attribute

- *notificationID*: *String*

(A unique identifier for each notification sent by the system.)

- *content*: *String*

(The text content of the notification, detailing the message to be conveyed to users.)

- *recipient*: *List<Student>*

(A list of students who are recipients of the notification, allowing for targeted messaging.)

- *schedule*: *Schedule*

(The schedule associated with the notification, which may contain information about events or practices.)

Methods

+ *sendNotification(content: String, recipients: List<User>): Boolean*

Description: This method sends a notification with specified content to a list of recipients.

Parameters:

- *content*: The message to be sent to the users.

- *recipients*: A list of Student objects representing the users who will receive the notification.

Returns: A Boolean indicating whether the notification was successfully sent (true) or not (false).

+ *scheduleReminder(schedule: Schedule): Boolean*

description: This method schedules a reminder notification for a specific practice or event based on the provided schedule.

Parameters:

- *schedule*: The schedule object that includes the details of the event for which reminders are to be sent.

Returns: A Boolean indicating whether the reminder was successfully scheduled (true) or not (false).

Advertisement:

Attribute

- *adID*: *String*

(A unique identifier for each training advertisement created within the system.)

- *title*: *String*

(The title of the training advertisement, summarizing the content.)

- *description*: *String*

(A detailed description of the training session being advertised.)

- *date*: *Date*

(The date of the training session.)

- *time*: *Time*

(The time of the training session.)

- *trainerName: String*

(The name of the trainer leading the session.)

- *trainerExperiente: String*

(A brief overview of the trainer's experience, detailing their qualifications or background.)

Methods:

+ *createAd(title: String, description: String, date: Date, time: Time, trainerName: String, trainerExperiente: String): Boolean*

description: This method allows an administrator or user to create a new training advertisement with relevant details.

Parameters:

- title: The title for the advertisement.
- description: A detailed description of the training session.
- date: The date when the training will occur.
- time: The time when the training will occur.
- trainerName: The name of the instructor.
- trainerExperience: The experience details of the trainer.

Returns: A Boolean indicating whether the advertisement was successfully created (true) or not (false)

+ *viewAds(): List<Advertisement>*

Description: This method retrieves a list of all currently available training advertisements in the system.

Returns: A list of Advertisement objects representing all active advertisements.

+ *deleteAd(adID: String): Boolean*

Description: This method allows the deletion of a specific advertisement using its unique ad ID.

Parameters:

- adID: The unique identifier of the advertisement to be deleted.

Returns: A Boolean indicating whether the advertisement was successfully deleted (true) or not (false).

Administrator

Attribute:

-*ID (String):*

A unique identifier assigned to each administrator. This ID is used to distinguish between different administrators in the system.

-*username (String):*

The university email or username of the administrator. This is used for login purposes and must be unique across the system.

-*password (String):*

The password associated with the administrator's account. It should follow security protocols to ensure the safety of the administrator's account.

Methods:

+ *login(username: String, password: String): Boolean*

Description: Authenticates the administrator by validating their username and password against stored credentials in the system.

Parameters:

username: The username (university email) of the administrator.

password: The password for the administrator's account.

Returns: A Boolean value (true if authentication is successful, false otherwise).

+ *addSchedule(scheduleID: String): Boolean*

Description: Adds a new schedule entry to the system using the provided schedule ID.

Parameters:

scheduleID: The unique identifier for the schedule being added.

Returns: A Boolean indicating whether the schedule was successfully added (true) or not (false).

+ *deleteSchedule(scheduleID: String): Boolean*

Description: Deletes an existing schedule entry from the system based on the provided schedule ID.

Parameters:

scheduleID: The unique identifier of the schedule to be deleted.

Returns: A Boolean indicating whether the schedule was successfully deleted (true) or not (false).

Facilities

Attribute:

-*facilityID (String):*

A unique identifier for each facility. This ID is used to manage and reference facilities in the system.

-*name (String):*

The name of the facility (e.g., "Main Gym", "Swimming Pool"). This is used for display purposes and user selection.

-*capacity (Integer):*

The maximum number of users that can use the facility at a given time. This helps in managing bookings and ensuring safety regulations.

Methods:

+ *checkAvailability(facilityID: String, date: Date, time: Time): Boolean*

Description: Checks if the specified facility is available for booking at the given date and time.

Parameters:

facilityID: The ID of the facility being checked.

date: The date for which availability is being checked.

time: The time for which availability is being checked.

Returns: A Boolean indicating whether the facility is available (true) or not (false).

Schedule

Attribute:

-scheduleID (String):

A unique identifier for each schedule entry. This ID is used to manage and reference schedules in the system.

-date (Date):

The date of the scheduled event or session. It helps in organizing and displaying schedules by day.

-time (Time):

The specific time of the scheduled event. This attribute is essential for managing when facilities are booked.

-facilityID (String):

The identifier for the facility associated with the schedule. This links the schedule to a specific facility available for booking.

-availableSlots (Integer):

The number of slots available for booking at the specified time and date. This helps in tracking how many reservations can still be made.

Methods:

+ fetchFromDatabase():

Description: Retrieves schedule data from the database, populating the class attributes with the relevant information.

Returns: Typically, does not return a value but updates the object's state with data fetched from the database.

+ addSession(date: Date, time: Time, facilityID: String, availableSlots: Integer): Boolean

Description: Adds a new session to the schedule for a specific date and time at a given facility.

Parameters:

date: The date for the new session.

time: The time for the new session.

facilityID: The ID of the facility being scheduled.

availableSlots: The number of slots available for booking.

Returns: A Boolean indicating whether the session was successfully added (true) or not (false).

+ deleteSession(scheduleID: String): Boolean

Description: Deletes a session from the schedule using its unique schedule ID.

Parameters:

scheduleID: The unique identifier of the session to be deleted.

Returns: A Boolean indicating whether the session was successfully deleted (true) or not (false).

ChatRoom

Attribute:

-chatRoomID (String):

A unique identifier for each chat room. This ID is used to manage and reference chat rooms within the system.

-sportType (String):

The type of sport associated with the chat room (e.g., "Football", "Basketball"). This helps categorize chat rooms by interest.

-participants (List<Student>):

A list of Student objects representing the users currently participating in the chat room. This allows for tracking who is involved in discussions.

Methods:

+ *sendMessage(message: String, userID: String): Boolean*

Description: Sends a message to the chat room from a specified user.

Parameters:

message: The content of the message being sent.

userID: The ID of the user sending the message.

Returns: A Boolean indicating whether the message was successfully sent (true) or not (false).

+ *receiveMessage(): List<String>*

Description: Retrieves a list of messages from the chat room for display to the participants.

Returns: A list of strings, each representing a message received in the chat room.

+ *loadHistory(): List<String>*

Description: Loads the chat history for the chat room, allowing users to view past messages.

Returns: A list of strings representing the historical messages exchanged in the chat room.

5.2 Architectural diagram

I. Identifying Subsystems

1. User Management Subsystem:

- Handles user registration, authentication, and profile management.
- Responsible for managing student and administrator accounts.

2. Booking Management Subsystem:

- Manages facility scheduling and bookings.
- Responsible for creating, viewing, and canceling bookings.

3. Notification Management Subsystem:

- Manages the notification system for reminders and alerts about training sessions.
- Handles scheduling and sending notifications to users.

4. Advertisement Management Subsystem:

- Manages the creation, viewing, and deletion of training advertisements.
- Responsible for maintaining the advertisement lifecycle.

5. Community Interaction Subsystem:

- Facilitates chat functionalities for students to interact based on sports.
- Supports text-based communication.

6. Integration Subsystem:

- Manages the integration with the university's Edugate system.
- Facilitates data synchronization, such as student schedules and authentication.

II. Suitable System Organization:

Client-Server Architecture is the most suitable system organization for this application.

Reasons:

- **Scalability:** The application needs to handle a growing user base (up to 5000 concurrent users). A client-server model can easily scale by adding more servers or resources.
- **Centralized Data Management:** The server can manage the shared data repository effectively, ensuring data consistency and integrity across the system (e.g., handling bookings and advertisements).
- **Separation of Concerns:** This architecture allows for a clear separation between the client (user interface) and server (business logic and data management), which simplifies development and maintenance.
- **Accessibility:** Users can access the application from various devices (iOS and Android), and the client-server model facilitates remote access to the centralized server.

III. Identifying Subsystems for Modular Decomposition

Subsystems Suitable for Modular Decomposition:

1. User Management Subsystem:

- a. **Modules:** Authentication Module, User Profile Module, Role Management Module.
- b. **Control Model:** Event-Driven Model, where user actions trigger events (e.g., login attempts) that invoke specific functionalities.

2. Booking Management Subsystem:

- a. **Modules:** Booking Creation Module, Booking Cancellation Module, Schedule Management Module.
- b. **Control Model:** Transactional Control Model, ensuring that bookings and cancellations are processed reliably and consistently.

3. Notification Management Subsystem:

- a. **Modules:** Notification Creation Module, Notification Scheduling Module, Notification Delivery Module.
- b. **Control Model:** Observer Model, where changes in the scheduling system trigger notifications to users.

4. Advertisement Management Subsystem:

- a. **Modules:** Advertisement Creation Module, Advertisement Listing Module, Advertisement Deletion Module.
- b. **Control Model:** Command Pattern, where commands for creating, viewing, or deleting advertisements are encapsulated and executed.

5. Community Interaction Subsystem:

- a. **Modules:** Chat Interface Module, Message Handling Module, User Presence Module.
- b. **Control Model:** Publish-Subscribe Model, where users can subscribe to chat rooms and receive messages published by others.

6. Integration Subsystem:

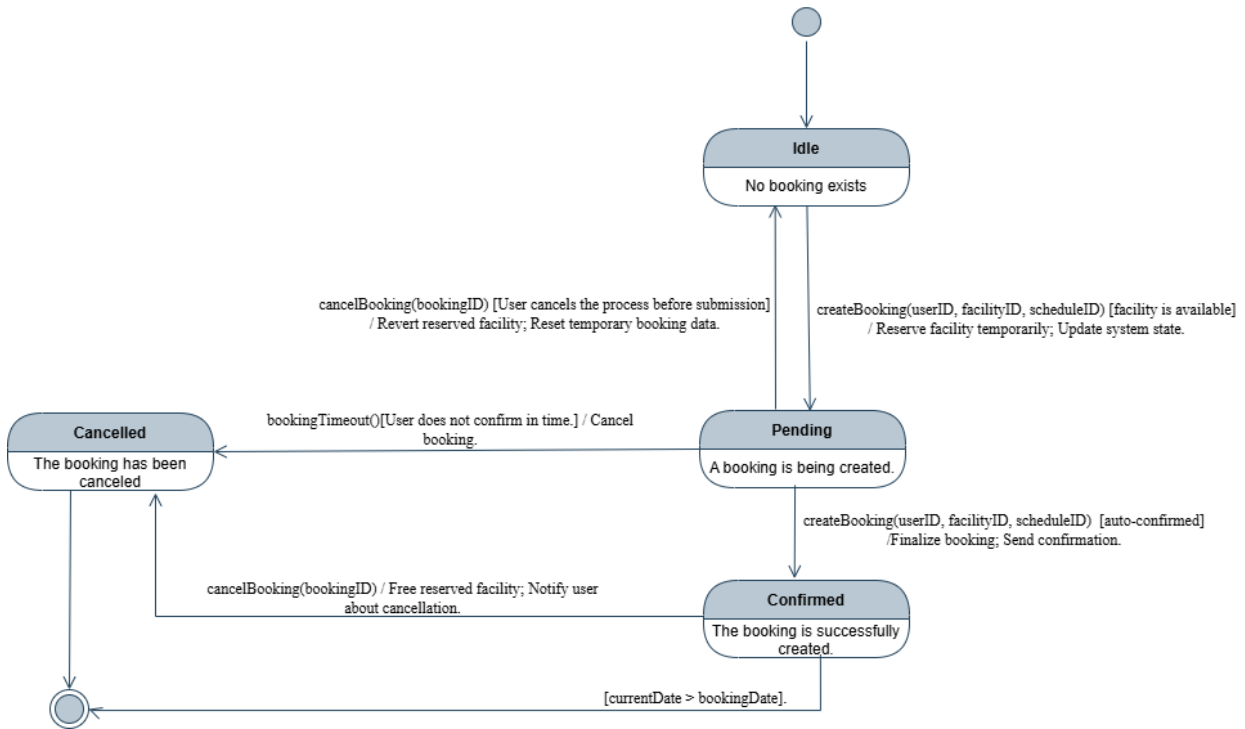
- a. **Modules:** Edugate Integration Module, Data Synchronization Module.

- b. **Control Model:** Mediator Model, coordinating interactions between the application and external systems (like Edugate).

By organizing the system into these subsystems and using a client-server architecture, the application can achieve modularity, flexibility, and scalability, making it easier to manage and evolve over time.

5.3 System behavioral diagram

Booking:



6. References :

- [1] Sommerville, I. (2010). Software Engineering (9th edition). [Online]. Available: <https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9thhttps://engineering.futureuniversity.com/BOOKS FOR IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdfEdition-by-Ian-Sommerville.pdf>