**COMSATS University \Islamabad,**

**Abbottabad Campus**

**SOFTWARE DESIGN DOCUMENT  
(SDD)**

**for**

**<University Attendance System>**  
Version 1.0

***By***

**Fawad Iqbal CIIT/FA21-BSE-012/ATD**

**Faizan CIIT/FA21-BSE-011/ATD**

***Supervisor*Prof. Mukhtiar Zamin**

***Bachelor of Science in Software Engineering (2021-2025)***

Table of Contents

[Chapter 1 Project Proposal 1](#_Toc140975690)

[1.1 Introduction 1](#_Toc140975691)

[1.2 Vision 1](#_Toc140975692)

[1.3 Business Case 2](#_Toc140975693)

[1.4 Functional Requirements 2](#_Toc140975694)

[1.4.1 Admin Requirements 2](#_Toc140975695)

[1.4.2 Teacher Requirements 3](#_Toc140975696)

[1.4.3 Student Requirements 3](#_Toc140975697)

[1.4.4 System Requirements 4](#_Toc140975698)

[1.5 Supplementary Specification 4](#_Toc140975699)

[1.5.1 User Interface: 4](#_Toc140975700)

[1.5.2 Performance: 5](#_Toc140975701)

[1.5.3 Security: 5](#_Toc140975702)

[1.5.4 Reports: 5](#_Toc140975703)

[1.5.5 Integration: 5](#_Toc140975704)

[1.5.6 Accessibility: 5](#_Toc140975705)

[1.5.7 Support and Maintenance: 6](#_Toc140975706)

[1.5.6 Performance Metrics: 6](#_Toc140975707)

[1.5.7 Conclusion: 6](#_Toc140975708)

[1.6 Glossary 6](#_Toc140975709)

[1. Attendance System: 6](#_Toc140975710)

[1.7 Risk List & Risk Management Plan 7](#_Toc140975711)

[1.7.1 Risk List 7](#_Toc140975712)

[1.7.2 Risk Management Plan 7](#_Toc140975713)

[Chapter 2 Use Cases 9](#_Toc140975714)

[2.1 Use Case Diagram 9](#_Toc140975715)

[2.2 Brief Level Use Case 9](#_Toc140975716)

[2.3 Fully Dressed Use Cases 9](#_Toc140975717)

[2.3.1 Login 9](#_Toc140975718)

[2.3.2 Add User 11](#_Toc140975719)

[2.3.3 Add Student 13](#_Toc140975720)

[2.3.4 Add Course 16](#_Toc140975721)

[2.3.5 Add Teacher 18](#_Toc140975722)

[2.3.6 Assign Course to Teacher 20](#_Toc140975723)

[Chapter 3 UI Prototypes 23](#_Toc140975724)

[3.1 Login UI 23](#_Toc140975725)

[3.2 Admin Dashboard 24](#_Toc140975726)

[3.3 Add User 25](#_Toc140975727)

[3.4 View Users 26](#_Toc140975728)

[3.5 Add Student 27](#_Toc140975729)

[3.6 View Students 28](#_Toc140975730)

[3.7 Add Course 29](#_Toc140975731)

[3.8 View Courses 30](#_Toc140975732)

[3.9 Add Teacher 31](#_Toc140975733)

[3.10 View Teachers 32](#_Toc140975734)

[3.11 Assign Course to Teacher 33](#_Toc140975735)

[Chapter 4 Domain Model 34](#_Toc140975736)

[4.1 Domain Model 34](#_Toc140975737)

[Chapter 5 System Sequence Diagram (SSD) 35](#_Toc140975738)

[5.1 SSD 35](#_Toc140975739)

[Chapter 6 Operation Contract (OC’s) 36](#_Toc140975740)

[6.1 Operation Contract 36](#_Toc140975741)

[6.1.1 Login 36](#_Toc140975742)

[6.1.2 Add User 36](#_Toc140975743)

[6.1.3 Add Student 37](#_Toc140975744)

[6.1.4 Add Course 37](#_Toc140975745)

[6.1.5 Add Teacher 38](#_Toc140975746)

[6.1.6 Assign Course to Teacher 38](#_Toc140975747)

[Chapter 7 Package Diagram 40](#_Toc140975748)

[7.1 Package Diagram 40](#_Toc140975749)

[Chapter 8 Class Diagram 41](#_Toc140975750)

[8.1 Class Diagram: 41](#_Toc140975751)

[Chapter 9 Interaction Diagram 42](#_Toc140975752)

[9.1 Login 42](#_Toc140975753)

[9.2 Sequence Diagram 43](#_Toc140975754)

[9.2.1 Login 43](#_Toc140975755)

[9.2.2 Add User 43](#_Toc140975756)

[9.2.3 Add Student 44](#_Toc140975757)

[9.2.4 Add Course 44](#_Toc140975758)

[9.2.5 Add Teacher 45](#_Toc140975759)

[9.2.6 Assign Course to Teacher 45](#_Toc140975760)

[9.2.7 View Users 46](#_Toc140975761)

[9.2.7 Delete User 46](#_Toc140975762)

[9.2.8 View Students 47](#_Toc140975763)

[9.2.9 View Courses 47](#_Toc140975764)

[9.2.10 View Teachers 48](#_Toc140975765)

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*Include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**<Prof. Mukhtiar Zamin>**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Chapter 1 Project Proposal

## 1.1 Introduction

Efficient management of attendance records is essential in the dynamic landscape of education. Traditional attendance tracking methods often involve laborious manual record-keeping and administrative efforts, leading to inefficiencies and potential errors. To address these challenges, this thesis presents a groundbreaking desktop application for a university attendance system. The application empowers teachers by providing them with a user-friendly interface to mark, modify, and delete attendance records, revolutionizing the way attendance is managed.

The core objective of this desktop application is to streamline the attendance management process within the university environment. By leveraging modern technology, the application simplifies and automates tasks that were once time-consuming and prone to mistakes. A key feature of the application is its fully functional login system, which ensures secure access, validates user credentials, and authenticates authorized users. This aspect guarantees the integrity and confidentiality of attendance data, providing peace of mind to both teachers and administrators.

With this application, teachers gain a powerful tool to mark attendance accurately and efficiently. They can easily modify attendance records to accommodate makeup classes, correct mistakes, or adapt to changing circumstances. The ability to delete attendance entries in certain situations adds flexibility and ensures accurate reporting. By digitizing the attendance tracking process, the application reduces the burden of manual paperwork and data entry, freeing up valuable time for teachers to focus on their core responsibilities of instruction and mentorship.

## 1.2 Vision

The vision for the desktop application for the university attendance system is to transform the way attendance management is conducted within educational institutions. By leveraging modern technology and intuitive design, the application aims to provide an efficient, accurate, and user-friendly solution for teachers and administrators.

The goal is to streamline attendance tracking processes, reduce administrative burdens, and enhance overall productivity within the academic environment.

## 1.3 Business Case

#### The development and implementation of the desktop application for the university attendance system presents several compelling business cases. These cases revolve around improving operational efficiency, enhancing data accuracy, and facilitating informed decision-making. The key business cases include:

#### Time and Cost Savings: The application reduces manual paperwork, streamlines attendance tracking, and automates administrative tasks. This leads to significant time and cost savings for educational institutions by minimizing the effort required for data entry, record-keeping, and report generation.

#### Improved Accuracy and Compliance: The application ensures accurate and reliable attendance data by eliminating the possibility of human errors inherent in manual processes. It also helps institutions meet regulatory requirements and maintain compliance by providing a systematic and auditable attendance management system.

#### Improved Student Experience: The application contributes to a positive student experience by ensuring accurate attendance records, minimizing disputes, and enabling prompt interventions when necessary. Students benefit from a fair and transparent attendance tracking system that supports their academic journey.

## 1.4 Functional Requirements

### 1.4.1 Admin Requirements

* **Course and Class Management:** The administrator can manage courses and classes within the system. This involves creating and updating course information, assigning teachers to specific courses or classes, and managing class schedules.
* **Attendance Reporting:** The administrator has access to comprehensive attendance reports for analysis and decision-making. They can generate attendance reports for individual students, classes, or courses over specific time periods. The reports may include details such as attendance percentages, late arrivals, and absences.
* **Data Analytics:** The administrator can leverage data analytics functionalities to gain insights into attendance trends and patterns. They can view graphical representations and statistical summaries of attendance data, enabling them to identify areas of concern, track improvement initiatives, and make data-driven decisions.

### 1.4.2 Teacher Requirements

* **Login and Authentication:** The teacher can securely log into the application using their credentials and undergo authentication to access the attendance system.
* **Course and Class Selection:** The teacher can select the relevant course and class for which they want to mark attendance. They can view their assigned courses, class schedules, and associated students.
* **Mark Attendance:** The teacher can mark attendance for students in their assigned classes. They can indicate whether students are present, absent, or late. The system should allow for batch marking of attendance if necessary.
* **Modify Attendance:** The teacher can modify attendance records if changes or corrections are required. They can update attendance statuses, adjust for makeup classes, or rectify any inaccuracies in the attendance data.
* **View Attendance Summary:** The teacher can view attendance summaries for individual students or entire classes. This includes the total number of classes attended, absences, late arrivals, and overall attendance percentages.

### 1.4.3 Student Requirements

* **Login and Authentication:** The student can securely log into the application using their credentials and undergo authentication to access the attendance system.
* **View Attendance Records:** The student can view their own attendance records for the courses they are enrolled in. They can see the number of classes attended, absences, and late arrivals, providing them with a clear overview of their attendance performance.
* **Receive Notifications:** The student can receive notifications or alerts related to attendance, such as reminders about upcoming classes, changes in schedules, or important attendance-related announcements.
* **Access Attendance Reports:** The student can access attendance reports to view their attendance history for specific courses or classes. These reports may include detailed information about individual attendance records and percentages.

### 1.4.4 System Requirements

* **User Authentication and Access Control:** 
  + The system should provide secure login functionality for admins, students, and teachers.
  + Only registered and authorized users should be able to access the attendance system.
  + User roles and permissions should be defined to control access to distinctive features and data within the system.
* **Attendance Marking and Modification:**
  + The system should allow teachers to mark attendance for students, specifying whether they are present, absent, or late.
  + Teachers should have the ability to modify attendance records when necessary, such as for makeup classes or corrections of errors.
* **Data Management and Storage:**
  + The system should securely store and manage attendance records for each student and class.
  + Attendance data should be associated with specific courses, classes, dates, and students.
  + The system should provide efficient methods for storing and retrieving attendance information.

## 1.5 Supplementary Specification

The Attendance System is an application designed to manage and track student attendance within a university environment. This Supplementary Specification outlines additional features and requirements for the system to enhance its functionality and usability.

### 1.5.1 User Interface:

The user interface should be intuitive and user-friendly, allowing teachers to easily mark attendance, modify records, and generate reports. The interface should provide clear instructions and guidance for efficient attendance management.

### 1.5.2 Performance:

The system should be capable of handling a large number of students, classes, and attendance records. It should be able to process attendance data quickly and generate reports in a timely manner. The system should maintain optimal performance even during peak usage periods.

### 1.5.3 Security:

The system should implement robust security measures to protect student attendance data. User authentication and authorization mechanisms should be in place to ensure that only authorized personnel can access and modify attendance records. All data transmissions should be encrypted to maintain data integrity and confidentiality.

### 1.5.4 Reports:

The system should generate comprehensive attendance reports for teachers, administrators, and students. The reports should include attendance percentages and absences for individual students, classes, and courses. The reports should be customizable and exportable in various formats for easy sharing and analysis.

### 1.5.5 Integration:

The attendance system should integrate with other university systems, such as the student information system and course management system. The integration should facilitate the exchange of relevant data, ensuring accuracy and consistency across different platforms.

### 1.5.6 Accessibility:

The system should adhere to accessibility standards and guidelines to ensure that it is accessible to all users, including those with disabilities. The user interface should be designed with considerations for different accessibility requirements, such as screen readers and keyboard navigation.

### 1.5.7 Support and Maintenance:

The system should be accompanied by comprehensive support and maintenance services. This includes user training, technical support, and regular system updates to address any issues or vulnerabilities that may arise.

### 1.5.6 Performance Metrics:

The system should be evaluated based on performance metrics, such as response time, system uptime, and user satisfaction. Regular performance testing and monitoring should be conducted to identify and address any performance bottlenecks or usability concerns.

### 1.5.7 Conclusion:

The Supplementary Specification highlights additional features and requirements for the Attendance System, ensuring its effectiveness and usability in managing student attendance within the university. By incorporating these specifications, the system can meet the specific needs and expectations of the institution, contributing to improved attendance tracking and management processes.

## 1.6 Glossary

1. Attendance System: The software application designed to track and manage student attendance within a university environment.
2. **Admin:** An administrative user with elevated privileges who has access to system configuration, user management, and overall system administration.
3. **Teacher:** A user role representing the faculty member responsible for marking and managing student attendance for their assigned classes.
4. **Student:** An enrolled individual attending classes who is subject to attendance tracking and record-keeping within the system.
5. **Attendance Record:** A specific instance of attendance for a student in a particular class on a given date, indicating whether the student was present or absent.
6. **Absence:** A status indicating that a student was not present for a specific class or session.
7. **Attendance Percentage:** The calculated percentage of classes attended by a student for a specific course or overall attendance.
8. **Makeup Class:** A rescheduled class to compensate for a missed session, typically arranged due to cancellations or special circumstances.
9. **User Authentication:** The process of verifying the identity and credentials of a user before granting access to the attendance system.
10. **Data Encryption:** The technique used to convert sensitive attendance data into a secure and unreadable format to protect it from unauthorized access.
11. **Attendance Report:** A document or presentation displaying attendance data and statistics, such as attendance percentages, absences, and late arrivals for students, classes, or courses.

## 1.7 Risk List & Risk Management Plan

### 1.7.1 Risk List

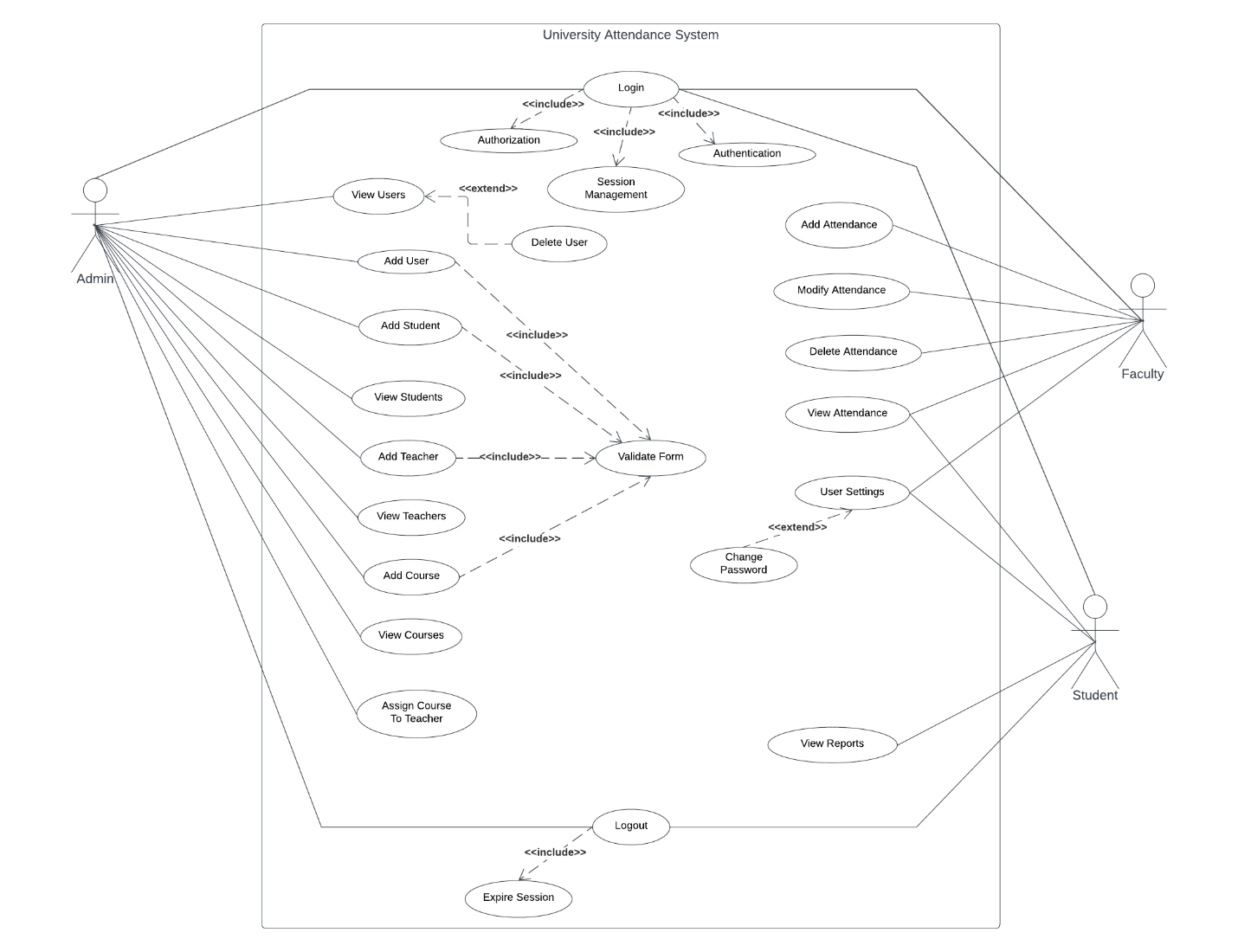
1. Data loss due to system failure or user error.
2. Inadequate system security leads to unauthorized access or data breaches.
3. Incorrect data input by users.
4. Insufficient capacity to handle large volumes of data.
5. Software bugs or errors leading to system crashes or malfunctions.
6. Inability to handle unexpected changes in scheduling or events.
7. Resistance from users to adopt the new system or change their habits.
8. Lack of resources (time, budget, personnel) to implement or maintain the system.
9. Lack of integration with other systems, such as student information systems or other systems.
10. Incomplete or inaccurate attendance data due to non-compliance or bypassing the system.
11. Insufficient user training and support resulting in user confusion or errors.

### 1.7.2 Risk Management Plan

1. Data loss: Implement regular backups and disaster recovery procedures.
2. System security: Use strong access controls, encryption, and regular security audits.
3. Incorrect data input: Implement data validation checks and provide user training.
4. Insufficient capacity: Architect the system to be scalable and conduct capacity planning.
5. Software bugs/errors: Conduct thorough testing and implement effective bug tracking.
6. Handling unexpected changes: Design the system to be flexible and have a change management process.
7. User resistance: Conduct extensive user training, involve users in design, and provide ongoing support.
8. Lack of resources: Thorough project planning and resource allocation.
9. Lack of integration: Establish collaboration, define protocols, and conduct thorough testing.
10. Incomplete/ inaccurate attendance data: Establish clear policies, implement auditing mechanisms, and provide training.
11. Insufficient user training/support: Develop comprehensive training materials, conduct sessions, and offer support channels.

# Chapter 2 Use Cases

## 2.1 Use Case Diagram



## 2.2 Brief Level Use Case

## 2.3 Fully Dressed Use Cases

### 2.3.1 Login

|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Login |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin, Faculty, Student |
| **Stakeholders and Interests** | * **System Admin:** wants to easily login to the system to access and manage the services offered efficiently. * **Teacher:** wants a seamless login process to access the system’s services. * **Students:** seek a straightforward and efficient login procedure, allowing them to access the system services. |
| **Pre-conditions** | * The user must be registered in the system either as a system admin, teacher, or a student. * The user possesses valid credentials, including email and password. |
| **Success Guarantee** | The users are logged into the system so they can perform within the system. |
| **Main success scenario** | 1. The user launches the system’s login screen within the application. 2. The system presents a login form to enter the email and password. 3. The user enters their registered email and password into the respective fields. 4. The system verifies the entered credentials against the stored user database. 5. The system identifies the user role (System Admin, Faculty, Student) based on the credentials provided. 6. The system grants appropriate system rights, permissions, and privileges to the user, depending on their role. 7. The system successfully logs the user into the system. 8. After successful login the system redirects the user to their respective dashboard, which is tailored to their role. |
| **Exceptions** | 1. If the user fails to provide both the email and password, the system will display an error message indicating that both fields are required. 2. If the user provides the password field but leaves the email field blank, the system will display an error message “Email is not valid, Provide valid email with at least 3 characters”. 3. If the user provides the email but leaves the password field blank the system will display an error message “Password is not valid, Provide valid password with at least 3 characters”. 4. If the email entered by the user does not follow the standard email format (e.g., missing '@' symbol, no domain name, etc.), the system should raise an exception indicating that the email is invalid. 5. If there is an error in connecting to the user database during the authentication process, the system will display an error message “Database connections issue please contact customer services”. 6. If the user enters invalid credentials (i.e., incorrect email and password) the system will display an error message “Invalid credentials, check your email and password”. |
| **Special Requirements** | 1. The login screen must be designed with a responsive layout to ensure the usability of various operating systems. 2. The system should be able to handle a large number of concurrent login requests to support scalability and accommodate peak usage periods. 3. The login process should be completed in a reasonable time frame to provide a smooth user experience. |
| **Technology and Data variations** | * The "Add User" feature should be compatible with various operating systems, such as windows and Linux. * The back end is developed using java programming language. * The database technology can vary, such as MySQL, PostgreSQL, MongoDB, or Oracle, to store user information. |
| **Frequency of occurrences** | The frequency of data entry for user addition can vary, ranging from occasional additions to bulk imports when onboarding multiple users. |
| **Miscellaneous** |  |

### 2.3.2 Add User

|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Add User |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin |
| **Stakeholders and Interests** | * **System Admin:** The Admin is responsible for managing user accounts creating new users, assigning roles, and maintaining the user database. |
| **Pre-conditions** | * **Authentication:** The admin must be authenticated and logged in to access the “Add User” functionality. Only authenticated users with admin privileges should have access to this feature. * **User Information Availability:** Before adding a new user, the admin should have all the required information about the user being added, such as their full name, username, email, and initial role assignment. This information is essential for creating the new user account accurately. * **Unique Email:** The admin must ensure that the username and email address for the new user are unique and not already used by another user in the system. Duplicate usernames or emails could lead to conflicts and potential login issues. * **Role Assignment:** The admin should determine the appropriate role and permissions to be assigned to the new user. Roles define the user's access rights and privileges within the system. * **User Policy Compliance:** The addition of new users must comply with any relevant user policies, rules, or regulations defined by the organization or system administrators. |
| **Success Guarantee** | Upon successful execution, the "Add User" functionality guarantees the creation of a unique, active user account with accurately assigned roles and compliance with system policies. |
| **Main success scenario** | 1. The administrator accesses the user management section in the system. 2. The administrator selects the option to add a new user. 3. The system presents a user creation form with necessary fields to enter user details, such as email, password, and role. 4. The administrator enters the required user information into the form. 5. The system validates the entered data for accuracy and completeness. 6. The system creates a new user account with the provided details and assigns the specified role. 7. The new user account is now active and can be used by the user to log into the system and access services based on their assigned role. |
| **Exceptions** | 1. If the user provides the password field but leaves the email field blank, the system will display an error message “Email is not valid, Provide valid email with at least 3 characters”. 2. **Invalid Email Format:** If the email entered by the user does not follow the standard email format (e.g., missing '@' symbol, no domain name, etc.), the system should display an error message “Email format is not valid, Provide a valid email address”. 3. If the admin provides the email but leaves the password field blank the system will display an error message “Password is not valid, Provide valid password with at least 3 characters”. 4. The system should verify that the role assigned to the new user (e.g., System Admin, Faculty, Student) is valid and allowed. If an invalid role is specified, the system should raise an exception and ask the administrator to select a valid role. 5. If there is an error in connecting to the user database during the authentication process, the system will display an error message “Database connections issue please contact customer services”. 6. **Duplicate User:** The system should check whether a user with the same email or username already exists in the database. If a duplicate user is found, the system should raise an exception and inform the administrator that the user already exists. |
| **Special Requirements** | 1. **Performance:** The "Add User" feature should respond quickly, and the user addition process should be completed within a reasonable time, even under high user loads. 2. **Security:** User data entered during the user addition process, such as passwords and personal information, should be securely stored and transmitted using encryption. Access to the "Add User" feature should be restricted to authorized administrators. |
| **Technology and Data variations** | The system should support authentication method such as email-password. |
| **Frequency of occurrences** | Frequent occurrence: During peak usage periods, such as the start of the semester, when many users may attempt to log in simultaneously. |
| **Miscellaneous** |  |

### 2.3.3 Add Student

|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Add Student |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin |
| **Stakeholders and Interests** | * **System Admin:** The Admin is responsible for managing students, adding new students. |
| **Pre-conditions** | * **Authentication:** The admin must be authenticated and logged in to access the “Add Student” functionality. Only authenticated users with admin privileges should have access to this feature. * **User Information Availability:** It is mandatory that the student should be first registered as user in the system along with their role as a student, the admin then adds the student by providing their necessary information. * **Unique Reg No:** The admin must ensure that each student is added with a unique Registration number. |
| **Success Guarantee** | Upon successful execution, the "Add Student" functionality guarantees the creation of a unique, active student is added to the system. |
| **Main success scenario** | 1. The administrator accesses the Admin Dashboard in the system. 2. The administrator selects the option to add a new student. 3. The system presents a student creation form with necessary fields to enter student details, such as reg no, name, father name, DOB, CNIC, phone number and email (the one the user is registered with). 4. The administrator enters the required student information into the form. 5. The system validates the entered data for accuracy and completeness. 6. The system creates a new student account with the provided details. |
| **Exceptions** | 1. **Missing Required Information:** If the admin attempts to submit the student creation form without filling in all the necessary fields (such as reg no, name, father name, DOB, CNIC, phone number, and email), the system should display an error message indicating that all required information must be provided. 2. **Invalid Date of Birth (DOB):** The system should validate the format and accuracy of the Date of Birth (DOB) entered by the admin. If an invalid or incorrect date format is provided, the system should display an error message asking the admin to enter the DOB in the correct format (e.g., DD/MM/YYYY). 3. **Invalid CNIC Format:** The system should verify that the CNIC (National Identity Card) number entered by the admin follows the correct format (e.g., 12345-1234567-1). If an invalid CNIC format is provided, the system should display an error message prompting the admin to enter a valid CNIC number. 4. **Invalid Phone Number Format:** The system should validate the format of the phone number provided by the admin. If an incorrect format is entered (e.g., missing area code, incorrect number of digits), the system should display an error message asking the admin to enter a valid phone number. 5. **Invalid Email Address:** The system should validate the format of the email address entered by the admin to ensure it follows the standard email format (e.g., user@example.com). If an invalid email address format is provided, the system should display an error message asking the admin to enter a valid email address. 6. **Duplicate Student:** The system should check whether a student with the same registration number, or email already exists in the database. If a duplicate student is found, the system should raise an exception and inform the administrator that the student already exists. 7. **Data Validation:** The system should perform data validation to ensure that the entered student details are accurate and valid. If any data inconsistencies or errors are detected, the system should display appropriate error messages, allowing the admin to correct the information before submitting the form. 8. **Database Connection Issue:** If there is an error in connecting to the database during the student creation process, the system should display an error message notifying the admin about the database connection issue and asking them to try again later or contact customer support for assistance. |
| **Special Requirements** | 1. **Performance:** The "Add Student" feature should respond quickly, and the user addition process should be completed within a reasonable time, even under high user loads. 2. **Security:** Student data entered during the user addition process, such as Reg no, name, CNIC, phone number, email, should be securely stored and transmitted using encryption. Access to the "Add Student" feature should be restricted to authorized administrators. |
| **Technology and Data variations** | * The system should support authentication methods such as email-password. * **User Interface:** The user interface for the student creation form can vary based on the technology used, such as a web-based form, a mobile app interface, or a command-line interface for administrative purposes. * **Platform Compatibility:** The system should be designed to work seamlessly on different platforms, such as Windows, macOS, Linux |
| **Frequency of occurrences** | 1. Semester Start: At the beginning of each academic semester or term, universities typically enroll new students, leading to a significant surge in student additions. During this period, the "Add Student" use case will be frequent as administrators add a large number of new students to the system. 2. Continuous Admissions: Some universities have continuous admissions, allowing students to join throughout the academic year. In such cases, the "Add Student" use case will occur frequently as new students are admitted on an ongoing basis. 3. Late Registrations: There may be instances of late registrations or special cases where students are added after the regular semester starts. These events will result in additional occurrences of the "Add Student" use case. 4. Student Transfers: Students might transfer from one course or program to another within the university, leading to updates or re-enrollment. This could cause additional occurrences of the "Add Student" use case. |
| **Miscellaneous** |  |

### 2.3.4 Add Course

|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Add Course |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin |
| **Stakeholders and Interests** | * **System Admin:** The Admin is responsible for managing course information, creating new courses, and maintaining the course database. |
| **Pre-conditions** | * **Authentication:** The admin must be authenticated and logged in to access the "Add Course" functionality. Only authenticated users with admin privileges should have access to this feature. * **Course Information Availability:** Before adding a new course, the admin should have all the required information about the course being added, such as course code, course name, and credit hours. This information is essential for accurately creating the new course. |
| **Success Guarantee** | Upon successful execution, the "Add Course" functionality guarantees the creation of a unique course with accurately provided details. |
| **Main success scenario** | 1. The administrator accesses the course management section in the system. 2. The administrator selects the option to add a new course. 3. The system presents a course creation form with necessary fields to enter course details, such as course code, course name, and credit hours (options: 2, 3, 4). 4. The administrator enters the required course information into the form. 5. The system validates the entered data for accuracy and completeness. 6. The system creates a new course with the provided details and adds it to the course database. 7. The new course is now available for students to enroll and for faculty to manage attendance. |
| **Exceptions** | 1. **Missing Required Information:** If the admin attempts to submit the course creation form without filling in all the necessary fields (course code, course name, or credit hours), the system should display an error message indicating that all required information must be provided. 2. **Invalid Credit Hours:** The system should verify that the chosen credit hours for the course (2, 3, or 4) are valid and allowed. If an invalid credit hour is specified, the system should raise an exception and ask the administrator to select a valid credit hour. 3. **Duplicate Course:** The system should check whether a course with the same course code already exists in the database. If a duplicate course is found, the system should raise an exception and inform the administrator that the course already exists. |
| **Special Requirements** | 1. **Performance:** The "Add Course" feature should respond quickly, and the course addition process should be completed within a reasonable time, even under high user loads. 2. **Security:** Course data entered during the course addition process should be securely stored and transmitted using encryption. Access to the "Add Course" feature should be restricted to authorized administrators. |
| **Technology and Data variations** | * The system should support authentication methods such as email-password. * **User Interface:** The user interface for the course creation form can vary based on the technology used, such as a web-based form, a mobile app interface, or a command-line interface for administrative purposes. * **Platform Compatibility:** The system should be designed to work seamlessly on different platforms, such as Windows, macOS, Linux |
| **Frequency of occurrences** | During course planning and setup periods, as new courses are added to the system, especially before the start of each academic semester or term. |
| **Miscellaneous** |  |

### 2.3.5 Add Teacher

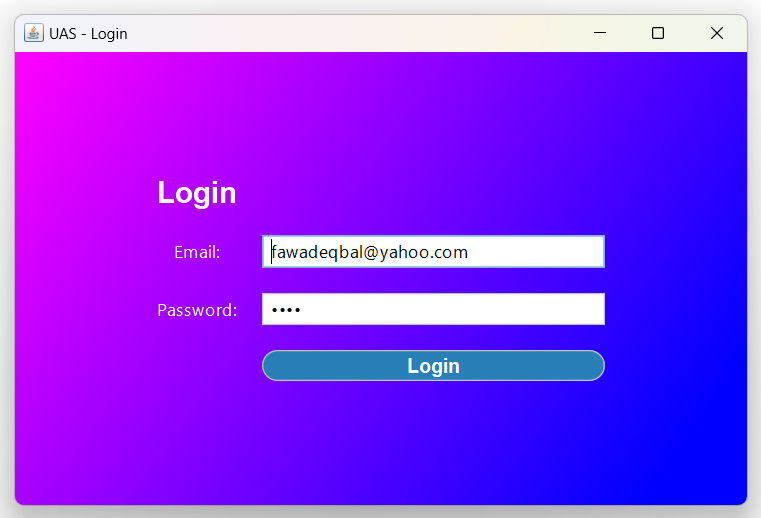
|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Add Teacher |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin |
| **Stakeholders and Interests** | * **System Admin:** The Admin is responsible for managing teacher information, creating new teacher profiles, and maintaining the teacher database. |
| **Pre-conditions** | * **Authentication:** The admin must be authenticated and logged in to access the "Add Teacher" functionality. Only authenticated users with admin privileges should have access to this feature. * **Teacher Information Availability:** Before adding a new teacher, the admin should have all the required information about the teacher being added, such as a unique ID, teacher name, phone number, and the email used for creating the teacher's user account. This information is essential for accurately creating the new teacher profile. |
| **Success Guarantee** | Upon successful execution, the "Add Teacher" functionality guarantees the creation of a unique teacher profile with accurately provided details. |
| **Main success scenario** | 1. The administrator accesses the teacher management section in the system. 2. The administrator selects the option to add a new teacher. 3. The system presents a teacher creation form with necessary fields to enter teacher details, such as a unique ID, teacher name, phone number, and the email used for creating the teacher's user account. 4. The administrator enters the required teacher information into the form. 5. The system validates the entered data for accuracy and completeness. 6. The system creates a new teacher profile with the provided details and adds it to the teacher database. 7. The new teacher profile is now available for assignment to courses and for managing attendance. |
| **Exceptions** | 1. **Missing Required Information:** If the admin attempts to submit the teacher creation form without filling in all the necessary fields (unique ID, teacher name, phone number, or email), the system should display an error message indicating that all required information must be provided. 2. **Invalid Email Format:** The system should verify that the email entered for the teacher follows the standard email format (e.g., user@example.com). If an invalid email format is provided, the system should display an error message asking the admin to enter a valid email address. 3. **Duplicate Teacher:** The system should check whether a teacher with the same unique ID or email already exists in the database. If a duplicate teacher is found, the system should raise an exception and inform the administrator that the teacher already exists. |
| **Special Requirements** | 1. **Performance:** The "Add Teacher" feature should respond quickly, and the teacher addition process should be completed within a reasonable time, even under high user loads. 2. **Security:** Teacher data entered during the teacher addition process should be securely stored and transmitted using encryption. Access to the "Add Teacher" feature should be restricted to authorized administrators. |
| **Technology and Data variations** | * The system should support authentication methods such as email-password. * **User Interface:** The user interface for the teacher creation form can vary based on the technology used, such as a web-based form, a mobile app interface, or a command-line interface for administrative purposes. * **Platform Compatibility:** The system should be designed to work seamlessly on different platforms, such as Windows, macOS, Linux |
| **Frequency of occurrences** | Frequent occurrence: During teacher hiring or assignment periods, as new teachers are added to the system, especially before the start of each academic semester or term. |
| **Miscellaneous** |  |

### 2.3.6 Assign Course to Teacher

|  |  |
| --- | --- |
| **Use Case Section** | **Comments** |
| **Use Case Name** | Assign Course to Teacher |
| **Scope** | University Attendance System |
| **Level** | User goal |
| **Primary Actor** | Admin |
| **Stakeholders and Interests** | * **System Admin:** The Admin is responsible for managing course assignments to teachers and maintaining the course and teacher database. |
| **Pre-conditions** | * **Authentication:** The admin must be authenticated and logged in to access the "Assigning Course to Teacher" functionality. Only authenticated users with admin privileges should have access to this feature. * **Course and Teacher Information Availability:** Before assigning a course to a teacher, the admin should have all the required information about the course and teacher, such as the course code, teacher ID, and their availability for assignment. |
| **Success Guarantee** | Upon successful execution, the "Assigning Course to Teacher" functionality guarantees that the selected course is assigned to the chosen teacher, forming a unique primary key for the course-teacher relationship. |
| **Main success scenario** | 1. The administrator accesses the course and teacher management section in the system. 2. The administrator selects the option to assign a course to a teacher. 3. The system presents a form with necessary fields to select the course and the teacher for assignment, such as course code and teacher ID. 4. The administrator enters the required course and teacher information into the form. 5. The system validates the entered data for accuracy and completeness, ensuring that the course code and teacher ID are valid and exist in the system. 6. The system creates an assignment record, associating the chosen course with the selected teacher, forming a unique primary key for the course-teacher relationship. 7. The course is now assigned to the teacher, and the teacher can now manage attendance and related activities for the assigned course. |
| **Exceptions** | 1. **Invalid Course Code or Teacher ID:** If the admin provides an invalid or non-existent course code or teacher ID, the system should display an error message indicating that the selected course or teacher does not exist in the database. 2. **Duplicate Assignment:** The system should check whether the combination of the chosen course and teacher is already assigned to each other. If a duplicate assignment is found, the system should raise an exception and inform the administrator that the course is already assigned to the teacher. |
| **Special Requirements** | 1. **Performance:** The "Assigning Course to Teacher" feature should respond quickly, and the assignment process should be completed within a reasonable time, even under high user loads. 2. **Security:** Course and teacher assignment data should be securely stored and transmitted using encryption. Access to the "Assigning Course to Teacher" feature should be restricted to authorized administrators. |
| **Technology and Data variations** | * The system should support authentication methods such as email-password. * **User Interface:** The user interface for the assignment of course to teacher form can vary based on the technology used, such as a web-based form, a mobile app interface, or a command-line interface for administrative purposes. * **Platform Compatibility:** The system should be designed to work seamlessly on different platforms, such as Windows, macOS, Linux |
| **Frequency of occurrences** | Frequent occurrence: During course planning and faculty assignments, as courses are assigned to teachers, especially before the start of each academic semester or term. |
| **Miscellaneous** |  |

# Chapter 3 UI Prototypes

## 3.1 Login UI



## 3.2 Admin Dashboard

A screenshot of a computer

Description automatically generated

## 3.3 Add User

A screenshot of a computer

Description automatically generated

## 3.4 View Users

A screenshot of a computer

Description automatically generated

## 3.5 Add Student

A screenshot of a computer

Description automatically generated

## 3.6 View Students

A screenshot of a computer

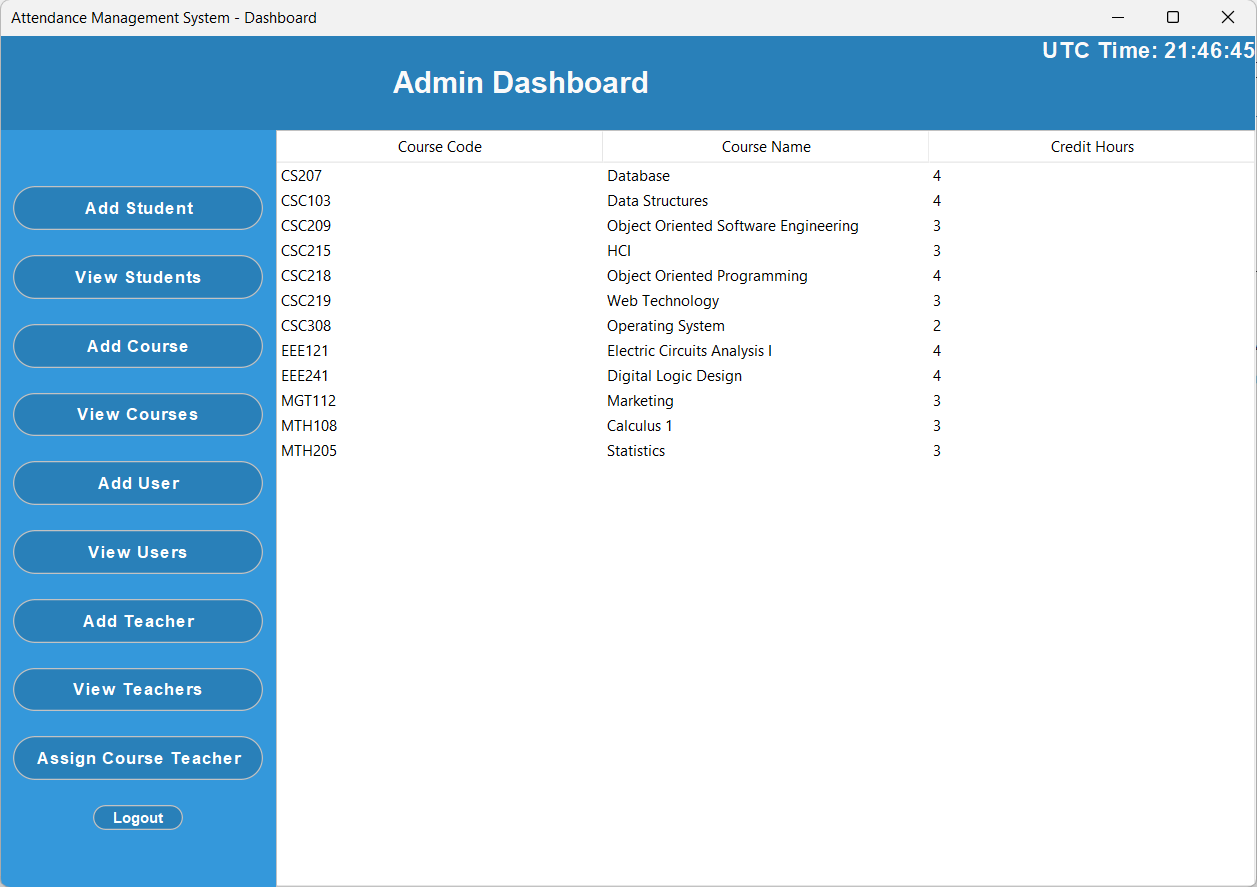
Description automatically generated

## 3.7 Add Course

A screenshot of a computer

Description automatically generated

## 3.8 View Courses



## 3.9 Add Teacher

A screenshot of a computer

Description automatically generated

## 3.10 View Teachers

A screenshot of a computer

Description automatically generated

## 3.11 Assign Course to Teacher

A screenshot of a computer

Description automatically generated

# Chapter 4 Domain Model

## 4.1 Domain Model

# Chapter 5 System Sequence Diagram (SSD)

## 5.1 SSD



# Chapter 6 Operation Contract (OC’s)

## 6.1 Operation Contract

### 6.1.1 Login

|  |  |
| --- | --- |
| **Contract ID** | **UC001 – Login** |
| **Operation** | verifyUser (user: UserDTO, responseObj:Response) |
| **Cross Reference** | This operation should occur within the “Login” use case. |
| **Preconditions** | 1. The `user` parameter must not be null. 2. The `user` object should contain valid user data, such as email and password. |
| **Postconditions** | 1. If the user data validation is successful, the system attempts to verify the user’s existence and credentials using the data access layer. 2. If the user is verified successfully, the user session is initialized, allowing the user to interact with the application using their authenticated credentials. 3. The `userDTO` object representing the user is set in the application session. |

### 6.1.2 Add User

|  |  |
| --- | --- |
| **Contract ID** | **UC002 – Add User** |
| **Operation** | addUser (userObj: UserDTO, responseObj: Response) |
| **Cross Reference** | This operation should occur within the “Add User” use case. |
| **Preconditions** | 1. The `userObj` parameter must not be null. 2. The `userObj` object should contain valid user data, such as email, password, and role. |
| **Postconditions** | 1. If the user data validation is successful (no validation errors), the system attempts to add the user to the user database using the data access layer. 2. If the user addition is successful, the responseObj will indicate the success status, and the new user will be added to the user database. 3. If the user data validation fails, the responseObj will contain error messages indicating the validation failures. |

### 6.1.3 Add Student

|  |  |
| --- | --- |
| **Contract ID** | **UC003 – Add Student** |
| **Operation** | addStudent (studentObj: StudentDTO, responseObj: Response) |
| **Cross Reference** | This operation should occur within the “Add Student” use case. |
| **Preconditions** | 1. The `studentObj` parameter must not be null. 2. The `studentObj` object should contain valid user data, such reg no, name, father name, DOB, CNIC, phone number, |
| **Postconditions** | 1. If the student data validation is successful (no validation errors), the system attempts to add the studnet to the student’s database using the data access layer. 2. If the student addition is successful, the responseObj will indicate the success status, and the new student will be added to the student’s database. 3. If the student data validation fails, the responseObj will contain error messages indicating the validation failures. |

### 6.1.4 Add Course

|  |  |
| --- | --- |
| **Contract ID** | **UC004 – Add Course** |
| **Operation** | addCourse (course: CourseDTO, responseObj: Response) |
| **Cross Reference** | This operation should occur within the “Add Course” use case. |
| **Preconditions** | 1. The `course ` parameter must not be null. 2. The `course ` object should contain valid user data, such as course code, course name, and credit hours. |
| **Postconditions** | 1. If the course data validation is successful (no validation errors), the system attempts to add the course to the course database using the data access layer. 2. If the course addition is successful, the responseObj will indicate the success status, and the new student will be added to the course database. 3. If the course data validation fails, the responseObj will contain error messages indicating the validation failures. |

### 6.1.5 Add Teacher

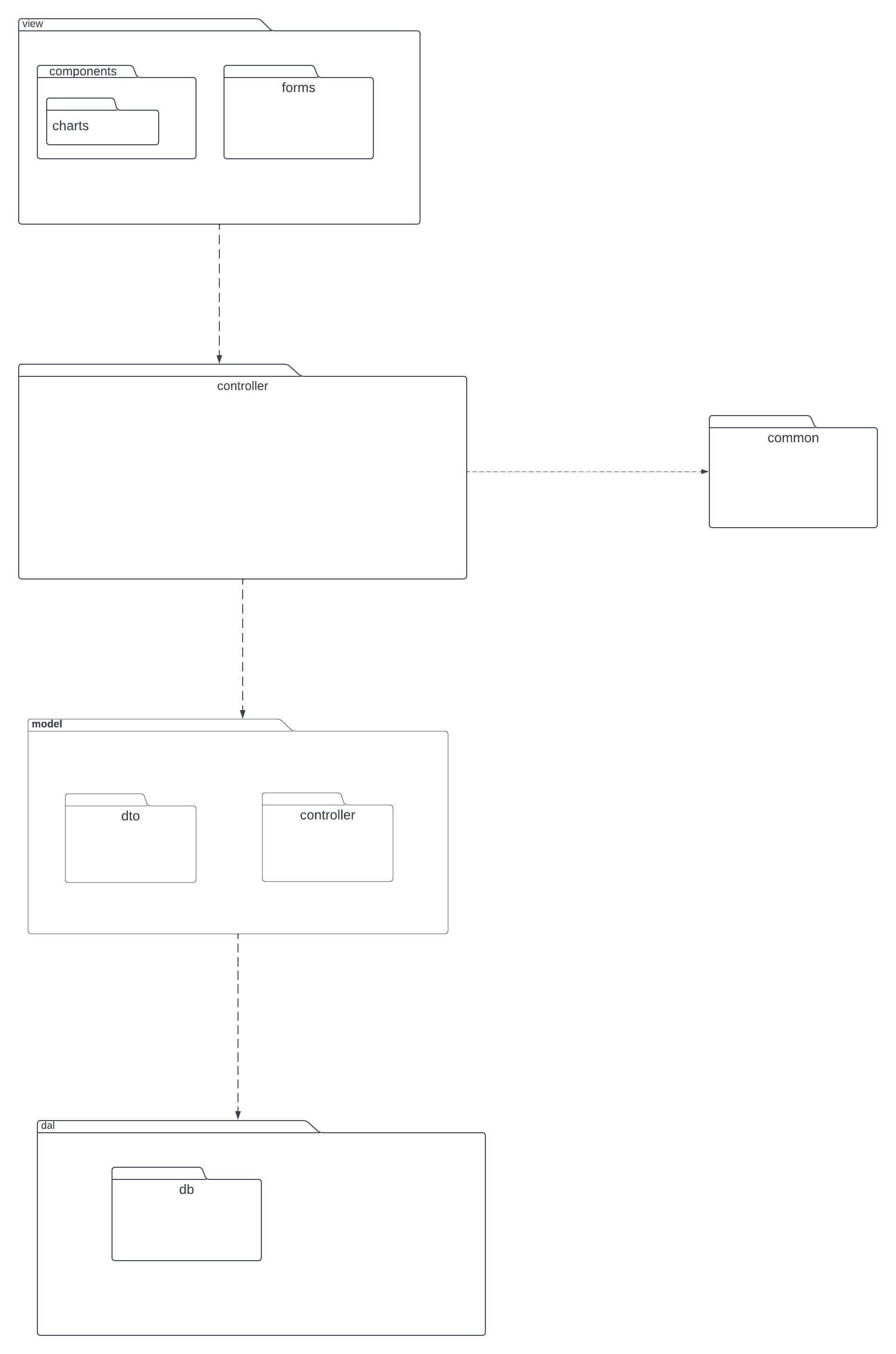
|  |  |
| --- | --- |
| **Contract ID** | **UC005 – Add Teacher** |
| **Operation** | addTeacher (teacher: TeacherDTO, responseObj: Response) |
| **Cross Reference** | This operation should occur within the “Add Teacher” use case. |
| **Preconditions** | 1. The `teacher ` parameter must not be null. 2. The `teacher ` object should contain valid user data, such as unique teacher id, name, phone number, and email. |
| **Postconditions** | 1. If the teacher data validation is successful (no validation errors), the system attempts to add the teacher to the teacher database using the data access layer. 2. If the teacher addition is successful, the responseObj will indicate the success status, and the new teacher will be added to the teacher database. 3. If the teacher data validation fails, the responseObj will contain error messages indicating the validation failures. |

### 6.1.6 Assign Course to Teacher

|  |  |
| --- | --- |
| **Contract ID** | **UC006 – Assign Course to Teacher** |
| **Operation** | assignCourseTeacher (teacher: TeacherDTO, course: CourseDTO, responseObj: Response) |
| **Cross Reference** | This operation should occur within the “Assign Course to Teacher” use case. |
| **Preconditions** | 1. The teacher parameter must not be null. 2. The course parameter must not be null. 3. The teacher object should contain valid teacher data, such as teacher id. 4. The course object should contain valid course data, such as course code. |
| **Postconditions** | 1. The system attempts to assign the specified course to the teacher in the course assignment database using the data access layer. 2. If the assignment is successful, the responseObj will indicate the success status, and the specified teacher will be associated with the specified course. 3. If the assignment fails due to data validation or other issues, the responseObj will contain error messages indicating the cause of the failure. |

# Chapter 7 Package Diagram

## 7.1 Package Diagram



# Chapter 8 Class Diagram

## 8.1 Class Diagram:

# 

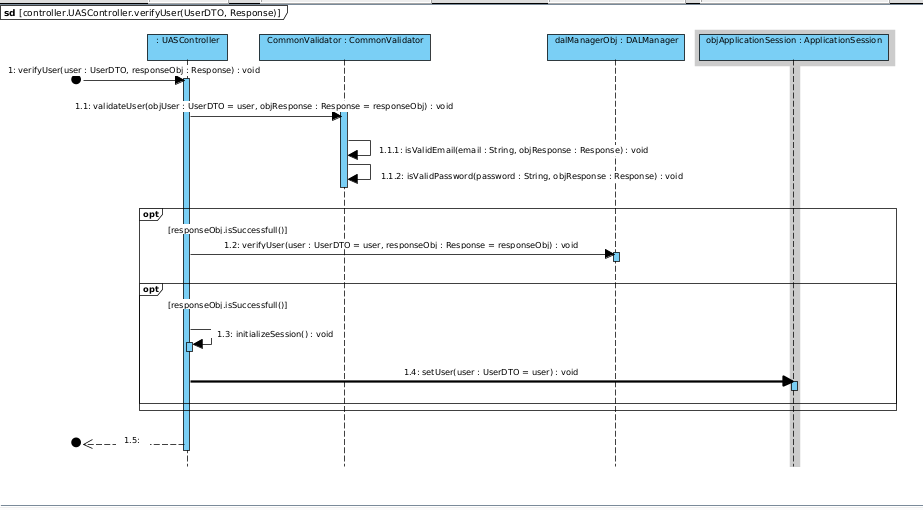
# Chapter 9 Interaction Diagram

## 9.1 Login

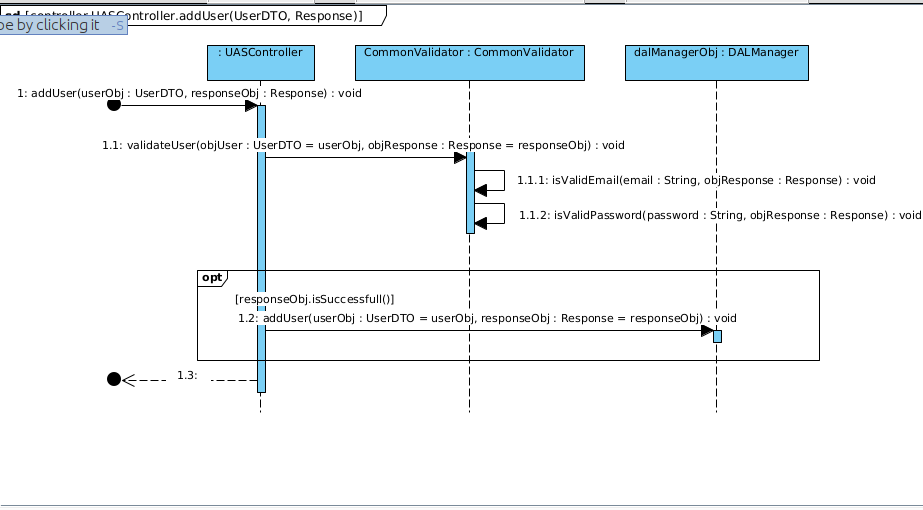


## 9.2 Sequence Diagram

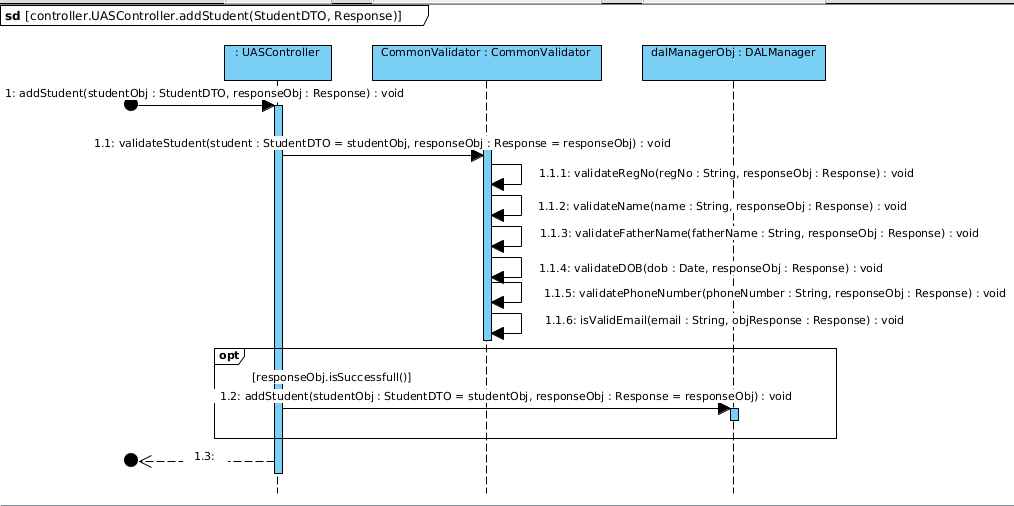
### 9.2.1 Login



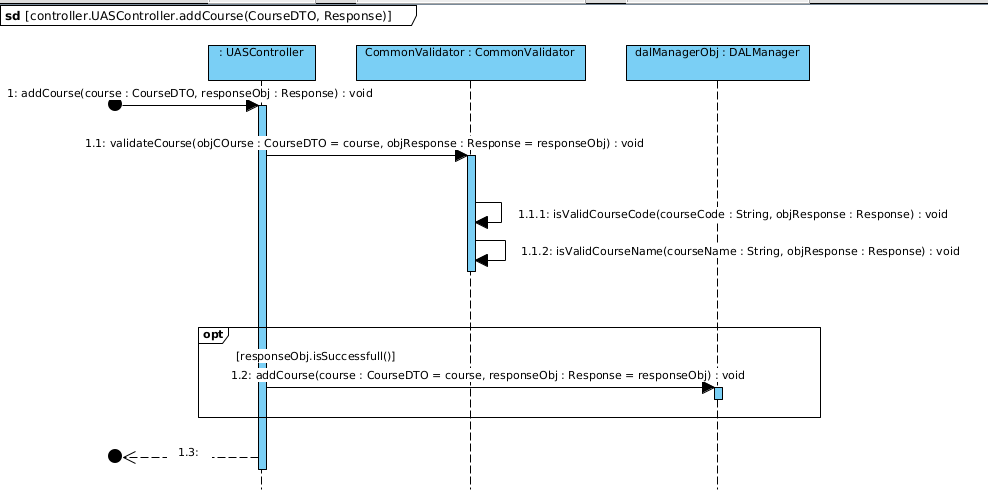
### 9.2.2 Add User



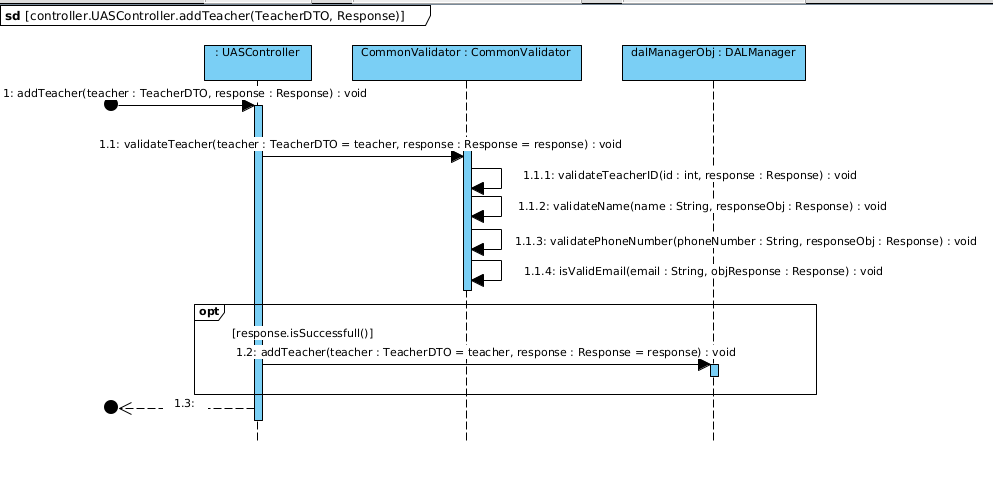
### 9.2.3 Add Student



### 9.2.4 Add Course



### 9.2.5 Add Teacher

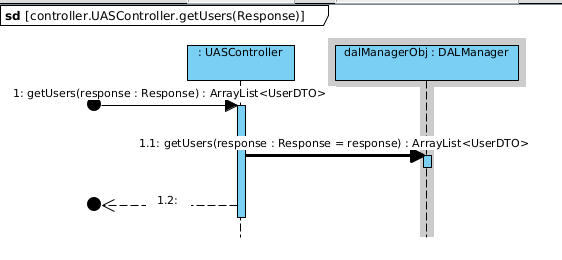


### 9.2.6 Assign Course to Teacher

A screenshot of a computer

Description automatically generated

### 9.2.7 View Users

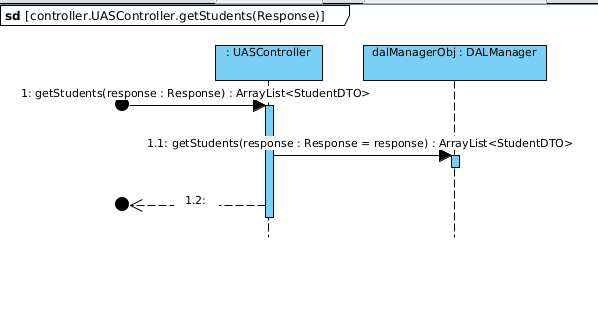


### 9.2.7 Delete User

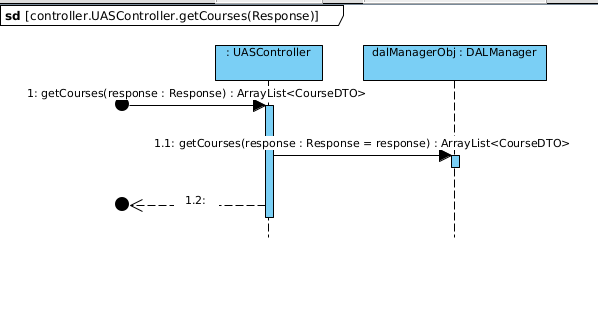
A screenshot of a computer

Description automatically generated

### 9.2.8 View Students



### 9.2.9 View Courses



### 9.2.10 View Teachers

