# WACHEMO UNIVERSITY

## COLLEGE OF ENGINEERING AND TECHNOLOGY
## DEPARTMENT OF SOFTWARE ENGINEERING

**PROJECT TITLE:** ONLINE CLASS SCHEDULING SYSTEM


**Group members**

| Name | ID |
|---|---|
| HIKMA MUSTEFA | WCU170132 |
| BEKELCHA NUREDIN | WCU175410 |
| KAMIL ERSIDO | WCU17D2702 |
| TSION AKLILU | WCU175475 |
| DAGMAWI LENCH | WCU1745338 |



Submission date 26/12/2025

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our advisor and the Department of Software Engineering at Wachemo University for their continuous guidance, support, and encouragement throughout this project.

We are also thankful to all instructors and staff members who provided us with valuable information and assistance during the development of this system.

Finally, we would like to thank each member of our project team for their cooperation, commitment, and hard work, which made this project successful.

# LIST OF FIGURES

# LIST OF TABLES

## ACRONYM

AI Artificial Intelligence
BR Business Rule
DB Database
DBMS Database Management System
ER Entity Relationship
ETB Ethiopian Birr
FK Foreign Key
FR Functional Requirement
G.C. Gregorian Calendar
GUI Graphical User Interface
IDE Integrated Development Environment
NF Normal Form
OS Operating System
SDLC Software Development Life Cycle
SQL Structured Query Language
UAT User Acceptance Testing
UML Unified Modeling Language
1NF First Normal Form
2NF Second Normal Form
3NF Third Normal Form

## EXECUTIVE SUMMARY

This project is entitled “Online Class Scheduling System for the Department of Software Engineering at Wachemo University.” The main purpose of the project is to design and develop a computerized system that helps in preparing, managing, and viewing class schedules efficiently.

Currently, the Department of Software Engineering uses a manual scheduling system, which is time-consuming and prone to errors such as class overlaps, conflicts in instructor availability, and difficulties in updating schedules. These problems affect the smooth academic process of the department.

To solve these issues, this project proposes an online system that allows the department head to manage schedules, assign instructors, and allocate classes, while students and instructors can easily view updated schedules. The system is designed using standard software engineering principles, UML diagrams, and a well-structured database.

The system is implemented using C++ for application logic and MySQL for database management. The development follows requirement analysis, design, implementation, and testing phases to ensure the system meets user needs.

The expected outcome of this project is a reliable, accurate, and easy-to-use scheduling system that improves efficiency, reduces manual work, and supports better academic planning in the department. The project also helps team members gain practical experience in system development and teamwork.

# CHAPTER ONE: INTRODUCTION

## CHAPTER TWO: REQUIREMENT ANALYSIS AND SPECIFICATIONS



## CHAPTER THREE: DATABASE DESIGN

# Chapter One: Introduction

## 1.1 BACKGROUND OF THE ORGANIZATION

Wachemo University began its academic functions in 2012 G.C and is located in Hossana town, about 232 km from Addis Ababa, Ethiopia, and is found in the central Ethiopia region. Currently, the university has 53 departments under 7 colleges and 2 schools. The university provides education at both undergraduate and postgraduate levels in regular, weekend, and night forms.

Under the College of Engineering and Technology, which has 13 departments, the Department of Software Engineering was established in 2016 G.C. Currently, the department has 3 lecture classrooms and 4 laboratory rooms that support both theoretical and practical learning processes. The department provides education for students in the Second Year – First Semester, Third Year – First Semester, and Third Year – Second Semester, while Fourth Year – Second Semester students participate in internship programs. In addition, the department offers various courses each semester for different batches based on the curriculum.

Currently, the Department of Software Engineering uses a manual class scheduling system. This system has led to several challenges, including class overlaps, scheduling conflicts, and time-consuming schedule preparation. Furthermore, manual record-keeping increases the risk of errors, inconsistencies, and data loss, making schedule management inefficient. Therefore, there is a need for an online class scheduling system to make the scheduling process faster, more efficient, and more accurate.

## 1.2 STATEMENT OF THE PROBLEM

The current manual class scheduling system used by the Department of Software Engineering faces several problems. Preparing schedules takes a long time and often results in class overlaps, room conflicts, and instructor time clashes. Any change in schedule requires rewriting and redistributing the whole timetable, which is inefficient. Manual record-keeping also increases the chance of human errors and loss of important data. These challenges make it difficult for the department to manage classes effectively and provide timely information to students and instructors.

## 1.3 OBJECTIVES OF THE PROJECT

### 1.3.1 GENERAL OBJECTIVE

The general objective of this project is to design and develop an online class scheduling system for the Software Engineering Department at Wachemo University.

### 1.3.2 SPECIFIC OBJECTIVE

- Identifying problems in the existing manual class scheduling process.
- Gathering system requirements from department heads and instructors.
- Analyzing the collected information to identify key system needs.
- Designing system models such as use case, ER, and sequence diagrams.
- Selecting an appropriate development model for the system.
- Developing the online class scheduling system based on the selected model.
- Testing the system to ensure it meets user requirements.
- Deploying the system for actual use in the department.

## 1.4 METHODOLOGIES

### 1.4.1 REQUIREMENT ELICITATION METHODOLOGY

Requirements will be collected through interviews and discussions with department heads and instructors.

### 1.4.2 REQUIREMENT ANALYSIS AND MODELING

The gathered requirements will be analyzed to remove ambiguities and inconsistencies. UML models such as use case diagrams, class diagrams, and sequence diagrams will be used to represent system behavior and structure.

### 1.4.3 SYSTEM IMPLEMENTATION METHODS

The Online Class Scheduling System will be developed using structured programming in C++, which ensures clear and organized system logic. MySQL will be used for database management to securely store academic data such as users, courses, instructors, and schedules.
The system will follow the Prototype development model, where an initial prototype will be developed to demonstrate the core functionalities of the system. This prototype will be reviewed by department heads and instructors to gather feedback. Based on the feedback, the system will be refined and enhanced through several iterations until the final system meets the required requirements.
The Prototype model was chosen because it allows early visualization of the system, supports requirement clarification, encourages user involvement, and enables easy modification during development. This approach improves system quality and reduces development risks.

## 1.5 SCOPE AND LIMITATION OF THE PROJECT

### 1.5.1 SCOPE

The project focuses on developing an online system that can:

- The system manages courses, instructors, students, and class schedules.
- Automatically generate class schedules without conflicts.
- Allow admins to update and manage schedules.
- Let instructors and students view schedules online.
- Store all scheduling data in a centralized database.

The system will mainly serve the Department of Software Engineering.

### 1.5.2 LIMITATION

- The system will initially work only for regular students of the department.
- Some features may remain basic due to time and resource constraints. This means the system will focus on essential scheduling functions first, while advanced improvements can be added in the future.
- The system depends on the availability of computers and internet access within the department
- Integration with other university systems is outside the scope of this project.

## 1.6 SIGNIFICANCE AND BENEFICIARIES OF THE PROJECT

- This project will benefit:
- Department Head: Simplifies creating, managing, and editing class schedules.
- Instructors: Access clear and updated schedules easily.
- Students: Get timely and accurate access to their class schedules.
- Department of Software Engineering: Improves planning and academic management.
- Team Members: Gain practical skills in teamwork and real software development.

## 1.7 FEASIBILITY ANALYSIS

### 1.7.1 OPERATIONAL FEASIBILITY

The system will be easy to use and fits well with current academic operations. Staff and students are already familiar with basic computer use, so adoption is expected to be smooth.

### 1.7.2 TECHNICAL FEASIBILITY

The required hardware and software tools such as computers, MySQL, and development environments are available. The team has basic programming knowledge to implement the system.

### 1.7.3 ECONOMIC FEASIBILITY

The system is economically feasible since its development will be carried out by students, which helps minimize development and implementation costs. Furthermore, the system reduces long-term expenses related to manual work, errors, and wasted time, proving its overall usefulness and benefits.

### 1.7.4 SCHEDULE FEASIBILITY

The project can be completed within the given academic timeframe by following a clear plan and dividing tasks among team members.

### 1.7.5 LEGAL FEASIBILITY

The proposed Online Class Scheduling System is legally feasible as it follows university rules and policies. It handles only academic data such as courses, instructors, and schedules, respects user privacy, and uses secure access control. Therefore, no legal or regulatory issues are expected.

### 1.7.6 OTHER FEASIBILITY

The project is socially acceptable and supports the university's goal of digital transformation.

## 1.8 RISK ASSESSMENT

| Risk Type | Description | Mitigation |
|---|---|---|
| Technical | Bugs, system errors, or | Proper testing, backups, and |
| Financial | Budget may not be enough for full | Careful planning and prioritizing |
| Operational | Users may struggle to adapt to the | Provide basic training and user- |
| Schedule | Delays in development or | Clear task division and regular |

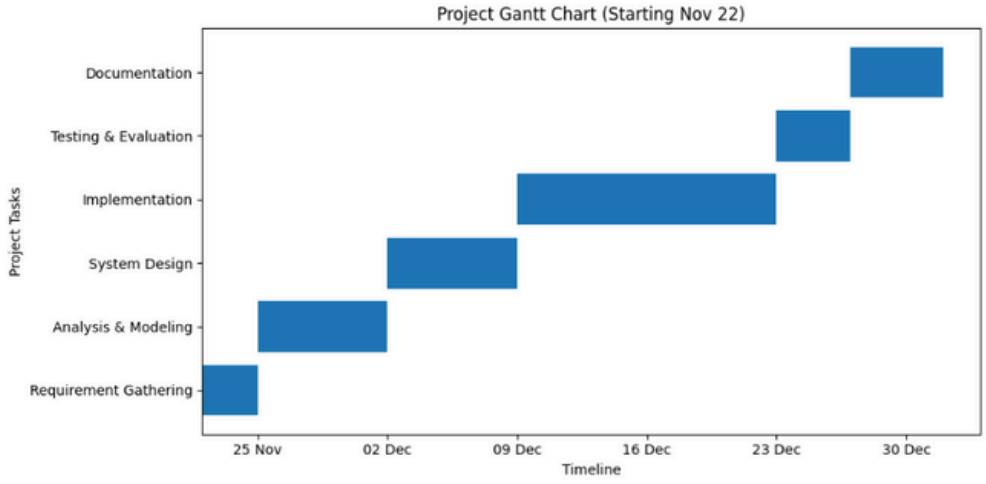
## 1.9 DEVELOPMENT TOOLS

The following tools will be used:

- Programming Language: **C++**
- Database: **SQL**
- Code Editor/IDE: **Visual Studio Code**
- Version Control: **Git**

## 1.10 WORK BREAKDOWN

### 1.10.1 PROJECT PLAN ACTIVITIES




These activities will be performed sequentially, with some overlap when needed.

### 1.10.2 PROJECT ORGANIZATION

The project organization defines the roles and responsibilities of the team members involved in the analysis, design, development, testing, and deployment of the Online Class Scheduling System. Each role ensures the successful completion of the project and smooth system operation.

#### I. PROJECT MANAGER

The Project Manager is responsible for planning, monitoring, and controlling the overall project activities. This role ensures the project is completed within the scheduled timeline and allocated resources. The Project Manager coordinates between team members, approves project deliverables, and communicates progress updates with the supervisor.

#### II. SYSTEM ANALYST

The System Analyst gathers and analyzes system requirements from users and stakeholders. This role designs system workflows related to class scheduling, instructor allocation, student enrollment, and timetable management. The System Analyst prepares system models such as use case diagrams, and ER diagrams, and ensures all requirements are clearly documented and traceable.

#### III. SYSTEM DESIGNER

The System Designer is responsible for designing the overall system architecture and user interface. This includes creating user-friendly interfaces for administrators, instructors, and students. The designer ensures consistency, usability, and ease of navigation across the system and refines designs based on feedback.

#### IV. SOFTWARE DEVELOPER

The Software Developer implements the functional modules of the Online Class Scheduling System. These modules include user management, course management, class scheduling, instructor assignment, timetable generation, and notification handling. The developer integrates all components, resolves system bugs, and ensures system reliability and performance.

#### V. TESTER / QUALITY ASSURANCE

The Tester is responsible for validating the system through unit testing, integration testing, and system testing. This role ensures that scheduling rules, conflict detection, and data input are accurate. The tester also participates in User Acceptance Testing (UAT) to confirm that the system meets user expectations.

#### VI. DEPLOYMENT AND MAINTENANCE TEAM

The Deployment and Maintenance Team installs the system on the required platforms and ensures proper configuration. This team provides basic user training and ongoing technical support, ensuring smooth system operation after deployment.

### 1.10.3 TEAM ORGANIZATION

| Role | Assigned Person(s) |
|---|---|
| 1, Project Manager | Hikma Mustefa |
| 2, System Analyst | Bekelcha Nuredin |
| 3, System Designer | Dagmawi Lencho, Bekelcha Nuredin |
| 4, Software Developer | Dagmawi Lencho |
| 5, System Tester | Kamil Ersido |
| 6, Deployment Team | Tsion Aklilu, Hikma Mustefa |

TABLE 1.1

## 1.11 BUDGET PLAN

| No | Budget Item | Description | Cost (Br) |
|---|---|---|---|
| 1 | Project Leader Allowance | Coordination, planning, | 60,000 |
| 2 | System Analyst | Requirement analysis, system | 55,000 |
| 3 | Back-end Developer | Core system logic, database | 90,000 |
| 4 | Front-end Developer | User interface design and | 70,000 |
| 5 | Tester & QA | System testing, bug fixing, | 45,000 |
| 6 | Documentation & Presentation | Writing reports, user manual, final | 40,000 |
| 7 | Tools & Resources | Internet, printing, software setup, | 35,000 |
| 8 | Training & Deployment | System setup, demo, basic | 30,000 |
| 9 | Contingency | For unexpected expenses | 50,000 |
| **Total** | | | **475,000 Br** |

**TABLE 1.2**

# Chapter Two: Requirement Analysis and Specifications

## 2.1. DESCRIPTION OF THE CURRENT SYSTEM

Currently, the Software Engineering Department at Wachemo University uses a manual class scheduling system. The department head prepares timetables using paper documents or simple spreadsheet files. Instructor availability, room allocation, and course information are collected manually.

Updating schedules is difficult and time-consuming. When changes occur (such as instructor replacement or room changes), the entire timetable must be reviewed again. Students and instructors are informed through notice boards or word of mouth, which often causes confusion and misinformation.

This manual approach results in:

- Class and room overlapping
- Instructor schedule conflicts
- High probability of human error
- Lack of centralized data storage

## 2.2 PLAYERS / ACTORS IN THE EXISTING SYSTEM

The existing system involves the following actors:

**Department Head**

- Prepares the timetable manually
- Assigns instructors, classes, and rooms
- Updates schedules when changes occur

**Instructor**

- Submits availability information
- Follows the printed timetable

**Student**

- Checks schedules from notice boards

## 2.3 USE CASE DIAGRAM FOR THE EXISTING SYSTEM

Since the current system is manual, the main use cases include:

- Create Schedule
- Assign Instructor
- Assign Class
- Update Schedule
- View Schedule

(The existing system does not support automated update checking or conflict detection.)

FIGURE 2.1

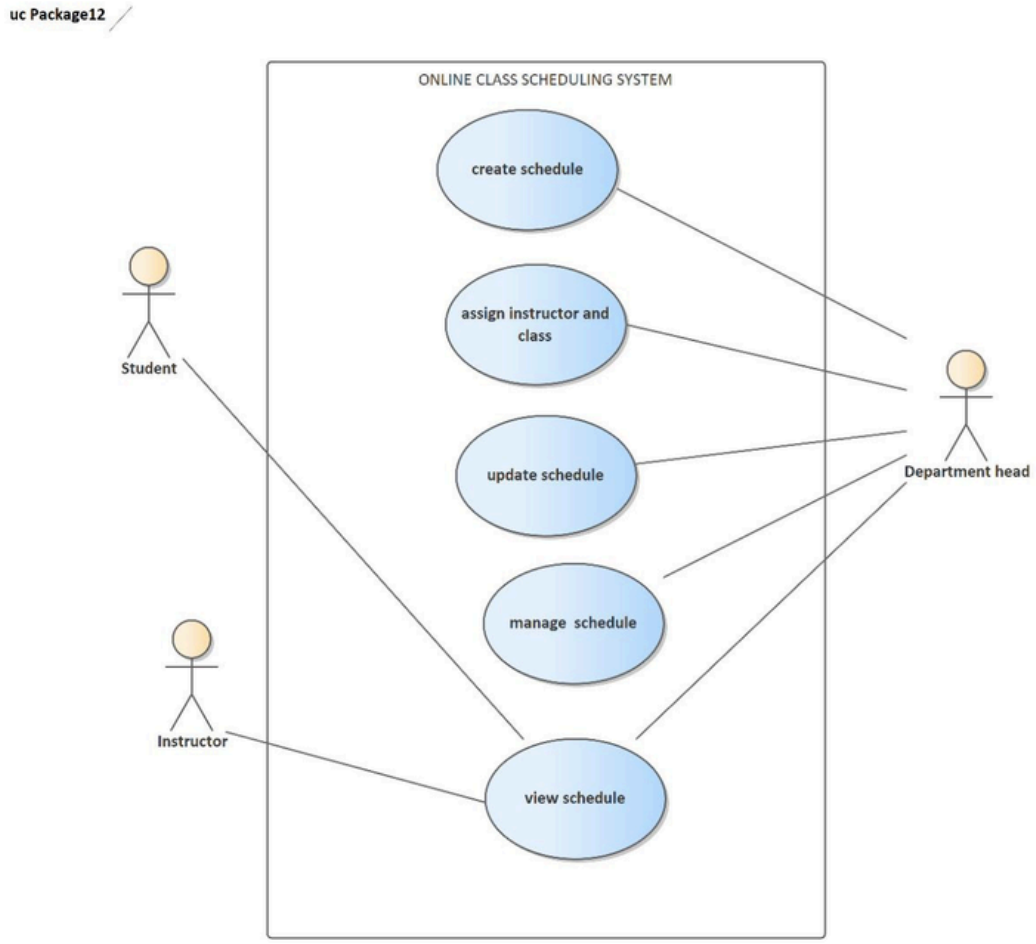



FIGURE 2.2

## 2.4 FORMS AND DOCUMENTS USED IN THE EXISTING SYSTEM

There is no form used to create the manual schedule.

ቀቸሞ ዩኒቨርሲቲ
ምህንድስና እና ቴክኖሎጂ ኮሌጅ
ሶፍትዌር ኢንጂነሪንግ ት/ክፍል

Wachemo University
College of Engineering &Technology
Department of Software Engineering

**Class Schedule for Software Engineering** **Date: 03/01/2018 E.C**

**Second Year – First Semester (Section-A)**

| Time | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| 2:00LT-5:00LT | Fundamentals of Software Engineering Ins:Senedu G. | Computer Programming II Ins:Alemayehu Sh. | Fundamentals of Database Systems Ins:Fozia A. | Discrete Mathematics and Combinatory TBA | Inclusiveness TBA |
| 5:00LT-6:00LT | **Mentoring** | **Online class** | **Mentoring** | **Online class** | **Mentoring** |
| **6:30LT-7:30LT Lunch Time** | | | | | |
| 8:00LT-11:00LT | Global Trends Ins: TBA | Fundamentals of Software Engineering**[Lab]** Ins:Senedu G | Computer Programming II**[Lab]** Ins:Alemayehu Sh. | Fundamentals of Database Systems**[Lab]** Ins:Fozia A. | Introduction to Economics TBA |

**Advisor: Alemayehu Sh.**

## 2.5 BUSINESS RULES OF THE EXISTING SYSTEM

- One course must be assigned to only one instructor.
- No two classes should use the same room at the same time.
- The timetable must match the department's academic calendar.
- Conflicts must be manually checked by the department head.

## 2.6 PROPOSED SYSTEM

### 2.6.1. OVERVIEW

The proposed system is an Online Class Scheduling System that automates timetable creation, updating, and checking.
It allows the Department Head to manage schedules digitally and enables students and instructors to view updated schedules in real time.
A key improvement is the Update Schedule (Update Check) use case, where the system validates changes before saving them to avoid conflicts.

### 2.6.2 BUSINESS RULES OF THE NEW SYSTEM

- The system must prevent overlapping schedules
- Instructor availability must be checked during updates
- A room cannot be assigned to multiple classes at the same time
- Only authorized users can update schedules
- All updates must be saved automatically in the database
- Updated schedules must be immediately visible to users

### 2.6.3 FUNCTIONAL REQUIREMENTS

**FR1**: Login
Users (Department Head, Instructor, Student) must log in before accessing the system.
**FR2**: Manage Schedule
The department head can manage all schedules.
**FR3**: Update Schedule (Update Check)
The system must allow the department head to:

- Modify class time
- Change assigned instructor
- Change room or section
- Validate updates before saving

**FR4**: Detect Overlap
The system checks:

- Instructor availability conflicts
- Room conflicts
- Time slot conflicts

**FR5**: View Schedule
Students and instructors can view updated schedules anytime.

### 2.6.4 NON-FUNCTIONAL REQUIREMENTS

**Security**

- Role-based access control
- Authentication required

**Performance**

- Schedule updates must be processed quickly
- Conflict checking must be automatic

**Usability**

- Simple and user-friendly interface
- Clear timetable presentation

**Reliability**

- No data loss during updates
- Accurate schedule information

## 2.6.5 ACTOR AND USE CASE IDENTIFICATION

**Actors**

- System Admin
- Department Head
- Instructor
- Student

**Main Use Cases**

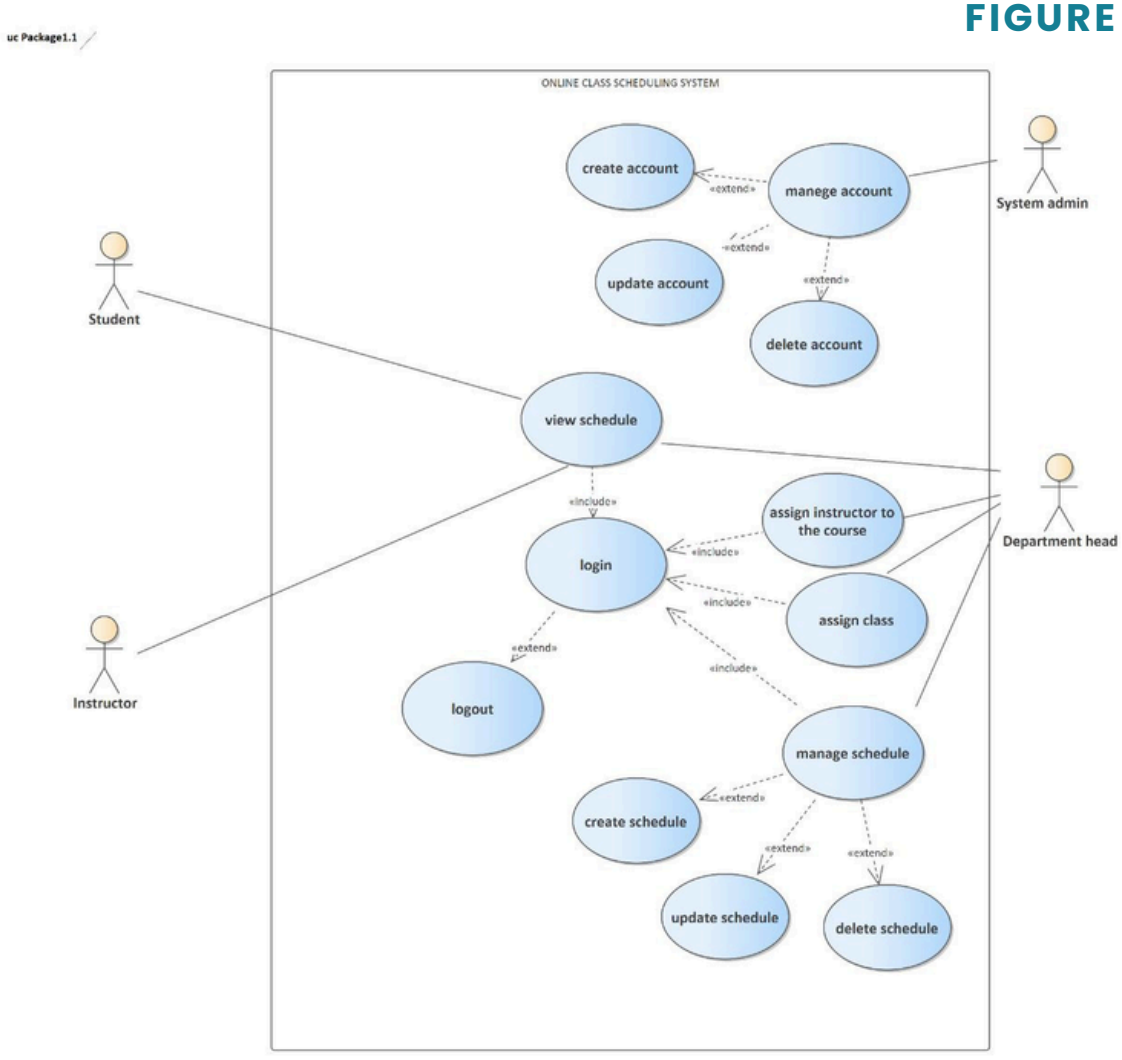
1. View schedule
2. Assign instructor to the  course
3. Assign class
4. Creat schedule
5. Update schedule
6. Delete schedule
7. Create account
8. Update account
9. Login
10. Logout

## 2.6.6. SYSTEM MODEL

We can describe our system using different modelling tools like use case diagrams, object models, and dynamic models. These diagrams help us clearly show the problem we want to solve and how the system should behave. Because they are visual, they make the requirements easier to understand compared to long paragraphs of text.

### 2.6.6.1. USE CASE MODEL

**FIGURE 2.3**




### 2.6.6.1.1. USE CASE DESCRIPTIONS

| Use Case Name | Create schedule | |
|---|---|---|
| **Actor** | Admin | |
| **Description** | The admin creates a new class schedule by entering course details, selecting time slots, and setting the instructor or room if required | |
| **Goal** | To successfully create and save a class schedule in the system | |
| **Precondition** | The admin must be logged into the system | |
| **Basic Flow of Event** | **Actor Action** | **System Response** |
| | Step 1: Admin enters username and password | System verifies login credentials |
| | Step 3: Admin selects “Create Schedule” option | System opens the schedule creation form |
| | Step 5: Admin enters schedule details (course, time, instructor, semester, etc.) | System validates the entered details |
| | Step 7: Admin submits the sched-ule | System saves the schedule and confirms creation |
| **Post Condition** | The system stores and displays the newly created schedule | |

TABLE 2.1

| | | |
|---|---|---|
| **Use Case Name** | **Assign instructor to course** | |
| **Actor** | Admin | |
| **Description** | The admin assigns an instructor to a course by selecting the instructor and the course | |
| **Goal** | To successfully assign an instructor to a course | |
| **Precondition** | The admin must be logged into the system | |
| **Basic Flow of Event** | **Actor Action** | **System Response** |
| | Step 1: Admin selects 'Assign Instructor' option | System displays an instactor course |
| | Step 2: Admin selects the instructor | System displays available courses |
| | Step 3: Admin selects the course | System validates selection |
| | Step 4: Admin submits the assignment | System confirms the assignment |
| | Step 5: Admin submits the | System conirms the assignment |
| **Post Condition** | The system updates and displays the course with the assigned instructor | |

TABLE 2.2

| | | |
|---|---|---|
| **Use Case Name** | **Assign class** | |
| **Actor** | Admin | |
| **Description** | The admin assigns a class to a course by selecting the class and the course | |
| **Goal** | To successfully assign an class to a course | |
| **Precondition** | The admin must be logged into the system | |
| **Basic Flow of Event** | **Actor Action** | **System Response** |
| | Step 1: Admin selects 'Assign Class' option | System displays an instructor form |
| | Step 2: Admin selects the class | System displays available courses |
| | Step 3: Admin selects the course | System validates selection |
| | Step 4: Admin submits the assignment | System confirms the assignment |
| | Step 5: Admin submits | System confirms the assignment |
| **Post Condition** | The system updates and displays the course with the assigned class | |

TABLE 2.3

| | |
|---|---|
| **Use Case Name** | **View Schedule** |
| **Actor** | Student, Instructor |
| **Description** | The student or instructor views the class schedule by accessing the schedule section of the system. |
| **Goal** | To allow the user to view the assigned class schedule. |
| **Precondition** | The user must be logged into the system. |

**Basic Flow of Event**

**Actor Action**

| **Actor Action** | **System Response** |
|---|---|
| Step 1: User selects the "View Schedule' option | System displays the schedule page |
| Step 2: User chooses schedule type (daily/weekly/semester) | System retrieves the requested schedule |
| Step 3: User views the schedule | System displays class details (course, instructor, time, room) |
| Step 4: User exits the schedule view | System returns to the main dashboard |

| | |
|---|---|
| **Post Condition** | The system successfully displays the schedule to the user. |

TABLE 2.4

<table>
<tr><td>Use Case Name</td><td colspan="2">Login</td></tr>
<tr><td>Actor</td><td colspan="2">Student, Instructor, Department Head, System Admin</td></tr>
<tr><td>Description</td><td colspan="2">A user logs into the system by entering valid credentials.</td></tr>
<tr><td>Goal</td><td colspan="2">To successfully authenticate and access the system.</td></tr>
<tr><td>Precondition</td><td colspan="2">The user must be logged into the system.</td></tr>
<tr><td rowspan="5">Basic Flow of Event</td><td colspan="2">Actor Action</td></tr>
<tr><td>Actor Action</td><td>System Response</td></tr>
<tr><td>Step 1: User enters username and passvord</td><td>System verifies the credentials</td></tr>
<tr><td>Step 2:<br>(a) If credentials are valid</td><td>System authenticates the user and grants access</td></tr>
<tr><td>(b) If credentials are invalid</td><td>System displays an error message and asks the user to retry</td></tr>
<tr><td></td><td>Step 4: User exits the schedule view</td><td>System returns to the main dashboard</td></tr>
<tr><td>Post Condition</td><td colspan="2">The user is logged into the system.</td></tr>
</table>

TABLE 2.5

| Use Case Name | Create Account | |
|---|---|---|
| Actor | System Admin | |
| Description | The system admin creates a new user account by filling in the required details for the user. | |
| Goal | To successfully create a new user account. | |
| Precondition | The admin must be logged into the system. | |
| Basic Flow of Event | Actor Action | |
| | Actor Action | System Response |
| | Step 1: Admin selects the 'Create Account' | System displays the account creation form. |
| | Step 2: Admin enters user details (username, password, role, etc.) | System validates the entered details. |
| | Step 3: Admin submits the new account | System creates the new account and confirms the creation. |
| | (b) If details are valid | System displays an error message prompting the admin to correct the information |
| | (b) If details are invalid | System displays an error message prompting the admin to correct the information |
| Post Condition | The new user account is successfully created in the system. | |

TABLE 2.6

| Use Case Name | Update Account | |
|---|---|---|
| Actor | System Admin | |
| Description | The system admin updates an existing user account by modifying the user's information. | |
| Goal | To successfully update an existing user account. | |
| Precondition | The admin must be logged into the system. | |
| Basic Flow of Event | Actor Action | |
| | Actor Action | System Response |
| | Step 1: Admin selects the 'Update Account' | System displays the account update form. |
| | Step 2: Admin searches and selects the user account to be modified | System displays the selected user information |
| | Step 3: Admin modifies the user details (password, role, etc.) | System validates the updated details |
| | (b) If details are valid | System saves the updated account information and confirms the update |
| | (b) If details are invalid | System displays an error message prompting the admin to correct the information |
| Post Condition | The user account is successfully updated in the system | |

TABLE 2.7

| | |
|---|---|
| **Use Case Name** | **Logout** |
| **Actor** | Any User |
| **Description** | The user logs out of the system, securely ending their session. |
| **Goal** | To securely end the user's session and log them out of the system. |
| **Precondition** | The user must be logged into the system. |

| | | |
|---|---|---|
| **Basic Flow of Event** | **Actor Action** | |
| | **Actor Action** | **System Response** |
| | Step 1: User selects the 'Logout' option | System terminates the session.<br>– The user is logged out of the system.<br>– System redirects the user to the login page. |
| **Post Condition** | The user's session is securely terminated and they are logged out of the system. | |

| | |
|---|---|
| **Post Condition** | **Actor Action** |
| | **Step 1**: User selects the 'Logout' option.<br>– The user is logged out of the system.<br>– System redirects the user to the login page. |

**TABLE 2.8**

| Use Case Name | Delete Schedule |
|---|---|
| Actor | Department Head |
| Description | The department head deletes an existing class schedule from the system. |
| Goal | To successfully delete a class schedule from the system. |
| Precondition | The department head must be logged into the system. |

| Basic Flow of Event | Actor Action | |
|---|---|---|
| | Actor Action | System Response |
| | **Step 1**: Department head selects the 'Delete Schedule' option. | System prompts the department head to confirm the deletion.<br>**(a) If department** head confirms the deletion.<br>– System deletes the class schedule.<br>**(b) If department** head cancels the deletion.<br>– System cancels the deletion operation and no changes are made. |
| **Post Condition** | The selected class schedule is successfully deleted from the system. | |

**TABLE 2.9**

| | |
|---|---|
| **Use Case Name** | **Delete Account** |
| **Actor** | System Admin |
| **Description** | The system admin deletes an existing user account from the system. |
| **Goal** | To successfully delete a user account from the system. |
| **Precondition** | The admin must be logged into the system. |

| | | |
|---|---|---|
| **Basic Flow of Event** | **Actor Action** | |
| | **Actor Action** | **System Response** |
| | Step 1: Admin selects the 'Delete Account' option. | System prompts the admin to confirm the deletion.<br>**(a) If admin** confirms the deletion.<br>– System deletes the user account.<br>**(b) If admin** cancels the deletion operation and no changes are made. |
| **Post Condition** | The selected user account is successfully deleted from the system. | |

| | |
|---|---|
| **Post Condition** | **Actor Action** |
| | Step 1: Admin selects the 'Delete Account' option.<br>– System prompts the admin to confirm the deletion.<br>– (b) If admin cancels the deletion.<br>– System cancels the deletion operation and no changes are made. |

**TABLE 2.10**

| | |
|---|---|
| **Use Case Name** | **Manage Account** |
| **Actor** | System Admin |
| **Description** | The system admin manages user accounts by updating account information or deactivating accounts. |
| **Goal** | To effectively manage user accounts by updating information or deactivating accounts. |
| **Precondition** | The admin must be logged into the system. |

| | | |
|---|---|---|
| **Basic Flow of Event** | **Actor Action** | |
| | **Actor Action** | **System Response** |
| | Step 1: Admin selects the 'Manage Account' option. | System displays the account management interface.<br>**(a)** Admin updates user information (username, password, role, etc.).<br>**(b)** Admin deactivates a user account.<br>– System deactivates the selected user account.<br>**(c) If** entered information is invalid.<br>– System displays an error message prompting the admin to correct the information. |
| **Post Condition** | The user account is successfully updated or deactivated based on the admin's actions. | |

**TABLE 2.11**

| | |
|---|---|
| **Use Case Name** | **Manage Schedule** |
| **Actor** | Department Head |
| **Description** | The department head manages class schedules by creating, updating, or deleting them within the system. |
| **Goal** | To effectively manage class schedules within the system, including creating, updating, or deleting schedules. |
| **Precondition** | The department head must be logged into the system. |

| | | |
|---|---|---|
| **Basic Flow of Event** | **Actor Action** | |
| | **Actor Action** | **System Response** |
| | Step 1: Department head selects the 'Manage Schedule' option. | System displays the schedule management interface.<br>(a) System creates the new schedule.<br>(b) Department head updates an existing schedule.<br>(c) Department head deletes a schedule.<br>(d) If entered information is invalid.<br>– System displays an error message prompting the department head to correct the information. |
| **Post Condition** | Class schedules are successfully managed based on the department head's actions. | |

| | |
|---|---|
| **Post Condition** | Class schedules are successfully managed based on the department head's actions. |

**TABLE 2.12**

| | |
|---|---|
| **Use Case Name** | **Update Schedule** |
| **Actor** | Department Head |
| **Description** | The department head updates an existing class schedule by modifying schedule details such as time, date, course, or instructor. |
| **Goal** | To keep the class schedule accurate and up to date. |
| **Precondition** | The department head must be logged into the system, and the schedule must already exist. |

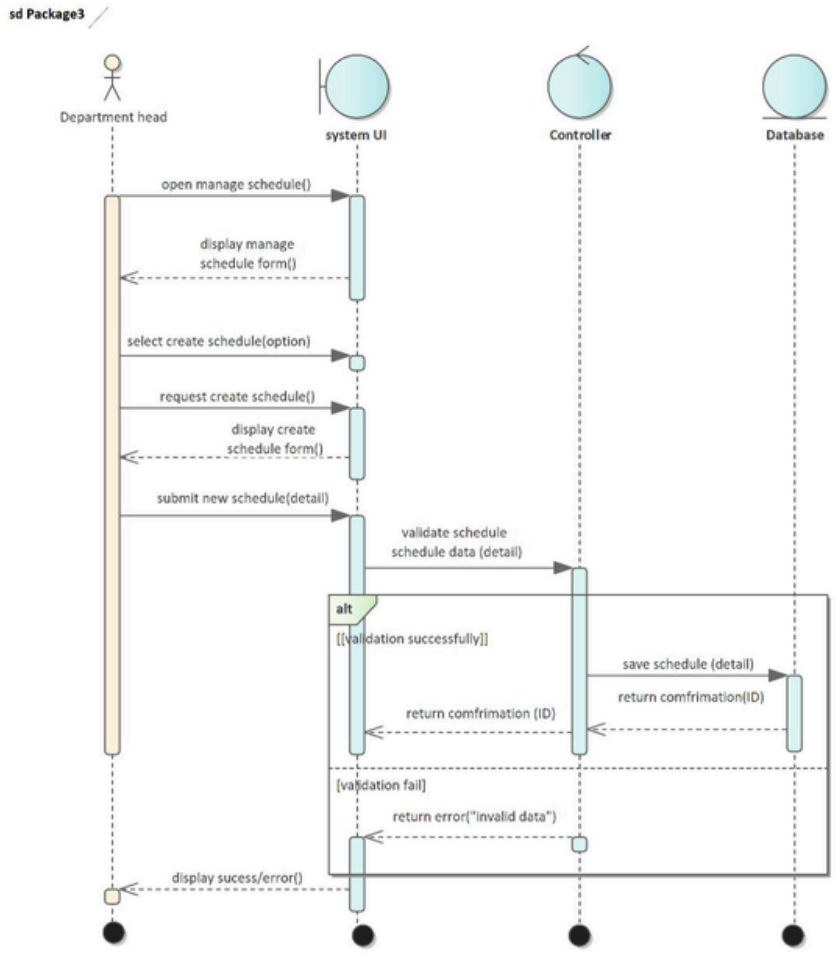
| **Basic Flow of Event** | **Actor Action** | |
|---|---|---|
| | **Actor Action** | **System Response** |
| | Step 1. Department head selects the 'Manage Schedule" option. | System displays the schedule management interface. |
| | Step 2. Department head selects a schedule to update. | System displays the selected schedule details. |
| | Step 3. Department head edits schedule information (time, date, instructor, etc.). | System validates the entered information.<br>**(a)** If information is valid, system updates the schedule and shows confirmation. |
| | Step 4. Department head submits the changes. | **(b)** If entered information is invalid, system displays an error message prompting correction.<br>**(b)** If entered information is invalid, system displays an error message prompting correction. |
| **Post Condition** | The schedule is successfully updated in the system. | |

**TABLE 2.13**

## 2.6.6.2. SEQUENCE DIAGRAM

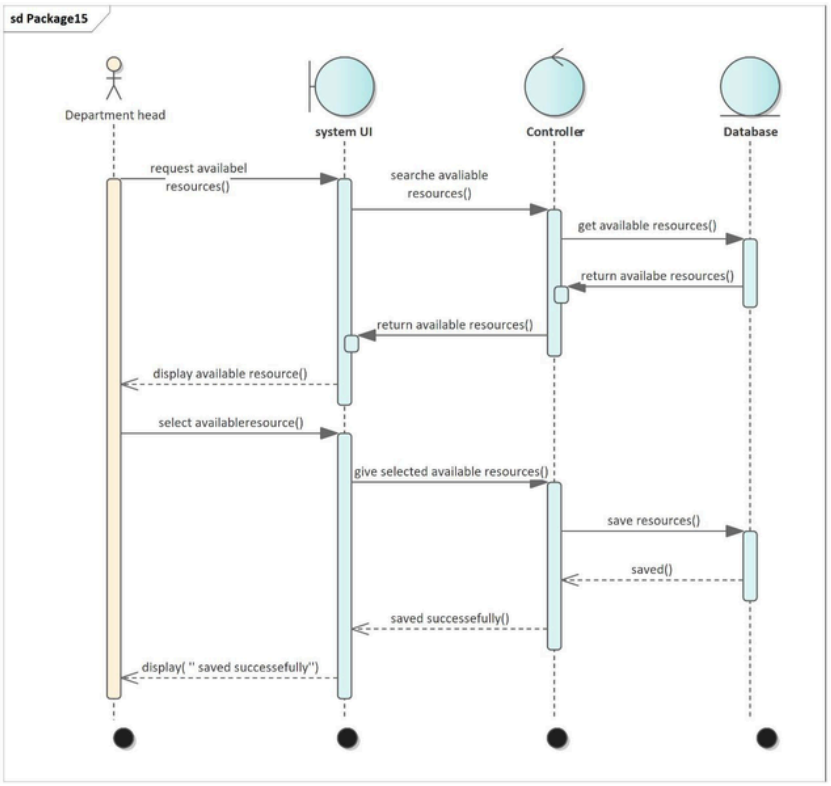
### 1. CREATE SCHEDULE

FIGURE 2.4



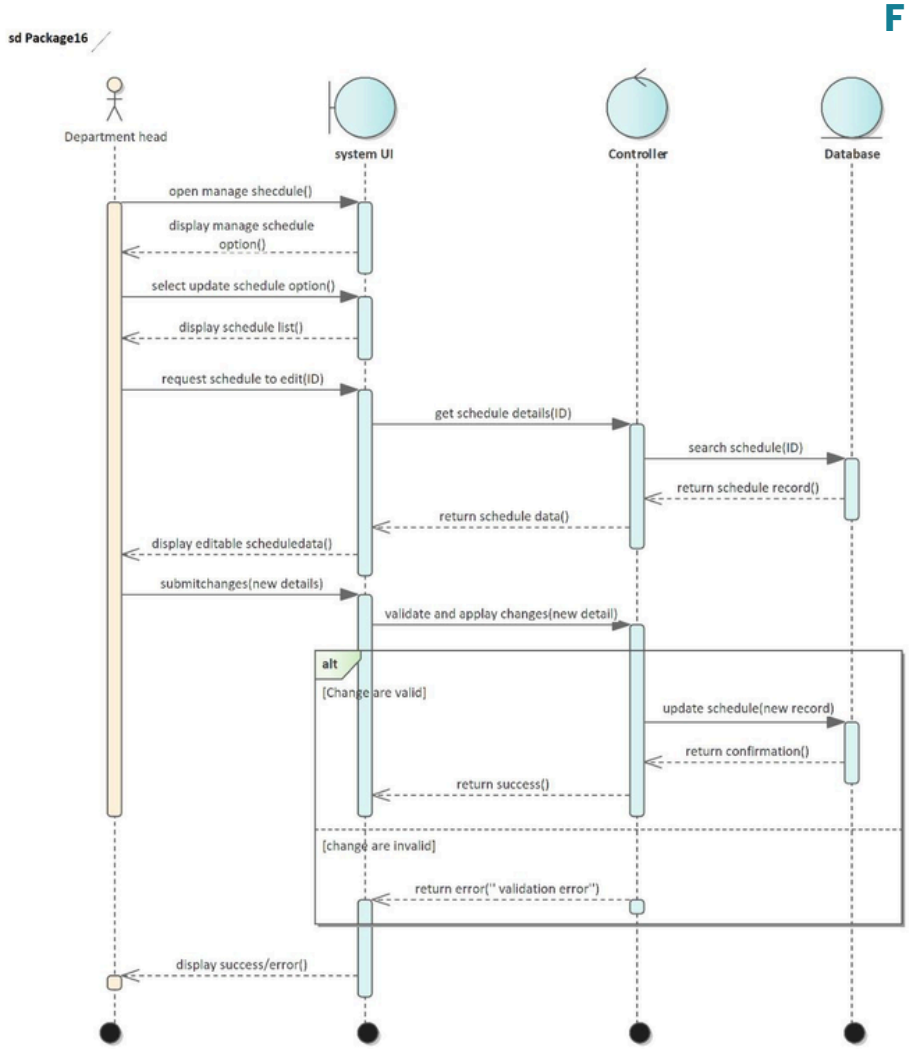

### 2. ASSIGN CLASS

FIGURE 2.5



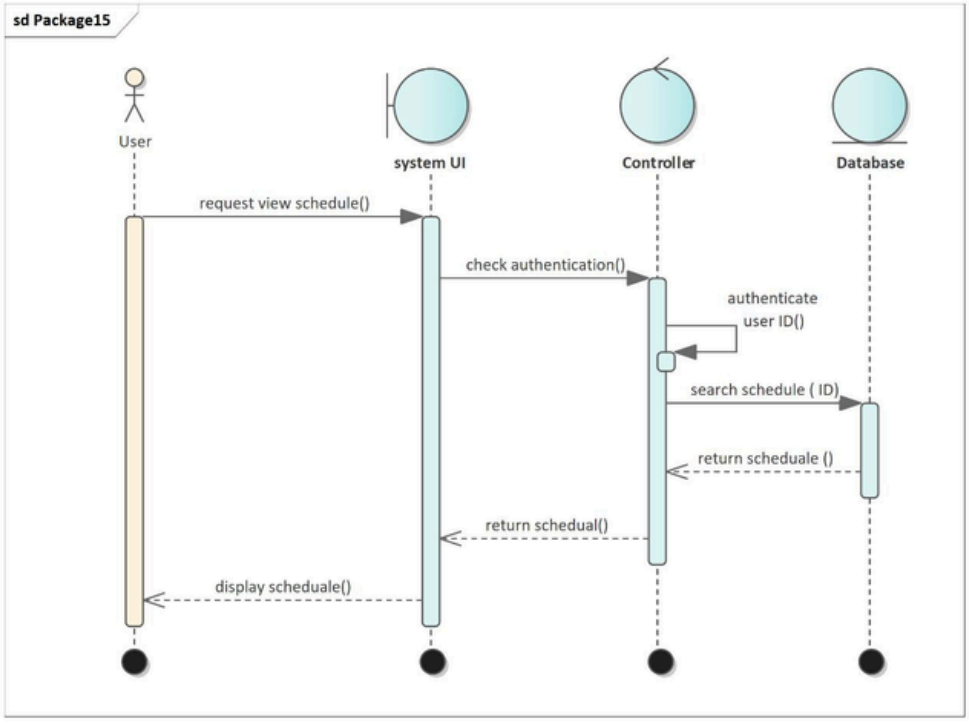
## 3. UPDATE SCHEDULE

FIGURE 2.6



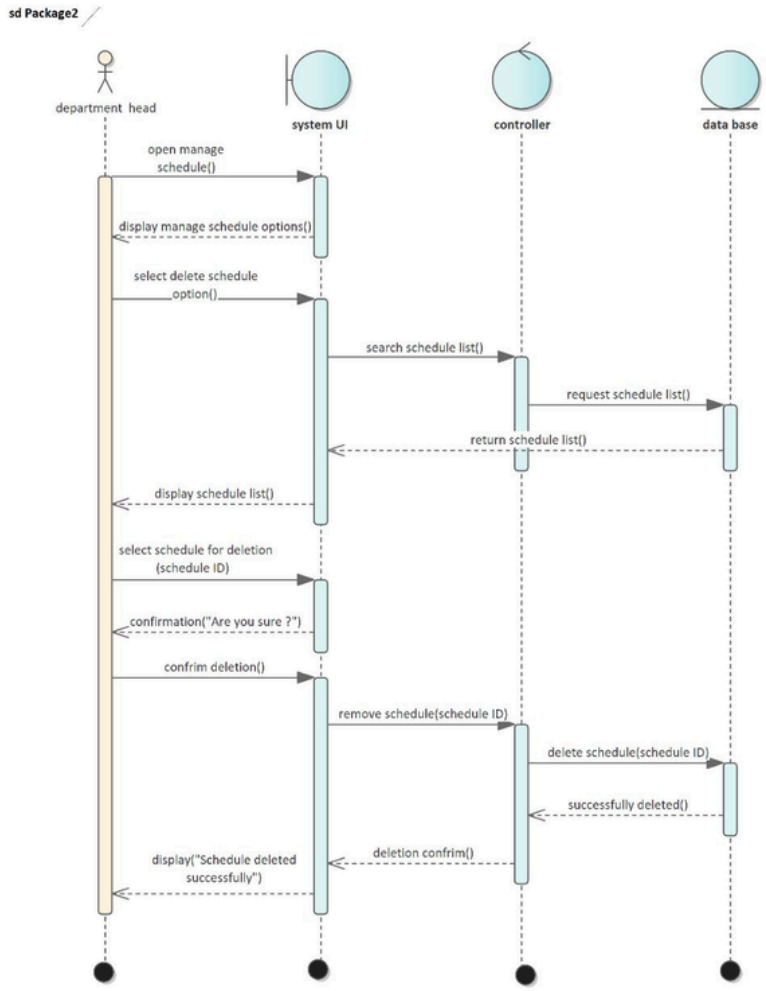

## 4. VIEW SCHEDULE

FIGURE 2.7



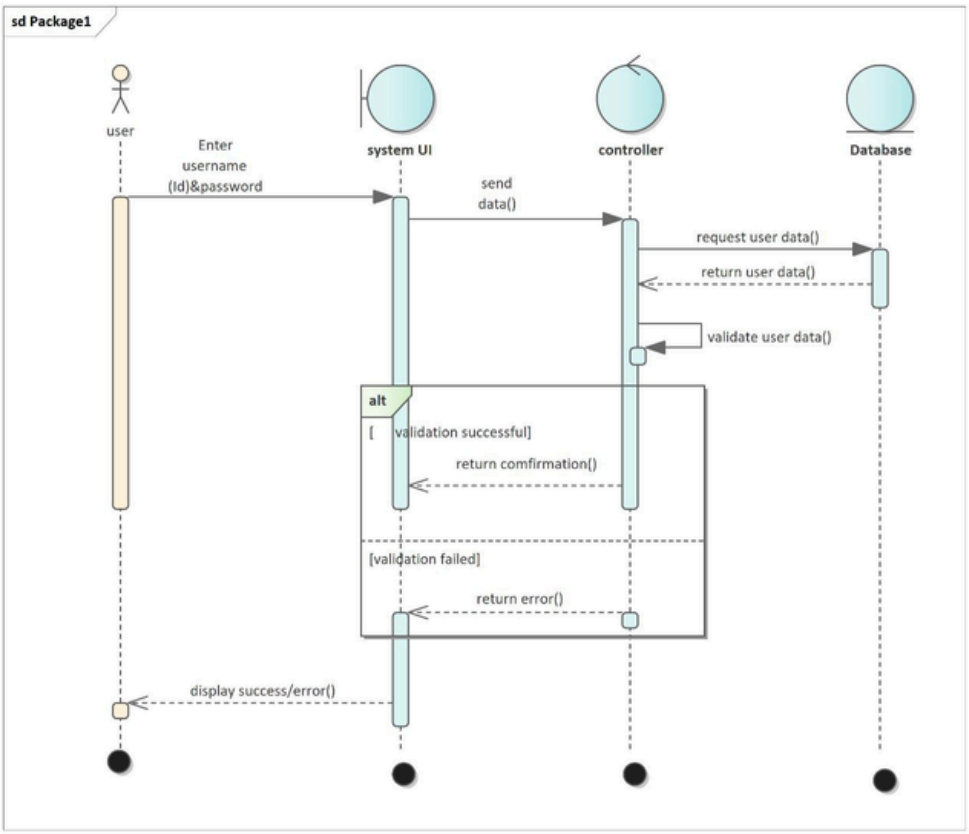
## 5. DELETE SCHEDULE

FIGURE 2.8



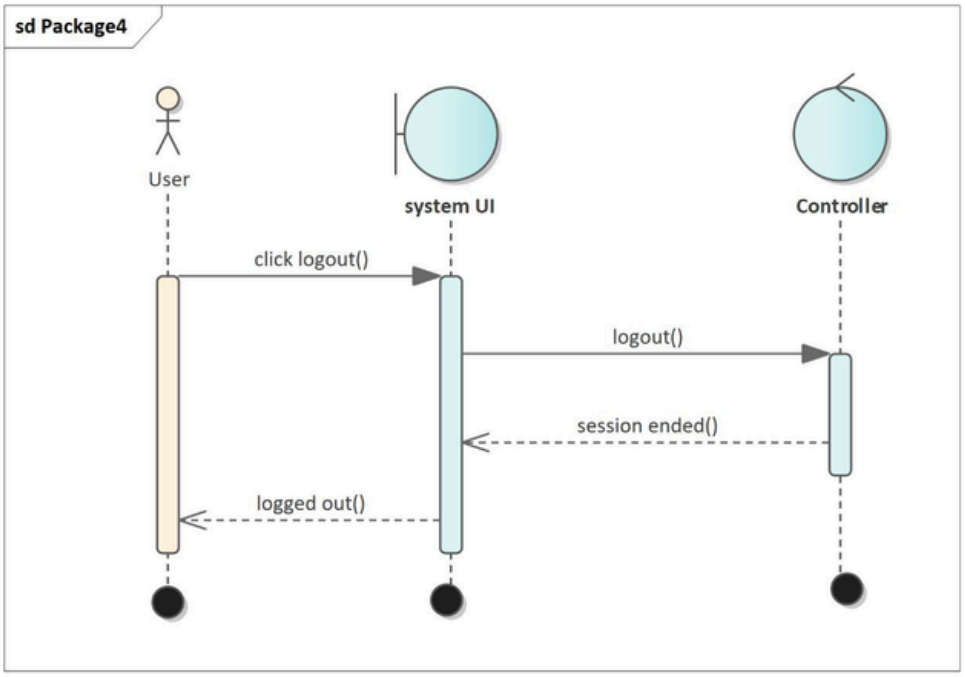

## 6. LOGIN

FIGURE 2.9



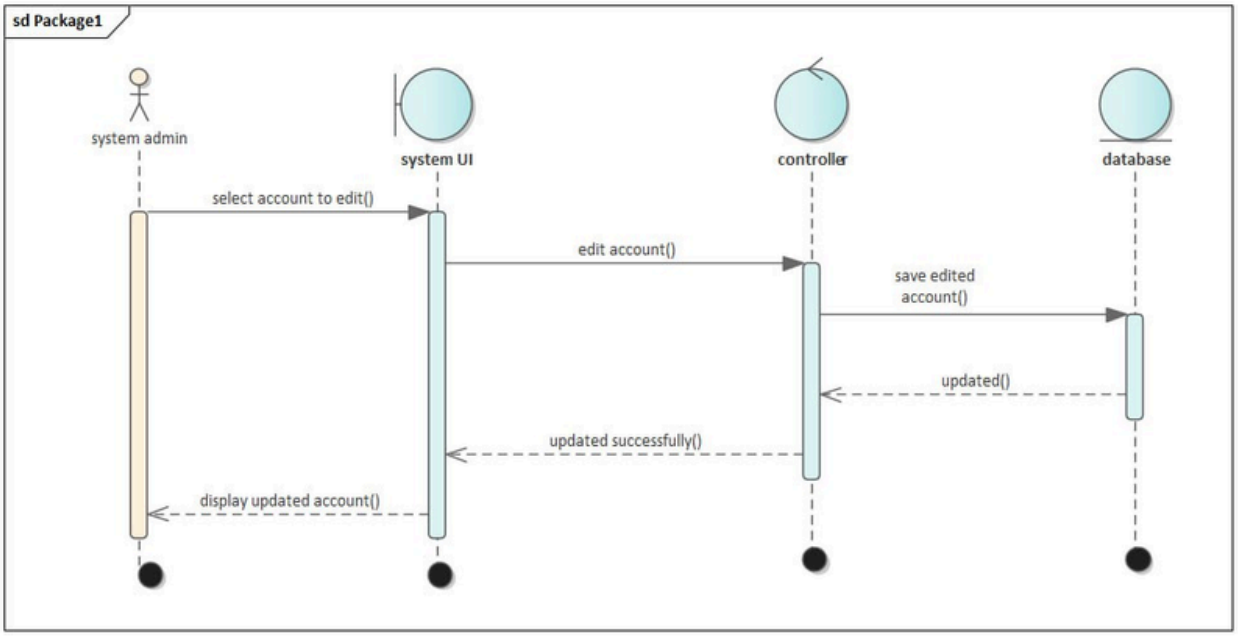
## 7, LOGOUT

FIGURE 2.10



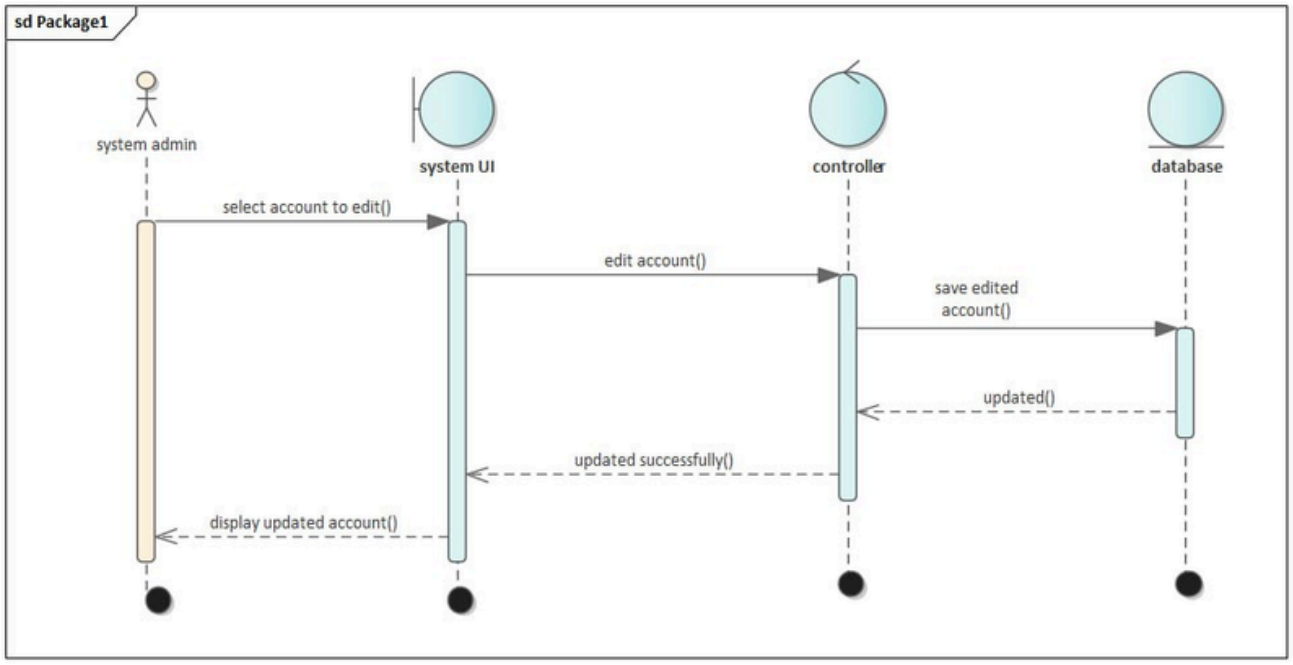

## 8, UPDATE ACCOUNT

FIGURE 2.11



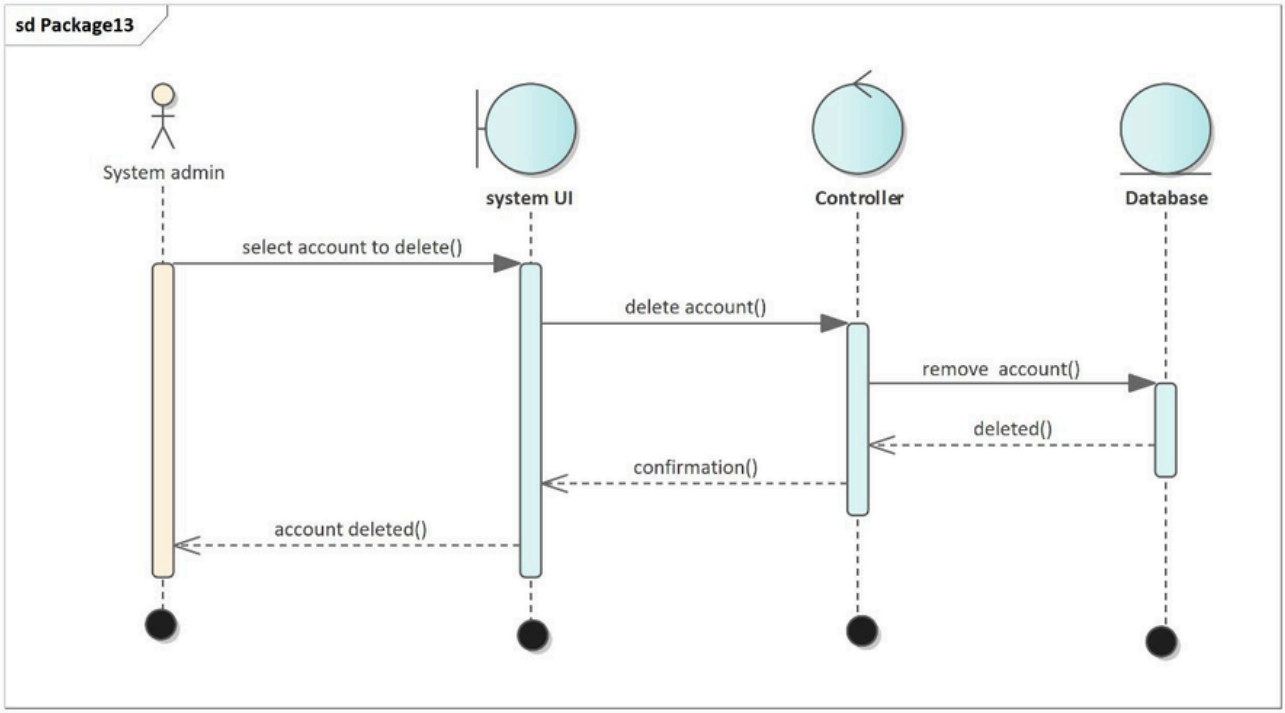
## 9, MANAGE SCHEDULE

FIGURE 2.12




## 10, DELETE ACCOUNT

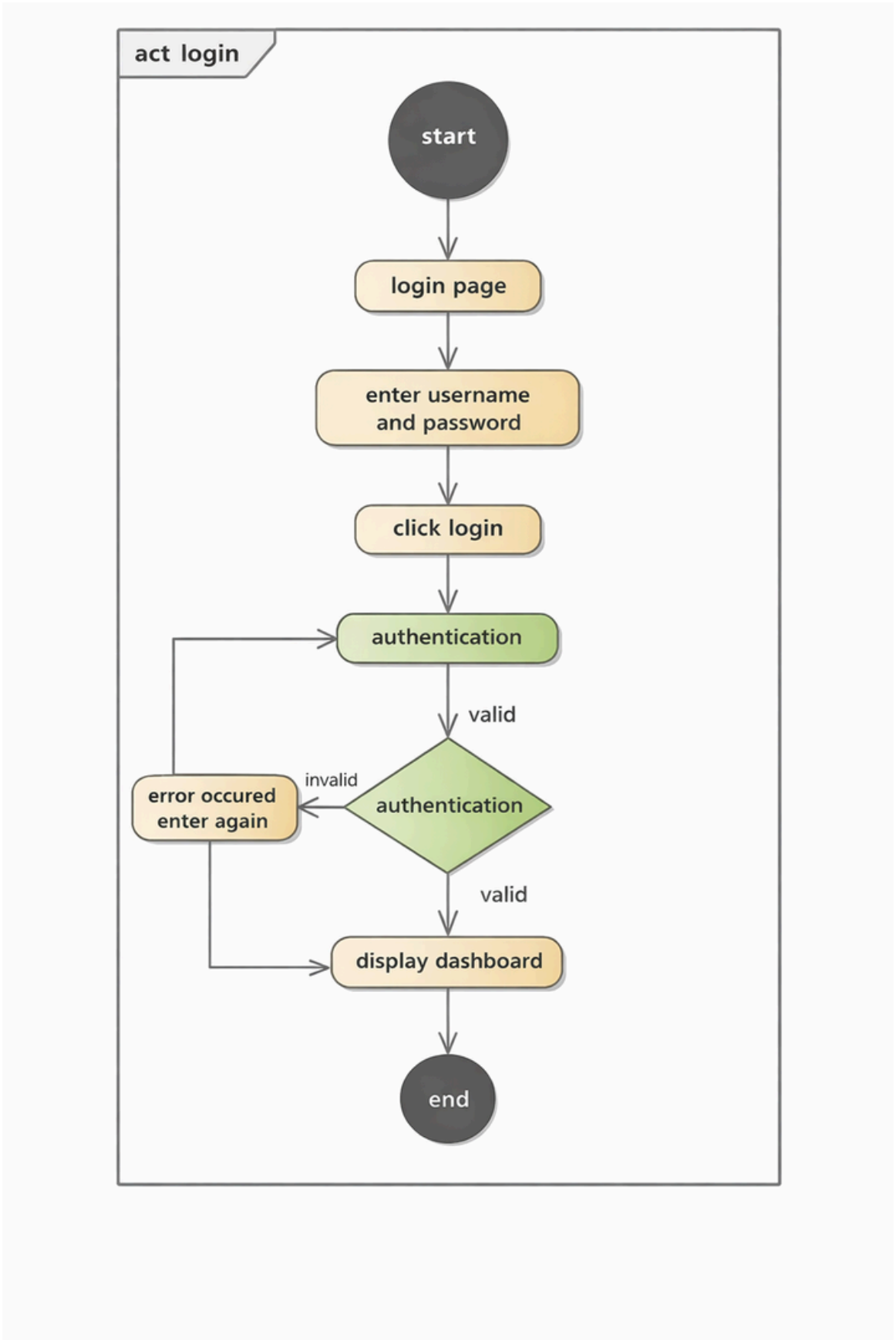
FIGURE 2.13



## 2.6.6.3. ACTIVITY DIAGRAM

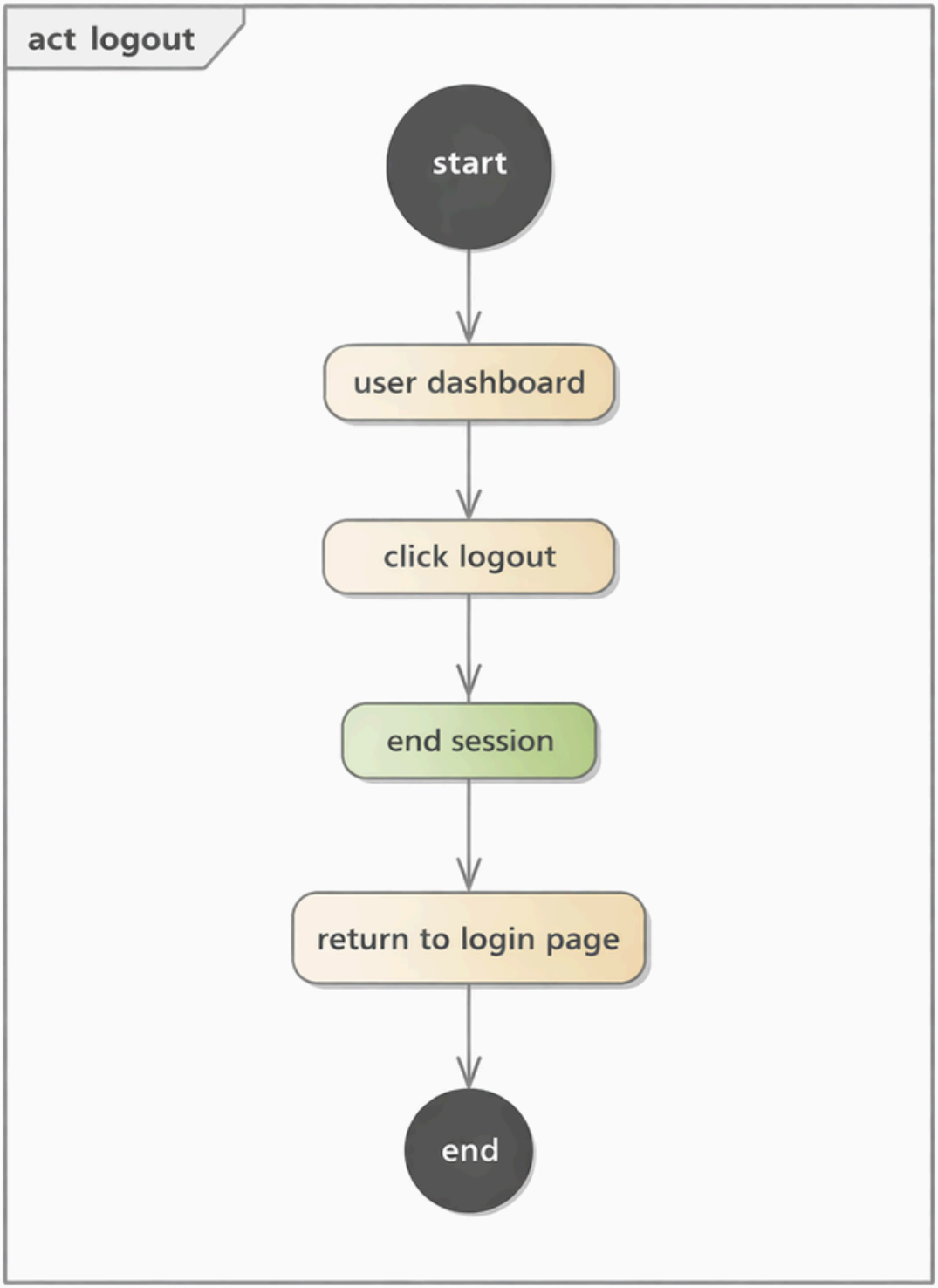
### 1 LOGIN

FIGURE 2.14



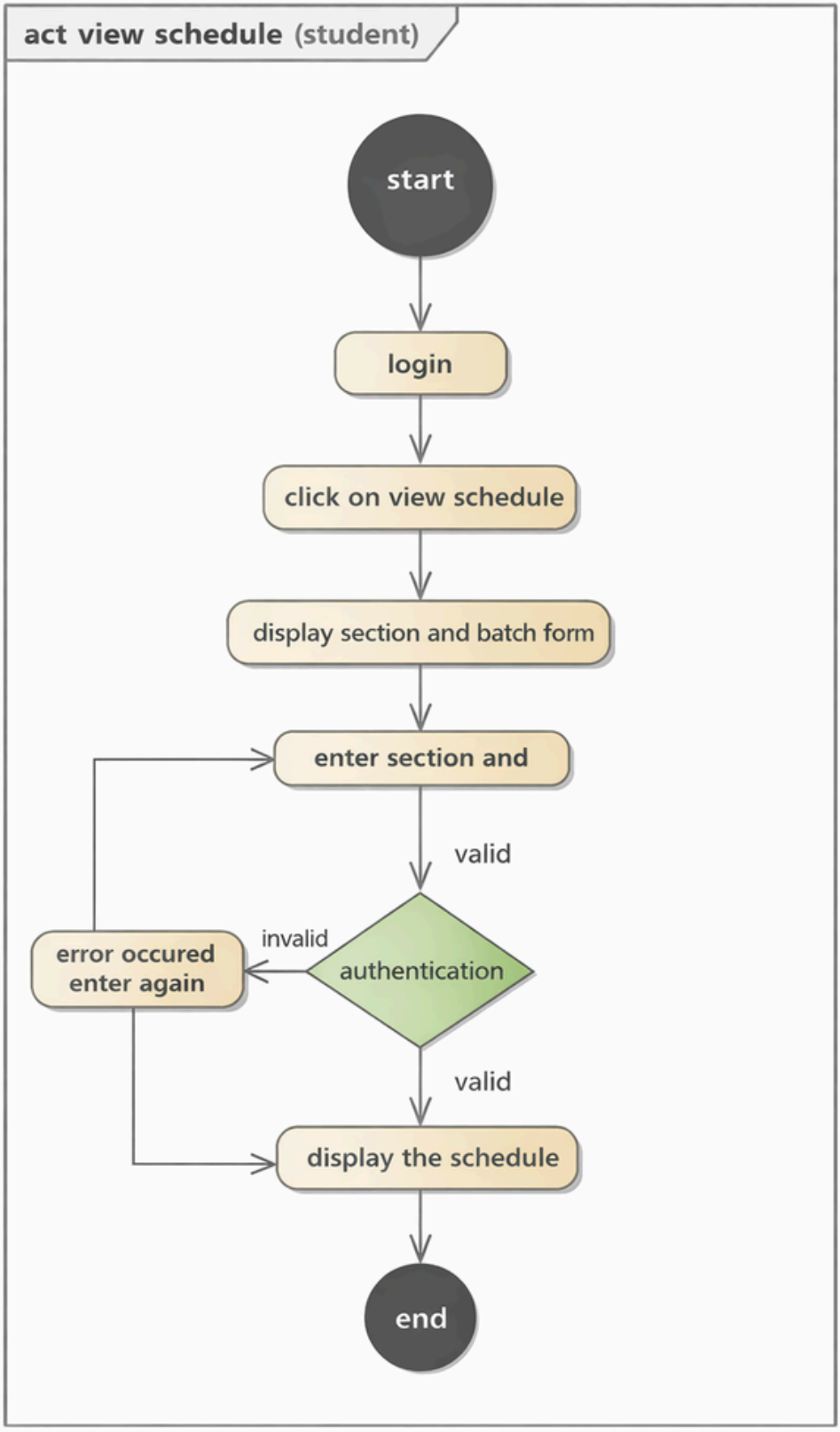
## 2 LOGOUT

FIGURE 2.15



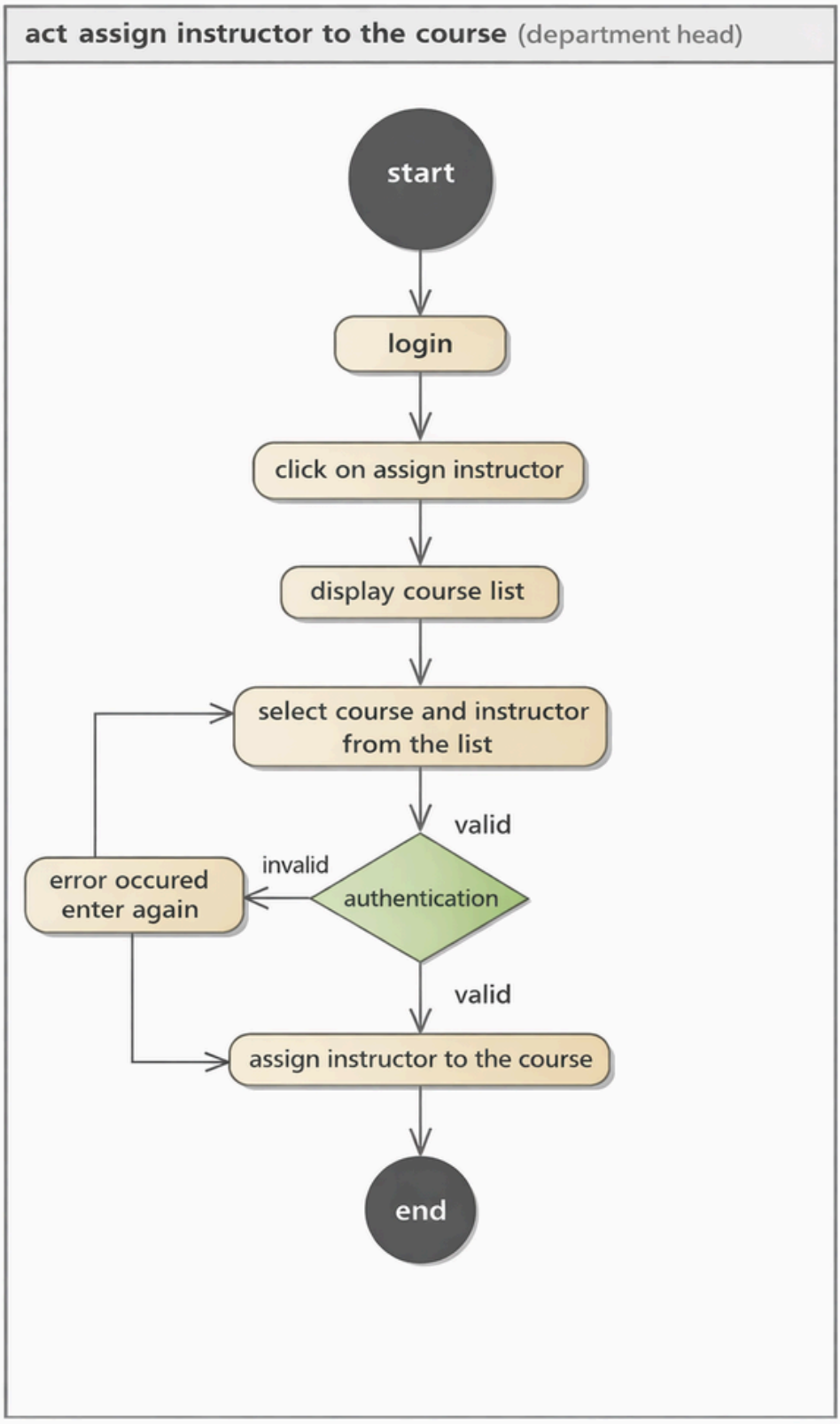
## 3 VIEW SCHEDULE

FIGURE 2.16



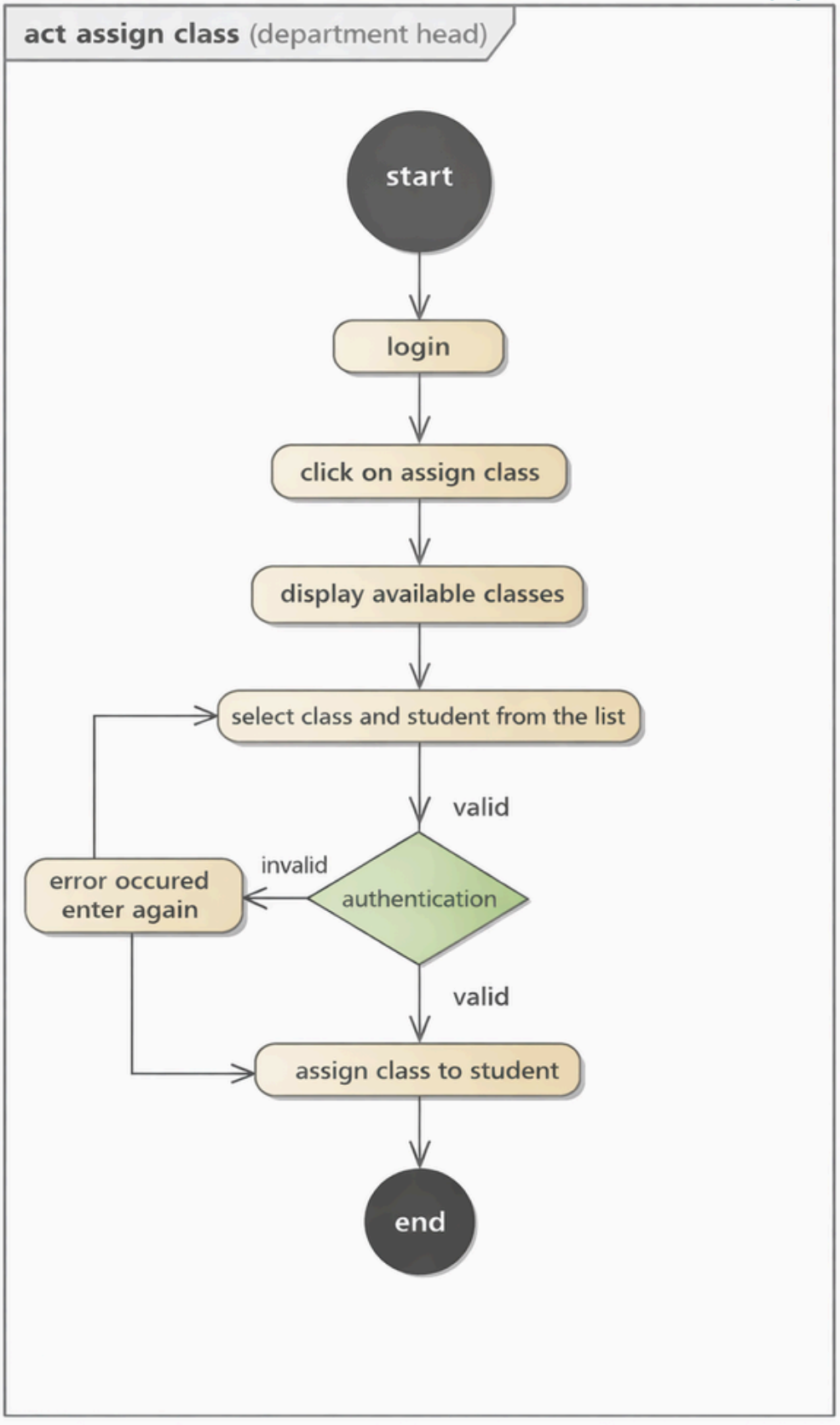
# 4 ASSIGN INSTRUCTOR TO THE COURSE

FIGURE 2.17



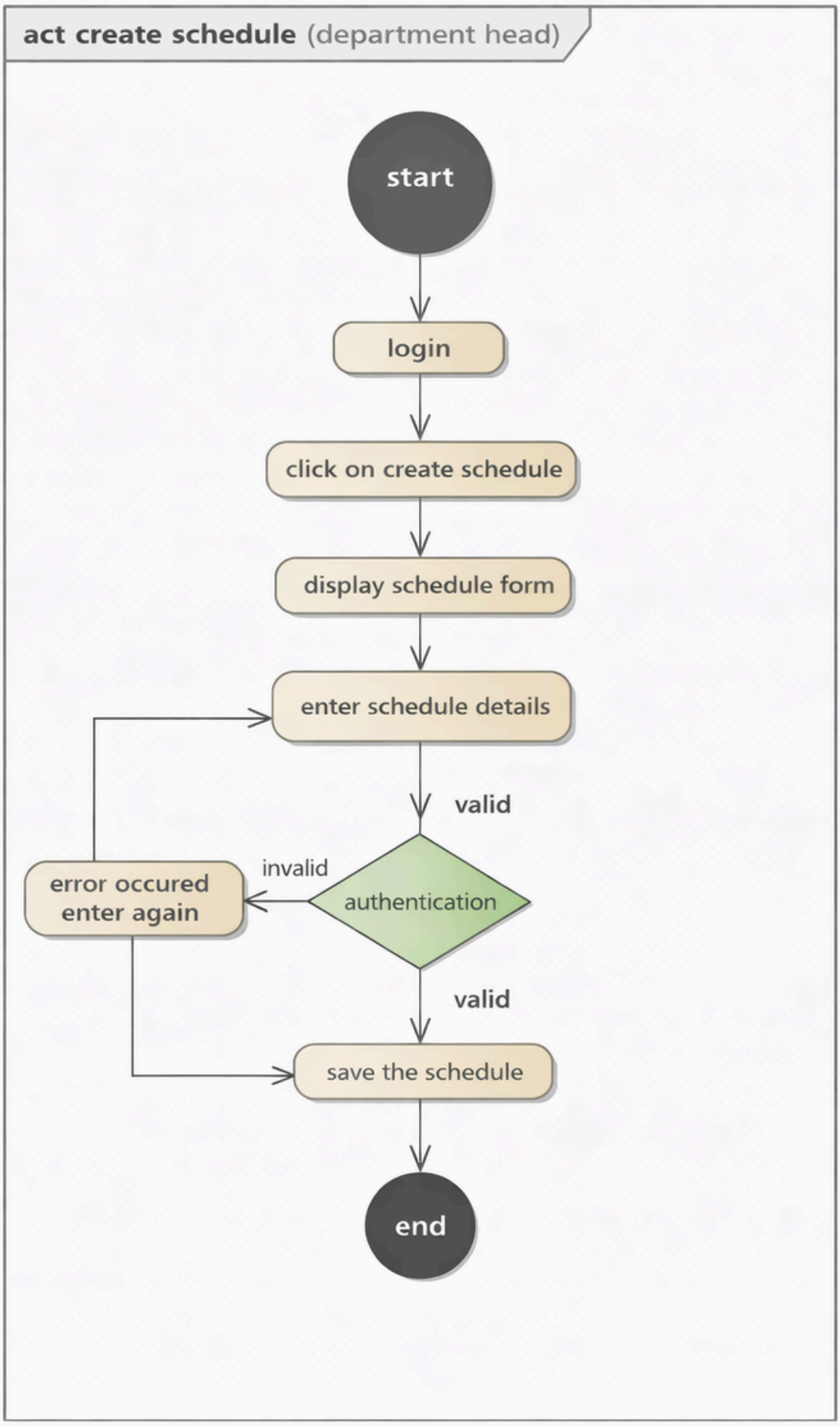
## 5 ASSIGN CLASS

FIGURE 2.18



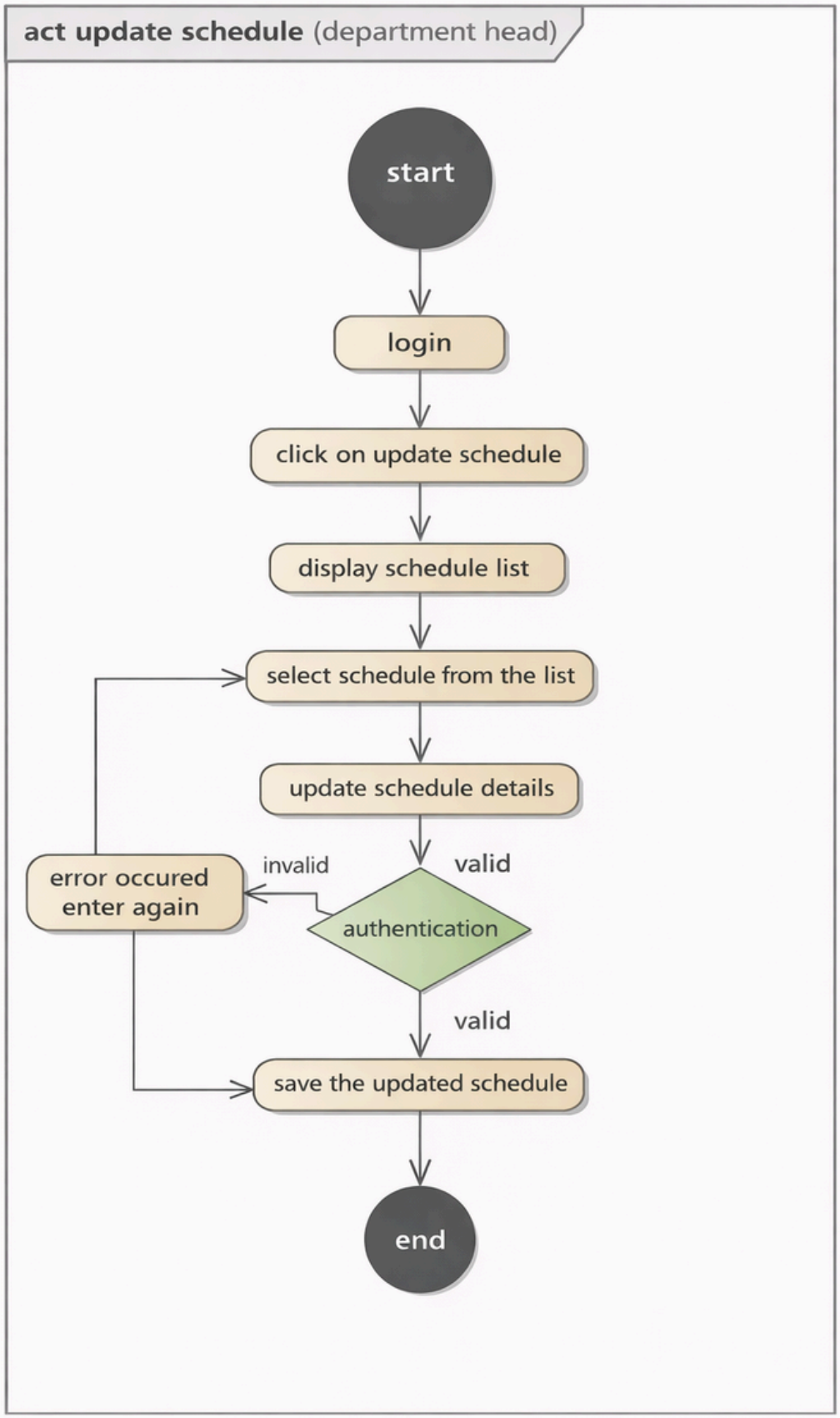
## 6 CREATE SCHEDULE

FIGURE 2.19



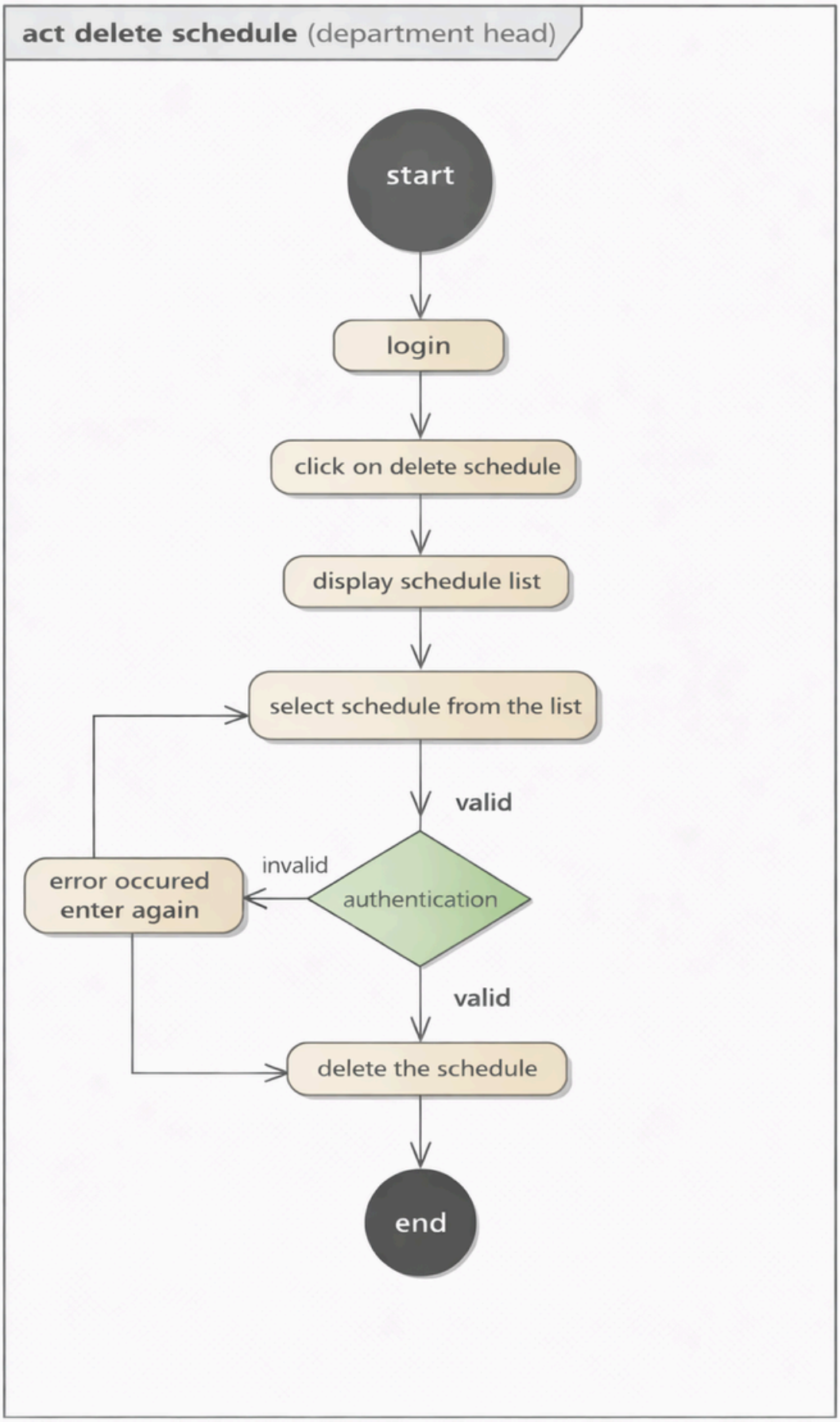
## 7 UPDATE SCHEDULE

FIGURE 2.20



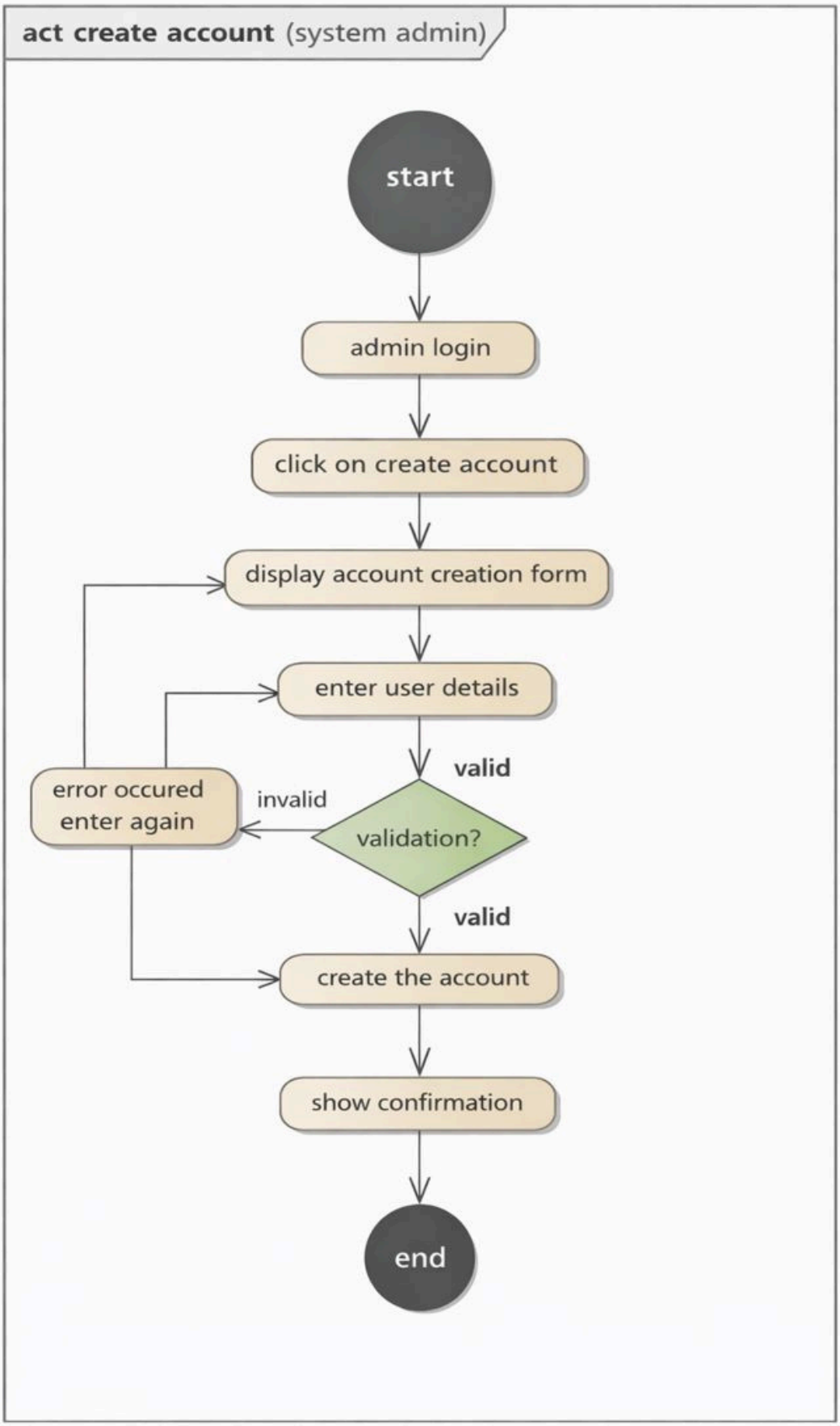
## 8 DELETE SCHEDULE

FIGURE 2.21



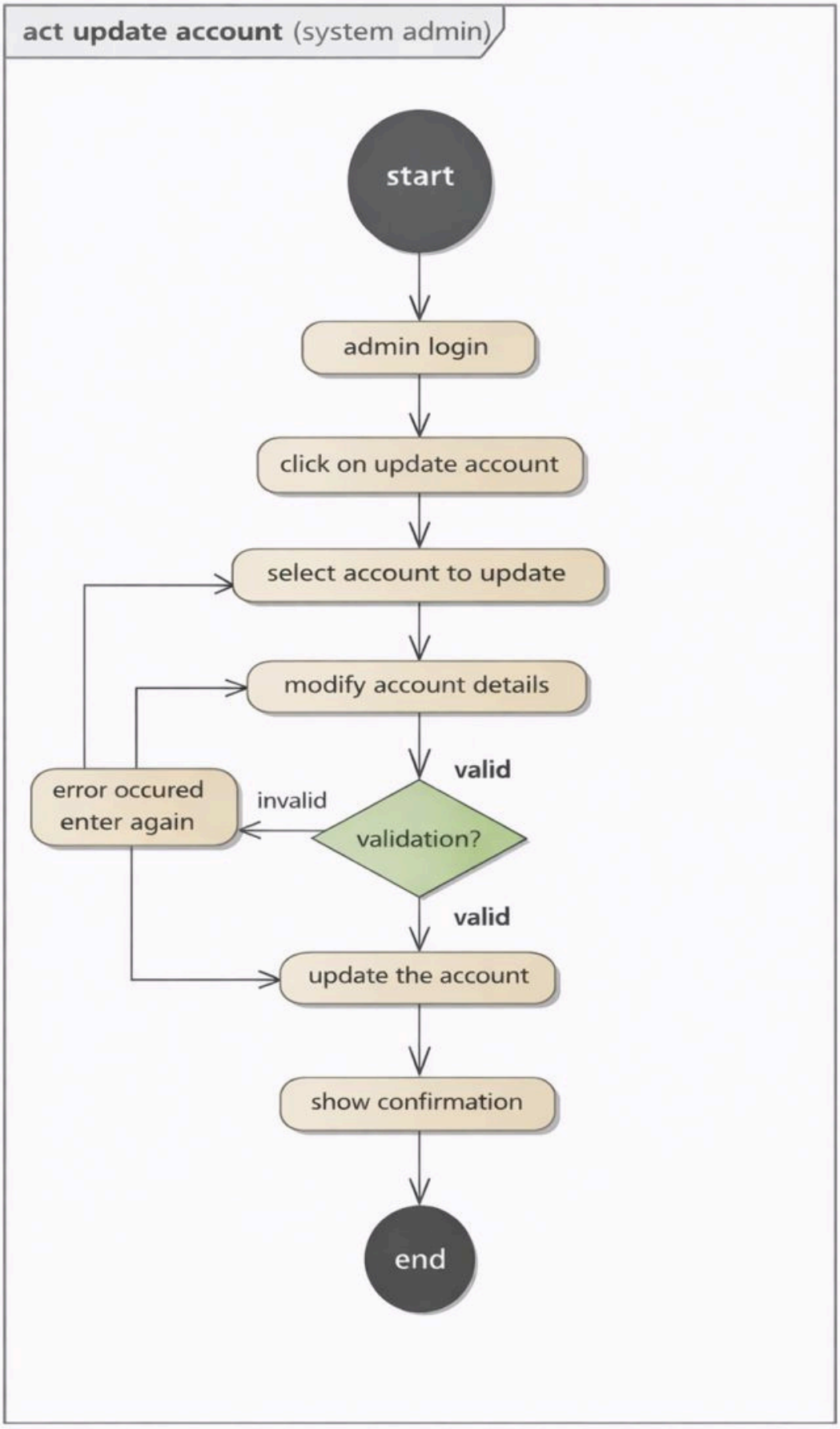
## 9 CREATE ACCOUNT

FIGURE 2.22



## 10 UPDATE ACCOUNT

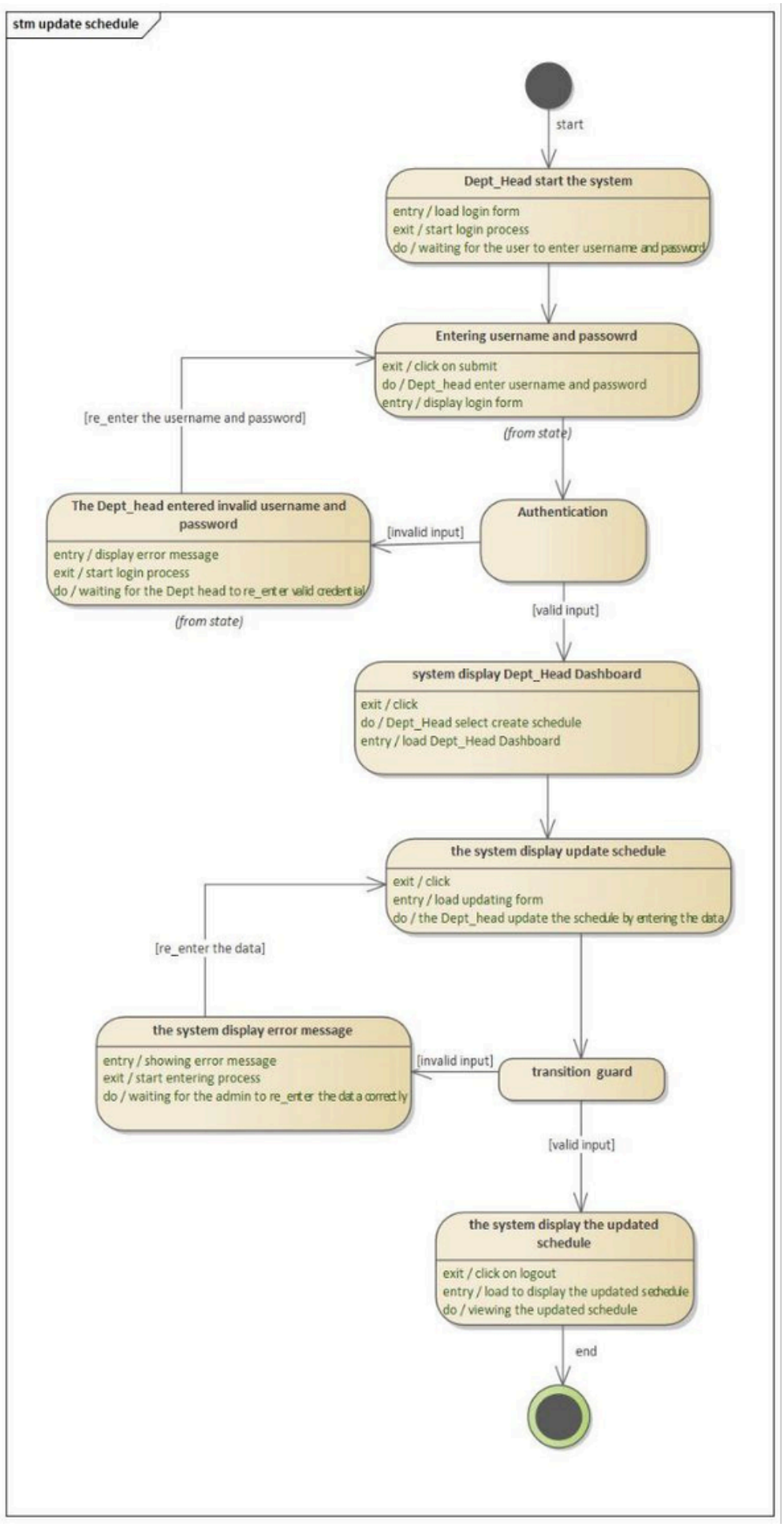
FIGURE 2.23




## 2.6.6.4. STATE CHART DIAGRAM

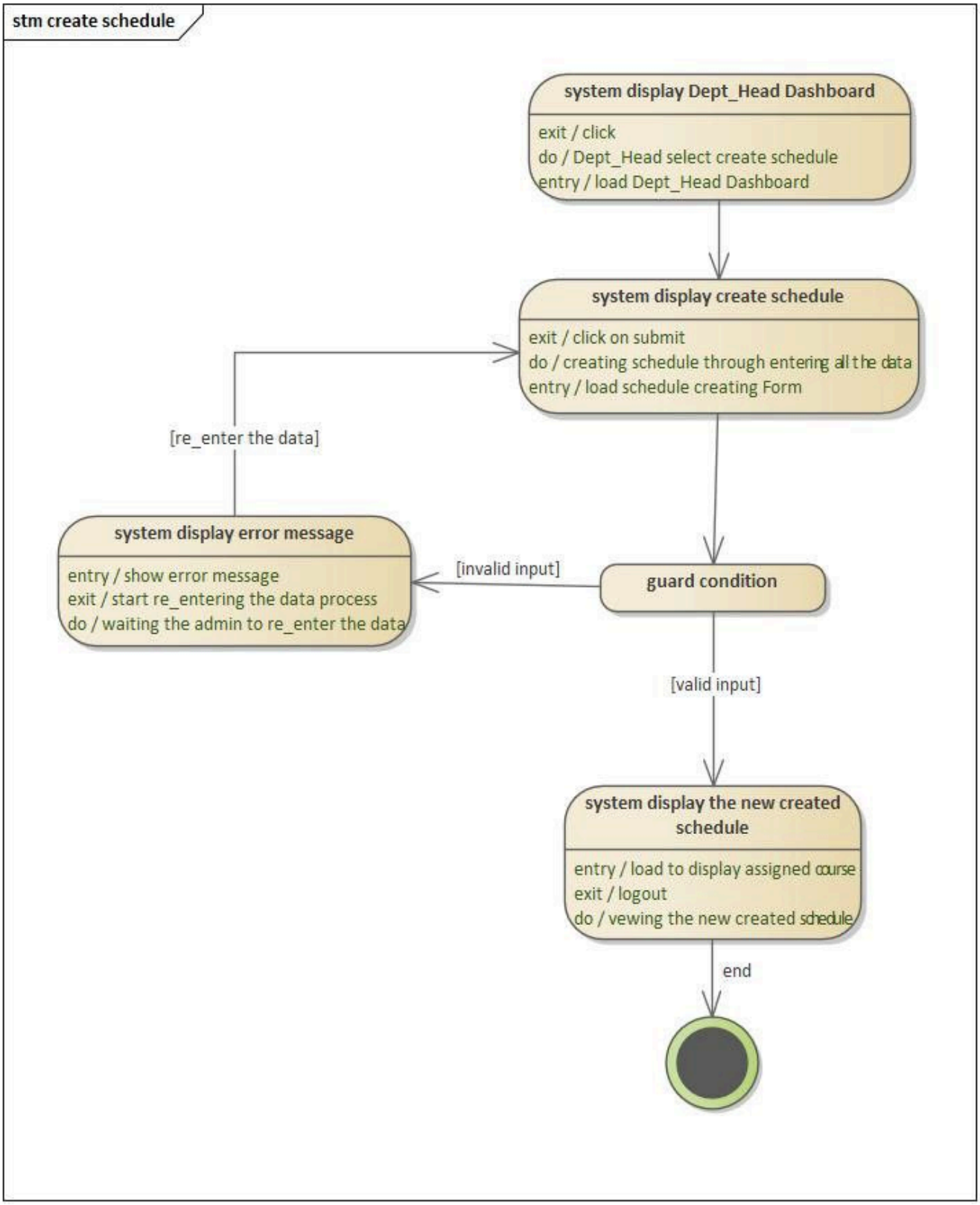
# UPDATE SCHEDULE

FIGURE 2.24



# CREATE SCHEDULE

**FIGURE 2.25**



# VIEW SCHEDULE

FIGURE 2.26

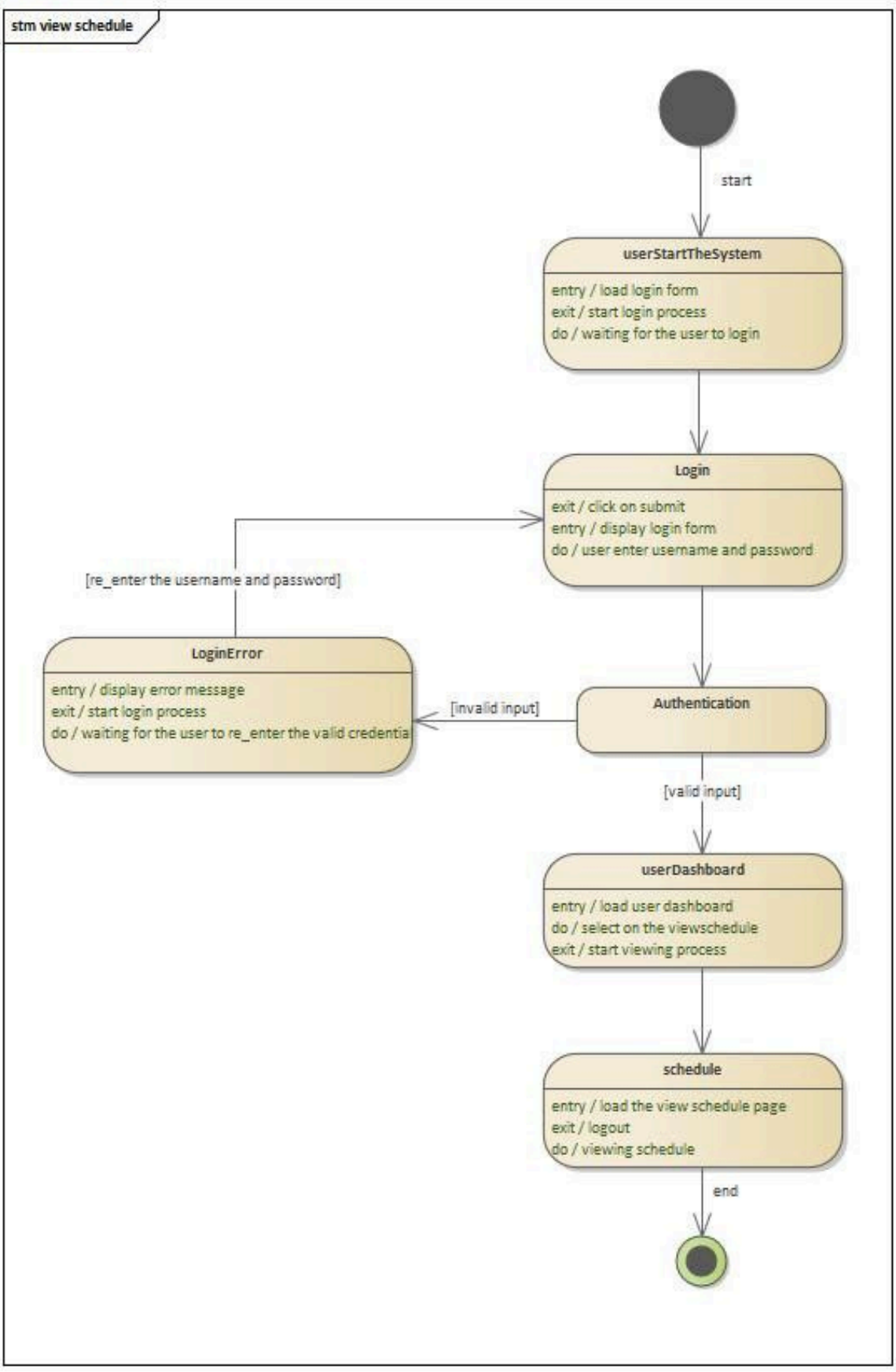
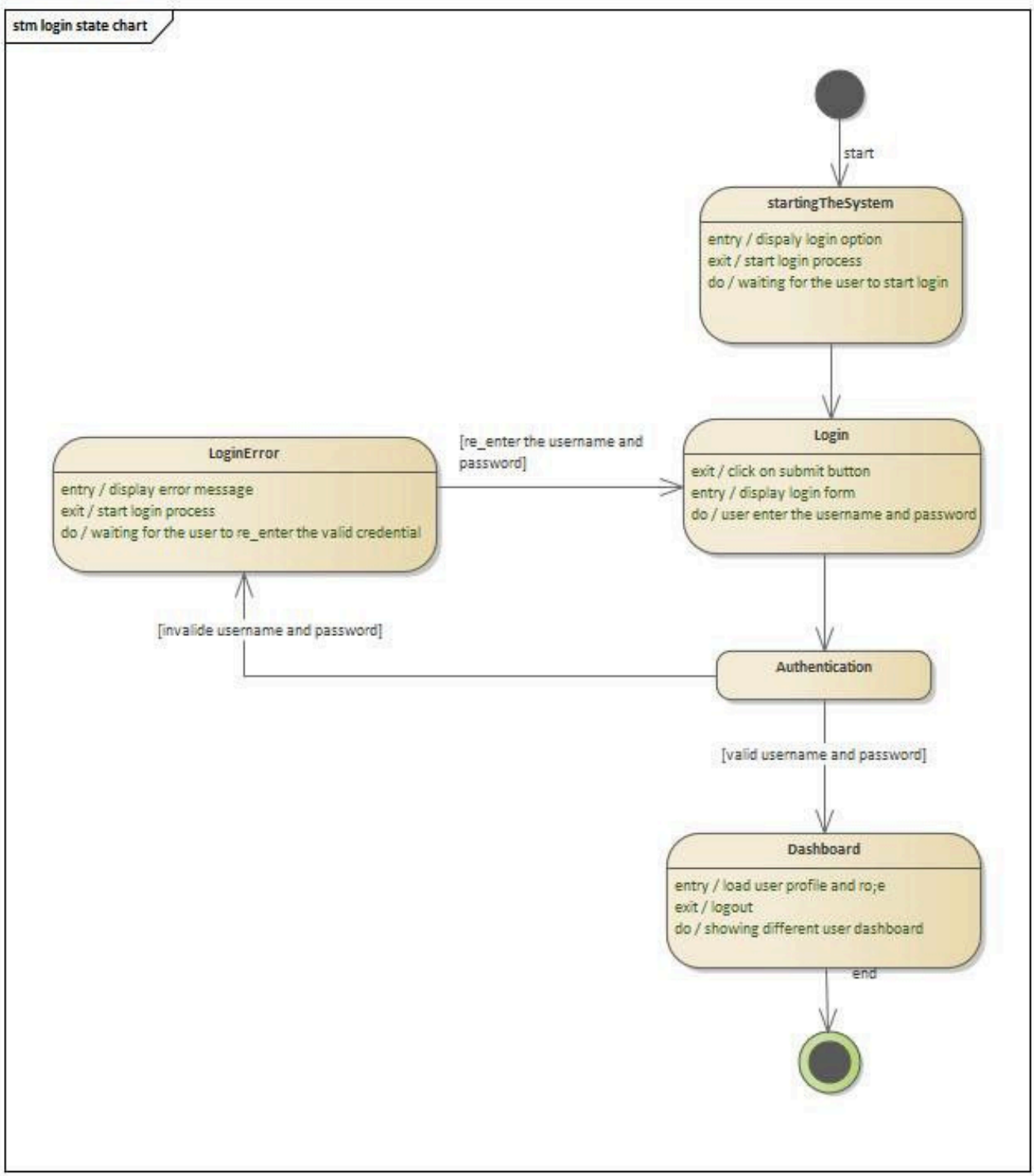
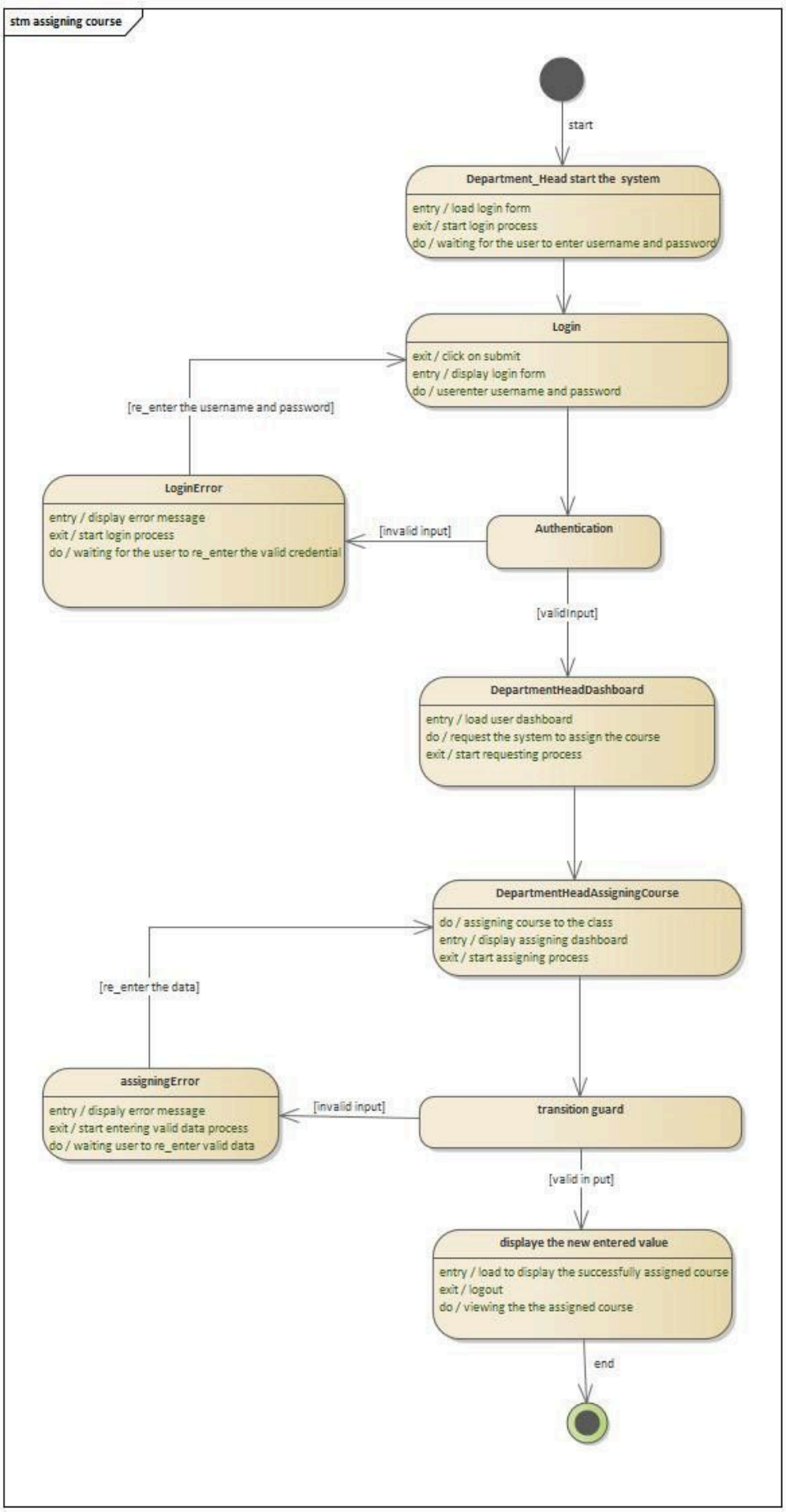


FIGURE 2.27


stm login state chart
start
startingTheSystem
entry / dispaly login option
exit / start login process
do / waiting for the user to start login
Login
exit / click on submit button
entry / display login form
do / user enter the username and password
[re_enter the username and password]
LoginError
entry / display error message
exit / start login process
do / waiting for the user to re_enter the valid credential
[invalide username and password]
Authentication
[valid username and password]
Dashboard
entry / load user profile and ro;e
exit / logout
do / showing different user dashboard
end

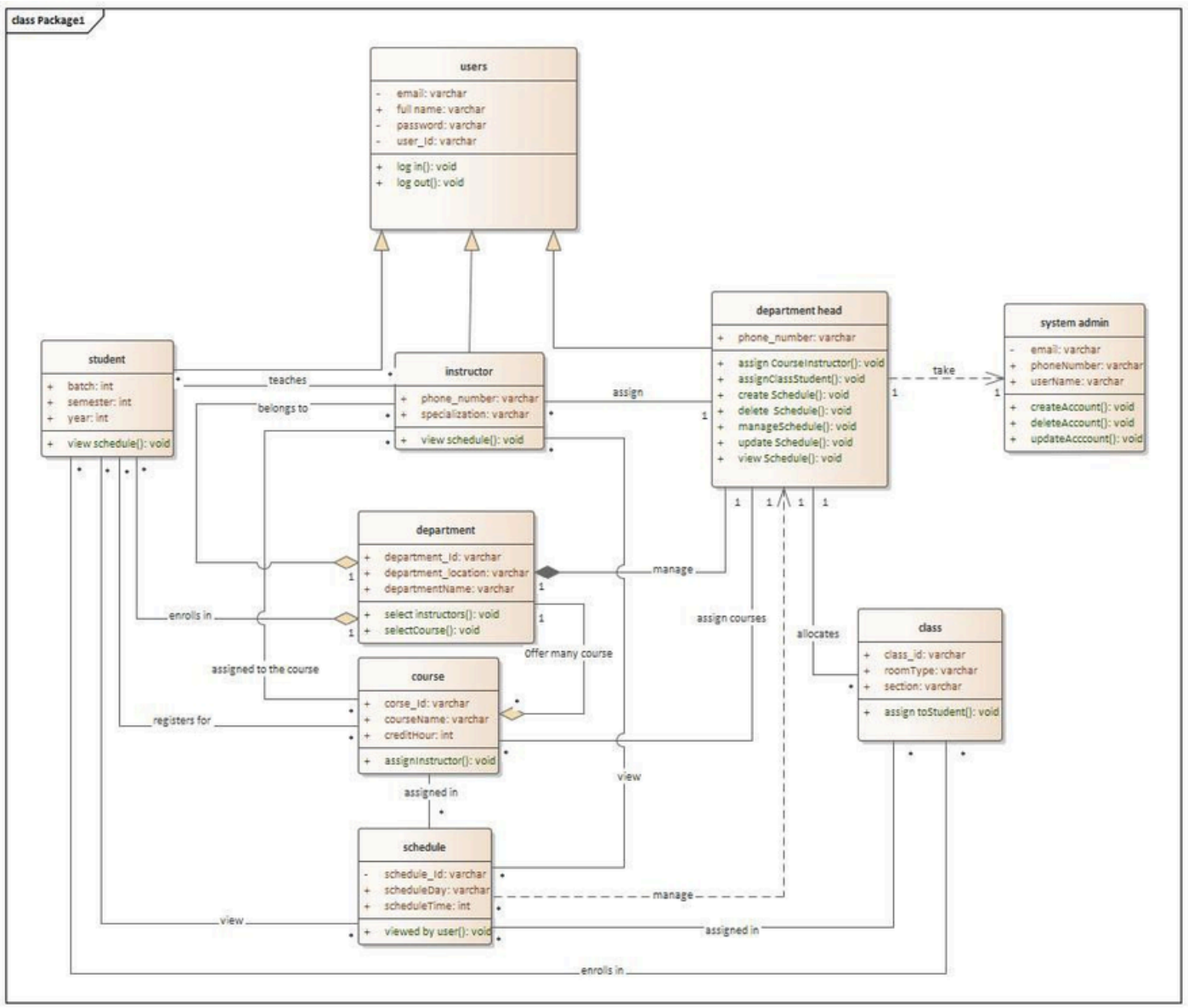
# ASSIGNING COURSE

FIGURE 2.28



## 2.6.6.5. CLASS DIAGRAM

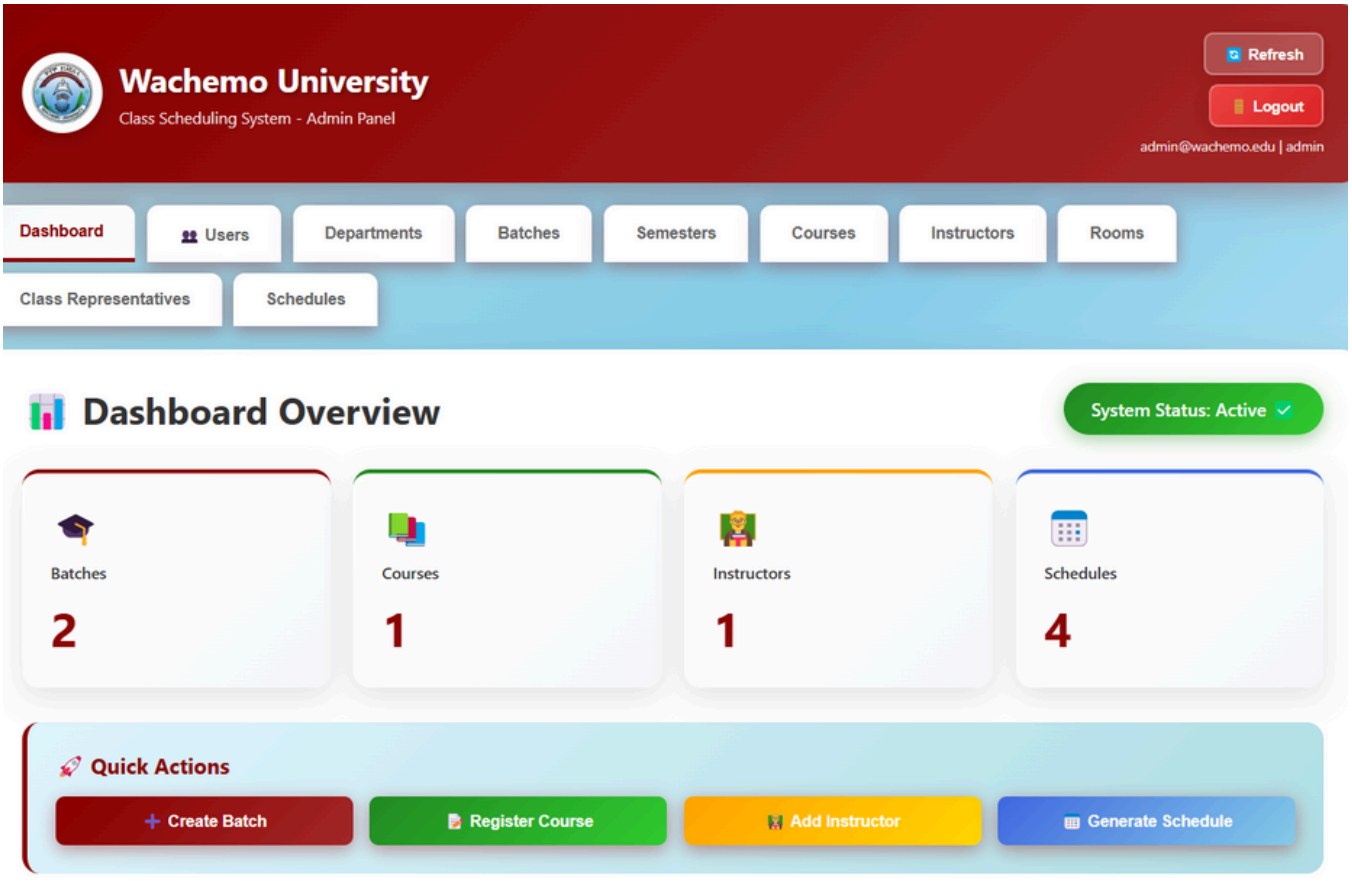
FIGURE 2.29



## 2.6.6.6. USER INTERFACE PROTOTYPING

FIGURE 2.30


FIGURE 2.31



# Chapter Three: Database Design

## 3.1 CONCEPTUAL DATABASE DESIGN

The conceptual database design describes the overall structure of the database using high-level concepts such as entities, attributes, and relationships. It focuses on what data is needed for the Online Class Scheduling System without considering how it will be physically stored.

The goal is to represent real-world objects in the Software Engineering department such as students, instructors, courses, classes, and schedules, and define how they are related.

### 3.1.1 ENTITIES IDENTIFICATION AND DESCRIPTION

From the ER diagram, the main entities identified are:

1. **Users** – Stores general login and identity information for all system users.
2. **Student** – Represents students who view schedules and enroll in courses.
3. **Instructor** – Represents instructors who teach courses and view schedules.
4. **Course** – Represents courses offered by the department.
5. **Class** – Represents physical classes with room and section details.
6. **Schedule** – Represents time and day information for classes.
7. **Department** – Represents the Software Engineering department.
8. **Department_Head** – Represents the head who manages schedules, classes, and instructors.

### 3.1.2 ATTRIBUTES IDENTIFICATION AND DESCRIPTION

**Users**

- user_id – Unique identifier
- fname – First name
- lname – Last name
- email – User email
- password – Login password

**Student**

- student_id – Unique student ID
- year – Academic year
- semester – Current semester
- user_id (FK) – Links to Users

**Instructor**

- instructor_id – Unique instructor ID
- instructor_specialization – Area of expertise
- phone_number – Contact number
- user_id (FK) – Links to Users

**Course**

- course_id – Unique course ID
- course_name – Name of the course
- course_hours – Credit/contact hours

**Class**

- class_id – Unique class ID
- section – Section name
- room_type – Type of room (lab/lecture)

**Schedule**

- schedule_id – Unique schedule ID
- scheduleDay – Day of class
- scheduleTime – Time of class

**Department**

- department_id – Unique department ID
- department_name – Name of department
- department_location – Location

**Department_Head**

- dephead_id – Unique head ID
- phone_number – Contact number
- user_id (FK) – Links to Users

### 3.1.3 RELATIONSHIP IDENTIFICATION AND DESCRIPTION

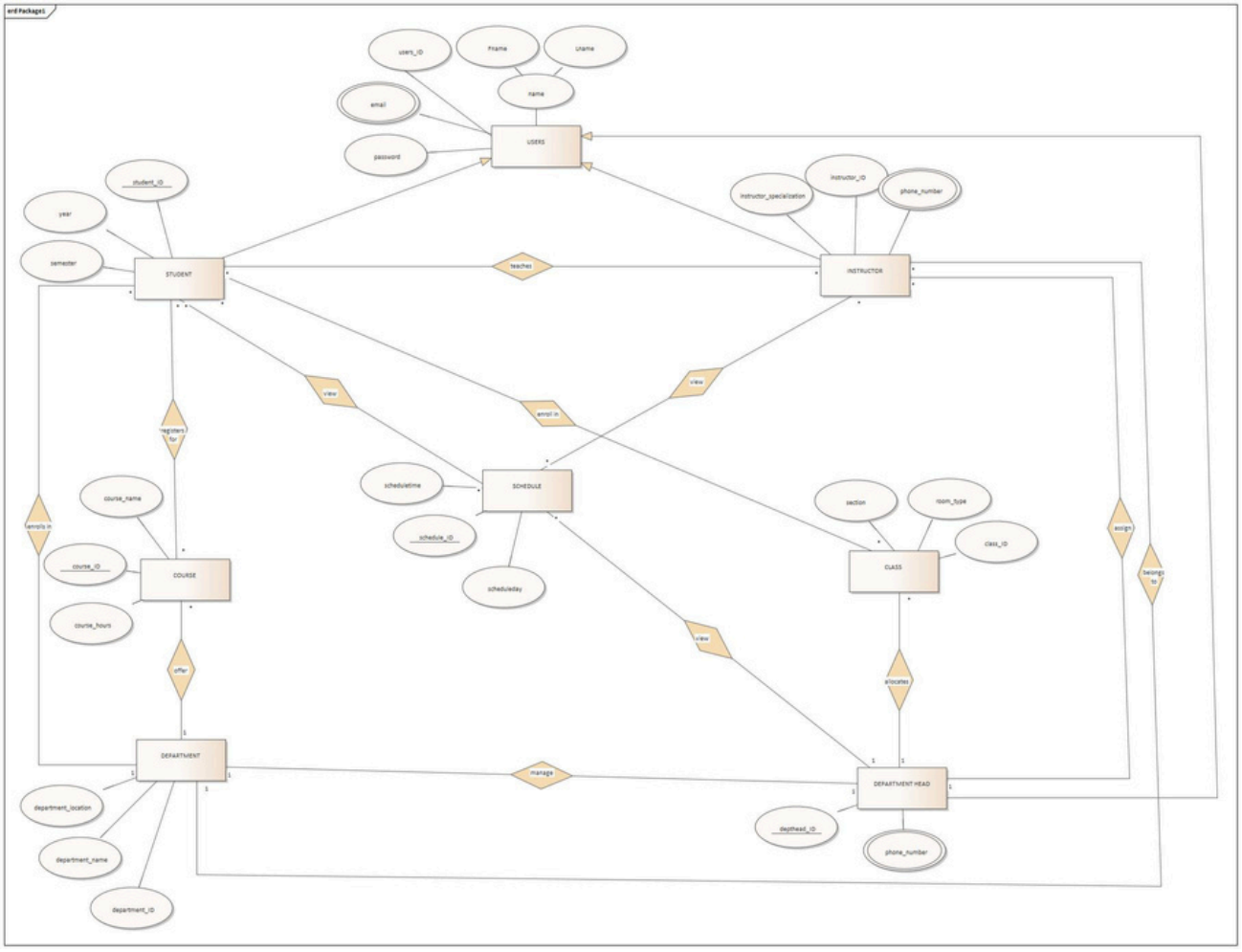
Key relationships in the system include:

- **Users – Student / Instructor / Department_Head**: One user can be a student, instructor, or department head (generalization).
- **Student enrolls in Course**: Many-to-many.
- **Instructor teaches Course**: Many-to-many.
- **Course assigned in Schedule**: One course can have many schedules.
- **Class allocated to Department_Head**: Managed by department head.
- **Department offers Course**: One-to-many.
- **Department managed by Department_Head**: One-to-one.
- **Student and Instructor view Schedule**: Many-to-many.

### 3.1.4 ER DIAGRAM

The ER diagram represents the entities, their attributes, and relationships as described above and serves as the conceptual model for the database design.

**FIGURE 3.1**




## 3.2 LOGICAL DATABASE DESIGN OF THE NEW SYSTEM

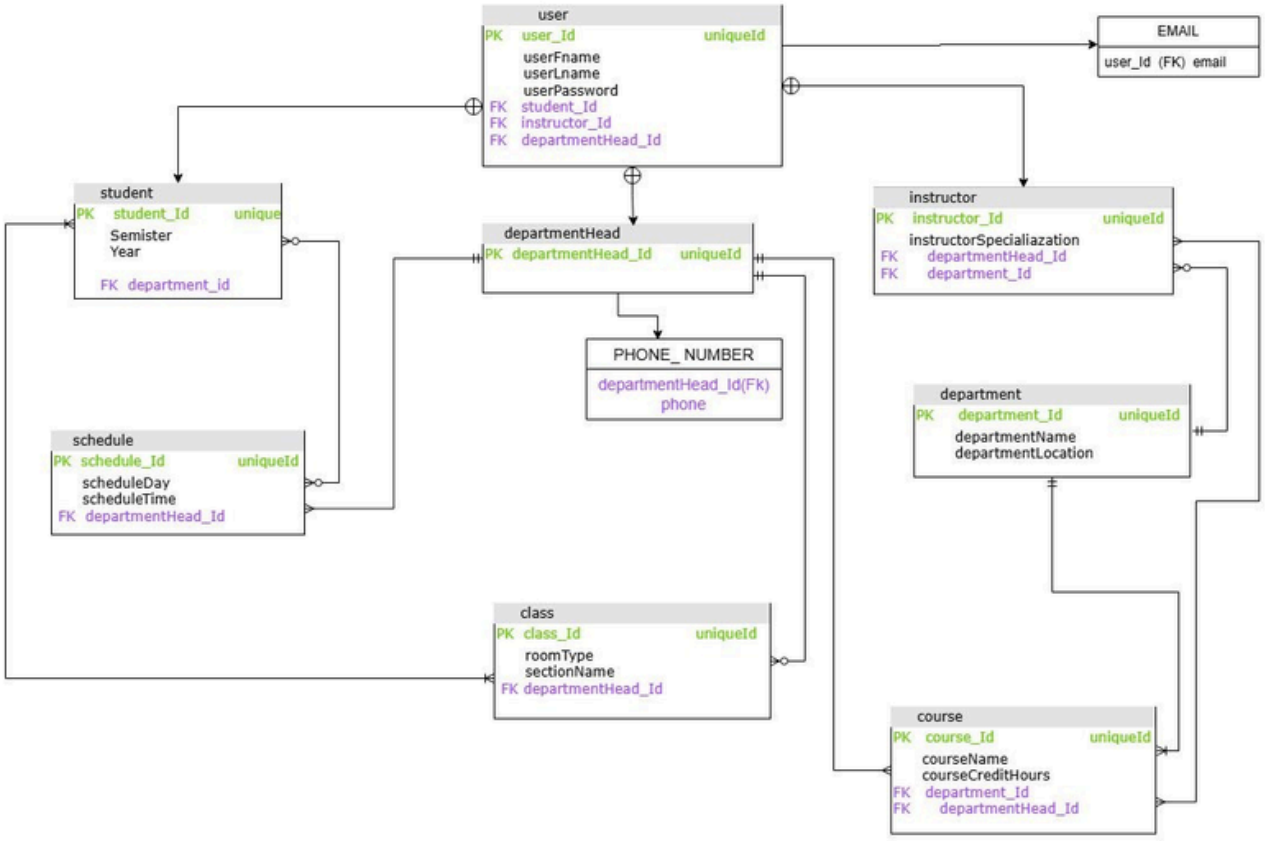
This phase converts the ER model into relational tables.

### 3.2.1 ER TO TABLE MAPPING

Each entity becomes a table:

- **USERS**(user_id, fname, lname, email, password)
- **STUDENT**(student_id, year, semester, user_id)
- **INSTRUCTOR**(instructor_id, instructor_specialization, phone_number, user_id)
- **DEPARTMENT**(department_id, department_name, department_location)
- **DEPARTMENT_HEAD**(dephead_id, phone_number, user_id, department_id)
- **COURSE**(course_id, course_name, course_hours, department_id)
- **Schedule** (schedule _Id,schedule day,schedule time)

**FIGURE 3.2**



### 3.2.2 VALIDATE MODEL USING NORMALIZATION

Normalization is applied to remove redundancy and ensure data integrity.

### 3.2.2.1 FIRST NORMAL FORM (1NF)

First Normal Form(1NF)

- USER TABLE

| User_id | Fullname | Password | Email | Student_Id(FK) | Instructor_id(FK) | Departmenthead_id(FK) |
|---|---|---|---|---|---|---|
| | | | | | | |

- ✓ It have primary key.
- ✓ It have composite and multivalue attributes. so we need to create other table to represent the multivalue attributes.

| User_Id | Fname | Lname | Password | Student_Id(FK) | Instructor_Id(FK) | DepartmentHead_Id(FK) |
|---|---|---|---|---|---|---|
| | | | | | | |

USER EMAIL

| Email_Id | Email | User_Id(FK) |
|---|---|---|
| | | |

- ✓ So now it's in 1 normal form.
- STUDENT TABLE

| Student_Id | Year | Semester | Department_Id(FK) |
|---|---|---|---|
| | | | |

- ✓ It have primary key.
- ✓ It have atomic (single).
- ✓ There is no multivalule.so it's in 1 normal form

- INSTRUCTOR TABLE

| Instructor_Id | Specailization | Phone_number | Department_Id(FK) | departmentHead_Id(FK) |
|---|---|---|---|---|
| | | | | |

| Instructor_Id | Specailization | Department_Id(FK) | departmentHead_Id(FK) |
|---|---|---|---|
| | | | |

- ✓ It have primary key.
- ✓ It have multivalue. So we need to create other table to represent it.

INSTRUCTOR_ PHONENUMBER

| Phone_Id | Phone_number | Instructor_Id(FK) |
|---|---|---|
| | | |

- ✓ Now it's in 1 normal form.

- DEPARTMENTHEAD TABLE

| DepartmentHead_Id | Phone _number | Department_Id(FK) |
|---|---|---|
| | | |

| DepartmentHead_Id | Department_Id(FK) |
|---|---|

- ✓ It have primary key.
- ✓ It have multivalue. So we need to create other table to represent it

DEPARTMENTHEAD_PHONENUMBER

| Phone_Id | Phone_number | DepartmentHead_Id(FK) |
|---|---|---|

- ✓ Now it's in 1 normal form.

➢ COURSE TABLE

| Course_Id | Coursename | Credithours | Department_Id(FK) | DepartmentHead_Id(FK) |
|---|---|---|---|---|

- ✓ It have primary key.
- ✓ It have atomic value.
- ✓ There is no multivalue .so it is in 1normal form

➢ CLASS TABLE

| Class_Id | Roomtype | Section | DepartmentHead(FK) |
|---|---|---|---|

- ✓ It have primary key.
- ✓ It have atomic value.
- ✓ There is no multivalue .so it is in 1 normal form

➢ SCHEDULE TABLE

| Schedule_Id | Scheduleday | Scheduletime | DepartmentHead_Id(FK) |
|---|---|---|---|

- ✓ It have primary key.
- ✓ It have atomic value.
- ✓ There is no multivalue .so it is in 1 normal form

➢ DEPARTMENT TABLE

| Department_Id | Departmentname | departmentlocation |
|---|---|---|

- ✓ It have primary key.
- ✓ It have atomic value.
- ✓ There is no multivalue .so it is in 1 normal form

### 3.2.2.2 SECOND NORMAL FORM (2NF)

Second normal form(2NF)

- USER TABLE

USER

| User_ID | Fname | Lname | Password | Student_Id(FK) | Instructor_Id(FK) | DepartmentHead_Id(FK) |
|---|---|---|---|---|---|---|

| Email_ Id | Email | User_id |
|---|---|---|

- ✓ It is already in 1NF.
- ✓ User_id is primary key.
- ✓ All attributes depend fully on the primary key.
- ✓ There is no partial dependency. so user table is in 2 normal form

- STUDENT TABLE

| Student_Id | semester | Year | Department_Id(FK) |
|---|---|---|---|

- ✓ It is in 1 NF
- ✓ Student_id is primary key.
- ✓ All attributes depend on student_id.
- ✓ There is no partial dependency.so student table is in 2NF

- INSTRUCTOR TABLE

| Instructor_Id | Specailization | Department_Id(FK) | departmentHead_Id(FK) |
|---|---|---|---|

INSTRUCTOR_PHONE

| Phone_Id | Phone_number | Instructor_Id(FK) |
|---|---|---|

- ✓ It is in 1 NF
- ✓ Instructor_id is primary key for instructor table.
- ✓ All attributes depend on instructor_id in instructor table.
- ✓ Phone_id primary key and instructor_id is foreign key for instructor phone table .
- ✓ Phone_number depends fully on phone_id.
- ✓ There is no partial dependency.so tables are in 2NF

❖ DEPARTMENTHEAD TABLE

| DepartmentHead_Id | Department_Id(FK) |
|---|---|

DEPARTMENTHEAD_PHONENUMBER

| Phone_Id | Phone_number | DepartmentHead_Id(FK) |
|---|---|---|

- ✓ It is in 1 NF.
- ✓ DepartmentHead_ id is a primary key.
- ✓ There is no partial dependency .
- ✓ So it is in 2NF.

❖ DEPARTMENT TABLE

| Department_Id | Departmentname | departmentlocation |
|---|---|---|

- ✓ It is in 1st normalization for it .
- ✓ It satisfy the rule of 2NF.
- ✓ So it is in 2NF.

➢ COURSE TABLE

| Course_Id | Coursename | Credithours | Department_Id(FK) | DepartmentHead_Id(FK) |
|---|---|---|---|---|

- ✓ The course table is in 1 normal form
- ✓ And it doesn't have partial dependency.
- ✓ So it is 2 normal form

➢ CLASS TABLE

| Class_Id | Roomtype | Section | DepartmentHead(FK) |
|---|---|---|---|

- ✓ Class_id is the primary key.
- ✓ All attributes depend fully on the key
- ✓ So it is in 2NF.

➢ SCHEDULE TABLE

| Schedule_Id | Scheduleday | Scheduletime | DepartmentHead_Id(FK) |
|---|---|---|---|

- ✓ The schedule table is in 1 normal form
- ✓ And it doesn't have partial dependency.
- ✓ So it is 2 normal form

### 3.2.2.3 THIRD NORMAL FORM (3NF)

Third normal form(3NF)

- USER TABLE

| User_Id | Fname | Lname | Password |
|---|---|---|---|

- ✓ It is in 2NF.
- ✓ There is no transitive depandecy.
- ✓ User table is in 3 NF

USER EMAIL

| Email_ Id | Email | User_id |
|---|---|---|

- ✓ It is in 2NF
- ✓ Email depend only on email_id
- ✓ No non key attribute depend on other nonkey attribute
- ✓ So the table is in 3NF

- STUDENT TABLE

There is transitive dependency

Student_id-departmenthead_Id

Department_id- Department_id

So we need to create new table to solve it

| Student_Id | User_Id(FK) | semester | Year | Department_Id(Fk) | Department_Id(FK) |
|---|---|---|---|---|---|

| Student_Id | User_Id(FK) | semester | Year | Department_Id(FK) |
|---|---|---|---|---|

| DepartmentHead_Id | Department_Id(FK) | Student_Id |
|---|---|---|

- ✓ It is in 2NF
- ✓ Non key attributes depend only on student_Id
- ✓ No non key attribute depend on other nonkey attribute
- ✓ So the table is in 3NF

- INSTRUCTOR TABLE

There is transitive dependency

| Instructor_Id | User_Id | Specailization | Department_Id(FK) |
|---|---|---|---|

| DepartmentHead_Id | Department_Id(FK) | Instructor_Id |
|---|---|---|

- ✓ It is in 2NF
- ✓ Non key attributes( specialization) depend only on Instructor_Id
- ✓ No non key attribute depend on other non key attribute
- ✓ So the table is in 3NF

➢ DEPARTMENT TABLE

| Department_Id | Departmentname | departmentlocation |
|---|---|---|

✓ it is in 2NF

✓ there is no transitive dependency. so it is in 3NF

➢ DEPARTMENTHEAD TABLE

| DepartmentHead_Id | Department_Id(FK) |
|---|---|

DEPARTMENTHEAD_PHONENUMBER

| Phone_Id | Phone_number | DepartmentHead_Id(FK) |
|---|---|---|

✓ it is in 2NF

✓ there is no transitive dependency. so it is in 3NF

➢ COURSE TABLE

| Course_Id | Coursename | Credithours | Department_Id(FK) | DepartmentHead_Id(FK) |
|---|---|---|---|---|

✓ It is 2NF but it have transitive dependency.
✓ So we need to create new table to solve this.

| Course_Id | Coursename | Credithours | Department_Id(FK) |
|---|---|---|---|

| DepartmentHead_Id | Department_Id(FK) | Course_Id |
|---|---|---|

✓ Now it is in 3NF.

➢ CLASS TABLE

| Class_Id | Roomtype | Section | DepartmentHead(FK) |
|---|---|---|---|

✓ it is in 2NF

✓ there is no transitive dependency. so it is in 3NF

### 3.2.2.4 OTHER NF

The design is sufficient up to 3NF for this system.

BCNF(Boyce-Codd Normal Form)

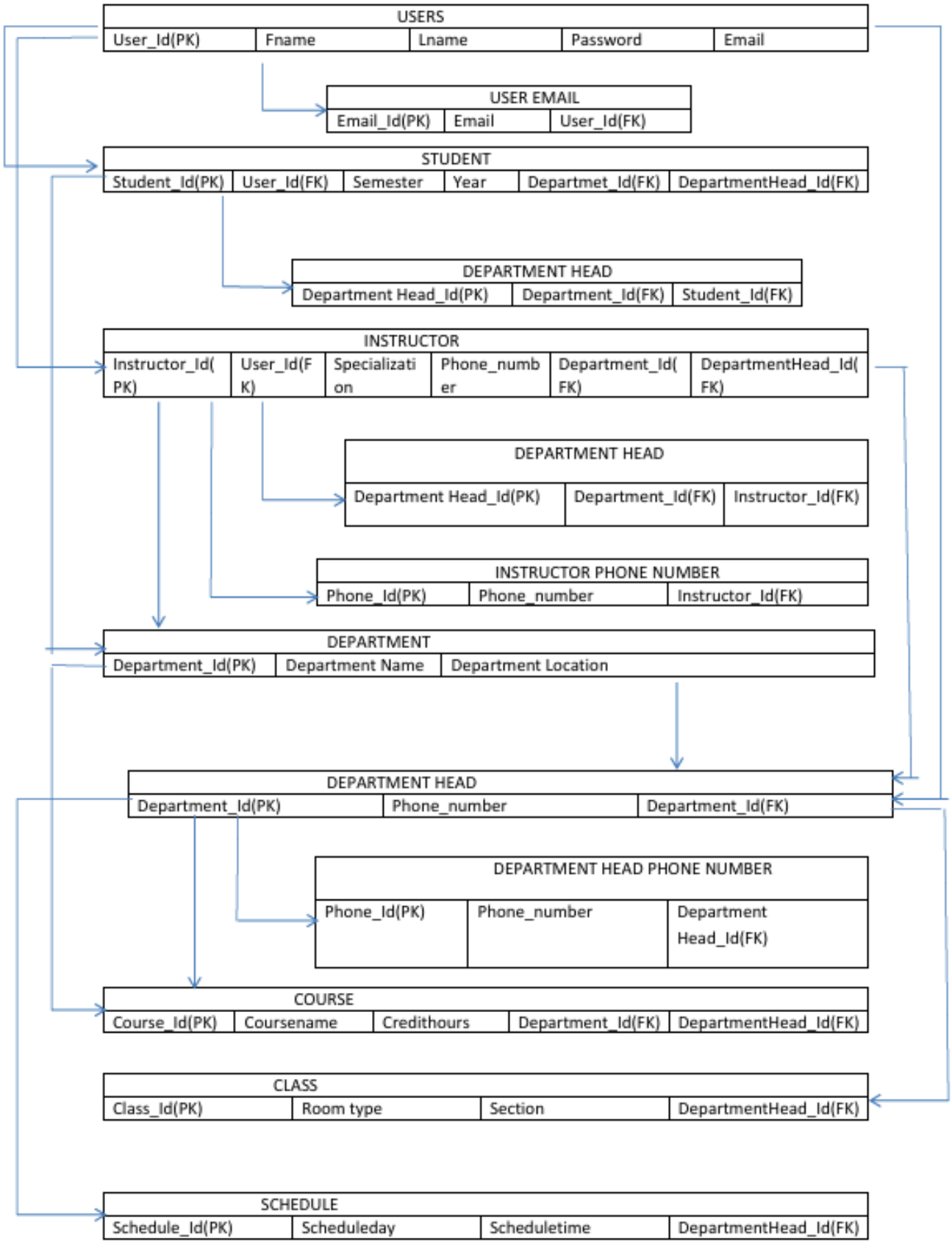
BCFN is an advance version of 3NF.it is stricter than 3NF.

A table in BCNF , if ever functional dependency X—Y, X IS a candidate key.

This means that the only primary key or candidate key determaine non key attribute.

- Each table have primary key
- All non key attributes depend only on the primary key
- So it satisfied BCNF

### 3.2.3 RELATIONAL SCHEMA WITH REFERENTIAL INTEGRITY



## 3.3 PHYSICAL DATABASE DESIGN OF THE NEW SYSTEM

This phase describes how the database will be implemented in MySQL.

### 3.3.1 PHYSICAL DESIGN STRATEGY

- MySQL will be used as DBMS.
- Tables will be created using InnoDB engine to support foreign keys.
- Indexes will be applied on primary and foreign keys for performance.
- Proper data types such as INT, VARCHAR will be used.

### 3.3.2 HARDWARE IMPLEMENTATION

The database will run on:

- A server or local machine with:
- Minimum 4 GB RAM
- Intel i3 or higher processor
- 20 GB free disk space
- Connected through local network or internet for system access.

# CHAPTER FOUR: SYSTEM DESIGN

## 4.1 INTRODUCTION

This chapter describes the design of the proposed Online Class Scheduling System for the Department of Software Engineering at Wachemo University. System design translates the requirements identified in the previous chapter into a technical blueprint that guides implementation. It explains how the system components are organized, how users interact with the system, and how data is processed and stored.

## 4.2 PURPOSE OF THE SYSTEM

The purpose of the system is to automate the online class scheduling process and replace the existing manual method. The system is designed to:

- Create and manage class schedules efficiently.
- Reduce conflicts and overlaps in schedules.
- Provide easy access to schedules for students and instructors.
- Support department heads in managing academic planning.
- Improve accuracy, consistency, and accessibility of scheduling data.

## 4.3 DESIGN GOALS

The main goals considered during system design are:

- Simplicity: Easy to use for students, instructors, and administrators.
- Reliability: Ensure correct and consistent scheduling data.
- Efficiency: Fast schedule creation and retrieval.
- Security: Protect user accounts and academic data.
- Scalability: Allow future expansion to other departments.
- Maintainability: Make the system easy to update and improve.

## 4.4 CURRENT SOFTWARE ARCHITECTURE

The current system follows a manual architecture, where:

- Class schedules are prepared using paper or basic office tools.
- Data is stored in files or notebooks.
- Communication is done verbally or through notice boards.

This approach:

- Has no centralized database.
- Is time-consuming and error-prone.
- Makes updating and sharing schedules difficult.
- Has no proper security or backup mechanism.

## 4.5 PROPOSED SOFTWARE ARCHITECTURE

The proposed system uses a **simple layered architecture** with clear separation of responsibilities. It includes:

- **Presentation Layer**: User interfaces for students, instructors, department heads, and admins.
- **Application Layer**: Handles business logic like assigning courses, creating schedules, and managing users.
- **Data Layer**: MySQL database to store users, courses, classes, schedules, and departments.

This structure improves maintainability and clarity of the system.

### 4.5.1 SUBSYSTEM DECOMPOSITION

The system is divided into the following subsystems:

**User Management Subsystem**
Handles login, logout, and user roles (student, instructor, department head, admin).

**Course Management Subsystem**
Manages courses offered by the department.

**Class and Room Management Subsystem**
Handles class sections and room allocation.

**Schedule Management Subsystem**
Creates, updates, and views schedules.

**Department Management Subsystem**
Manages department information and links users to departments.

### 4.5.2 COMPONENT DIAGRAM

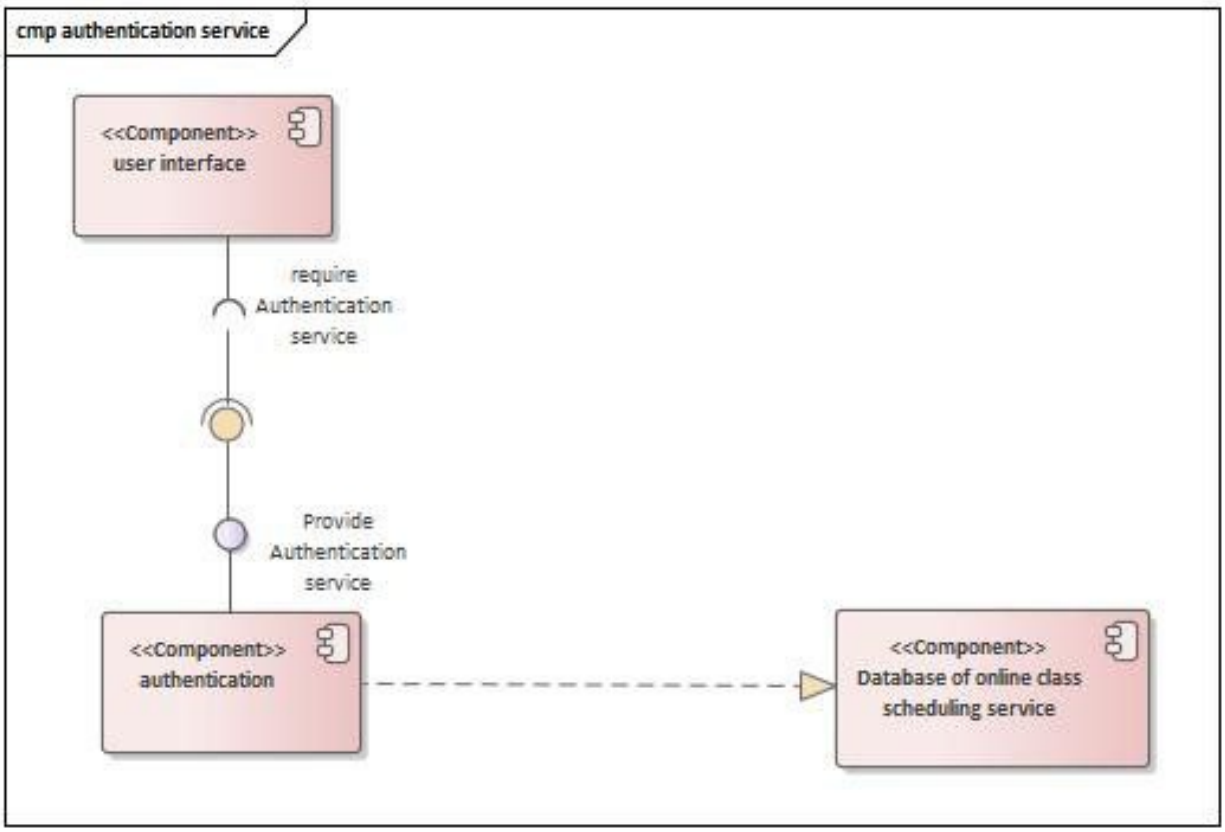
The main components of the system include:

- User Interface Component
- Authentication Component
- Schedule Controller
- Course Controller
- Class Controller
- Database Access Component

Each component interacts through well-defined interfaces to perform its tasks.

## AUTHENTICATION SERVICES

FIGURE 4.1




## COURSE ASSIGNING SERVICE

FIGURE 4.2

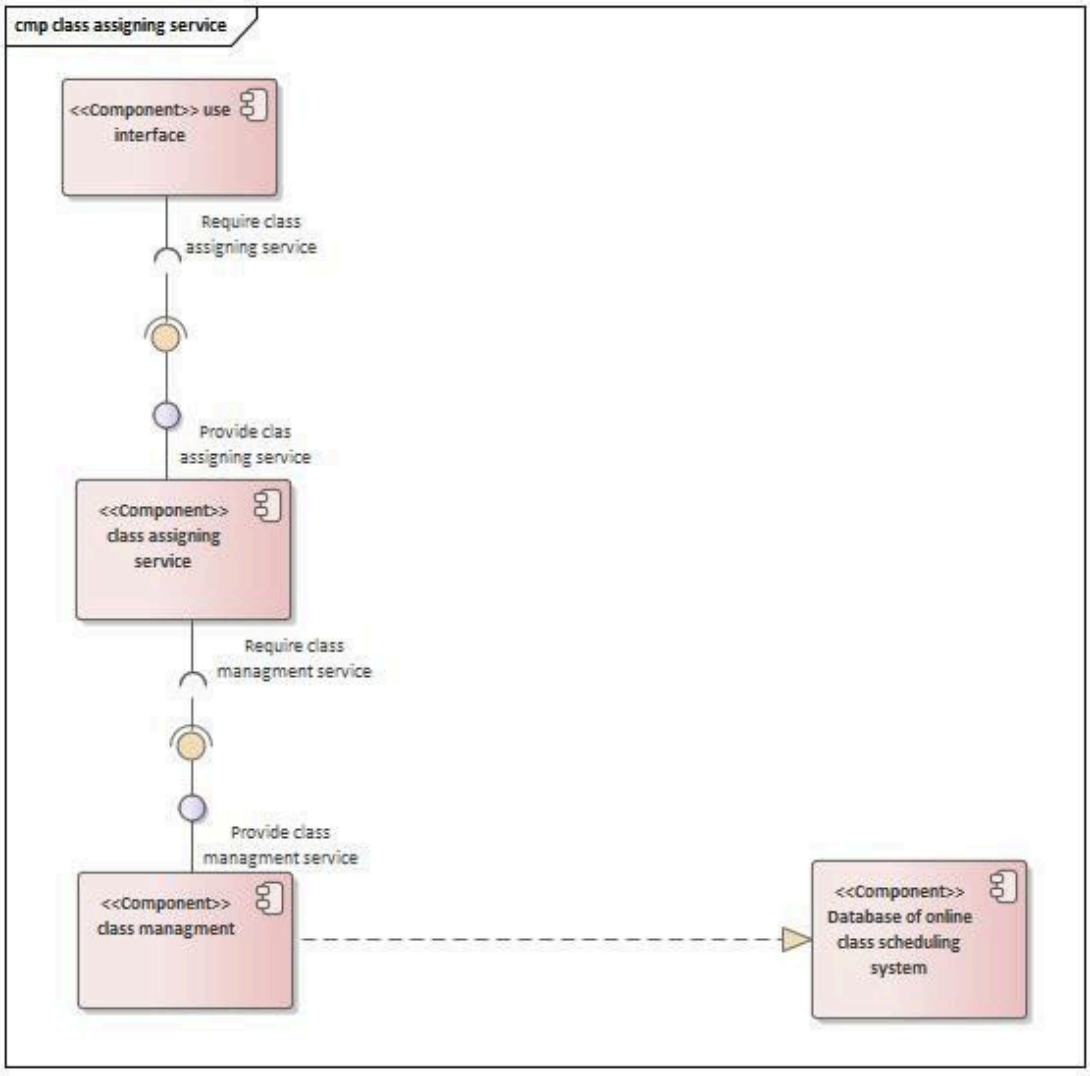
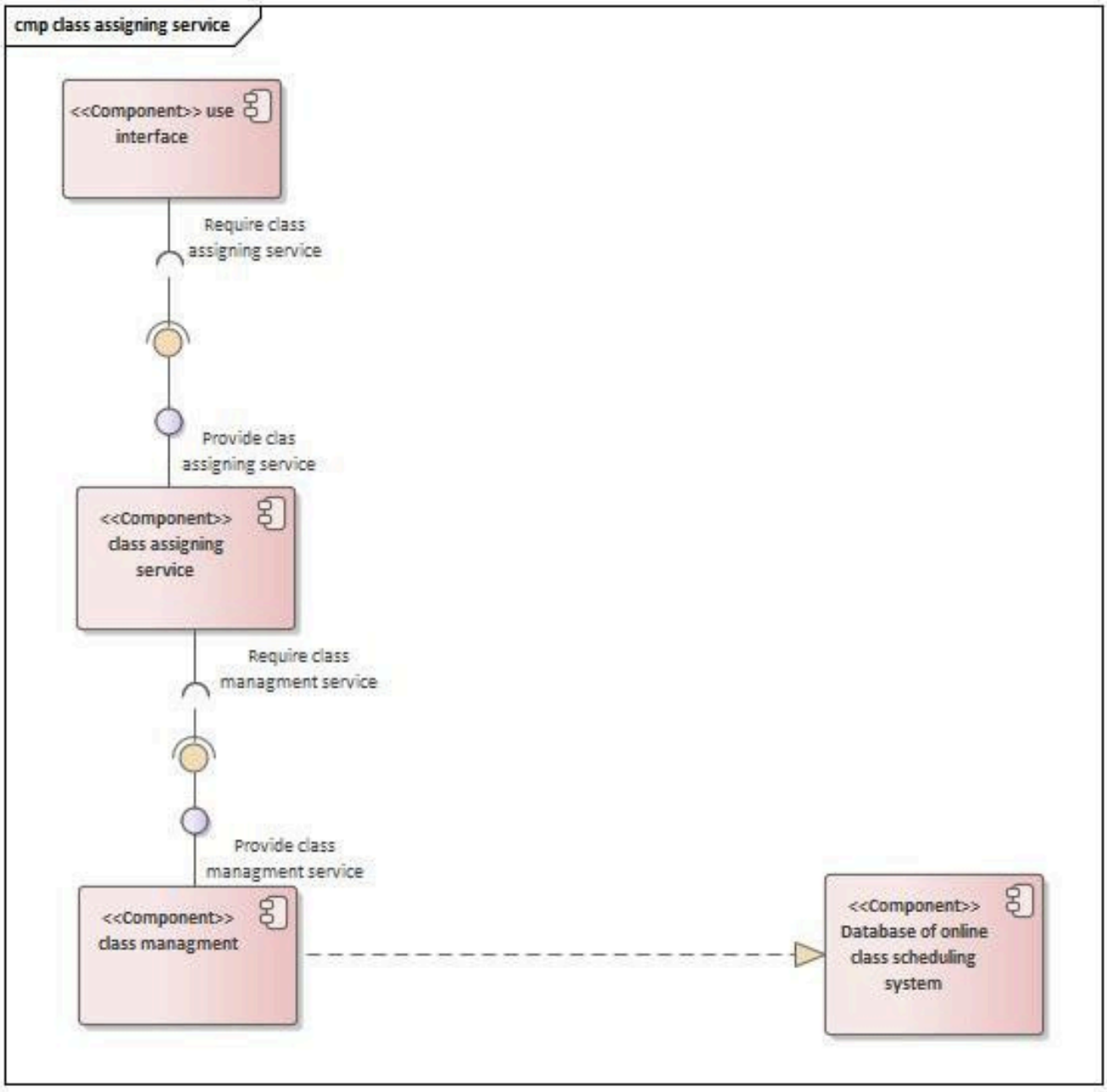
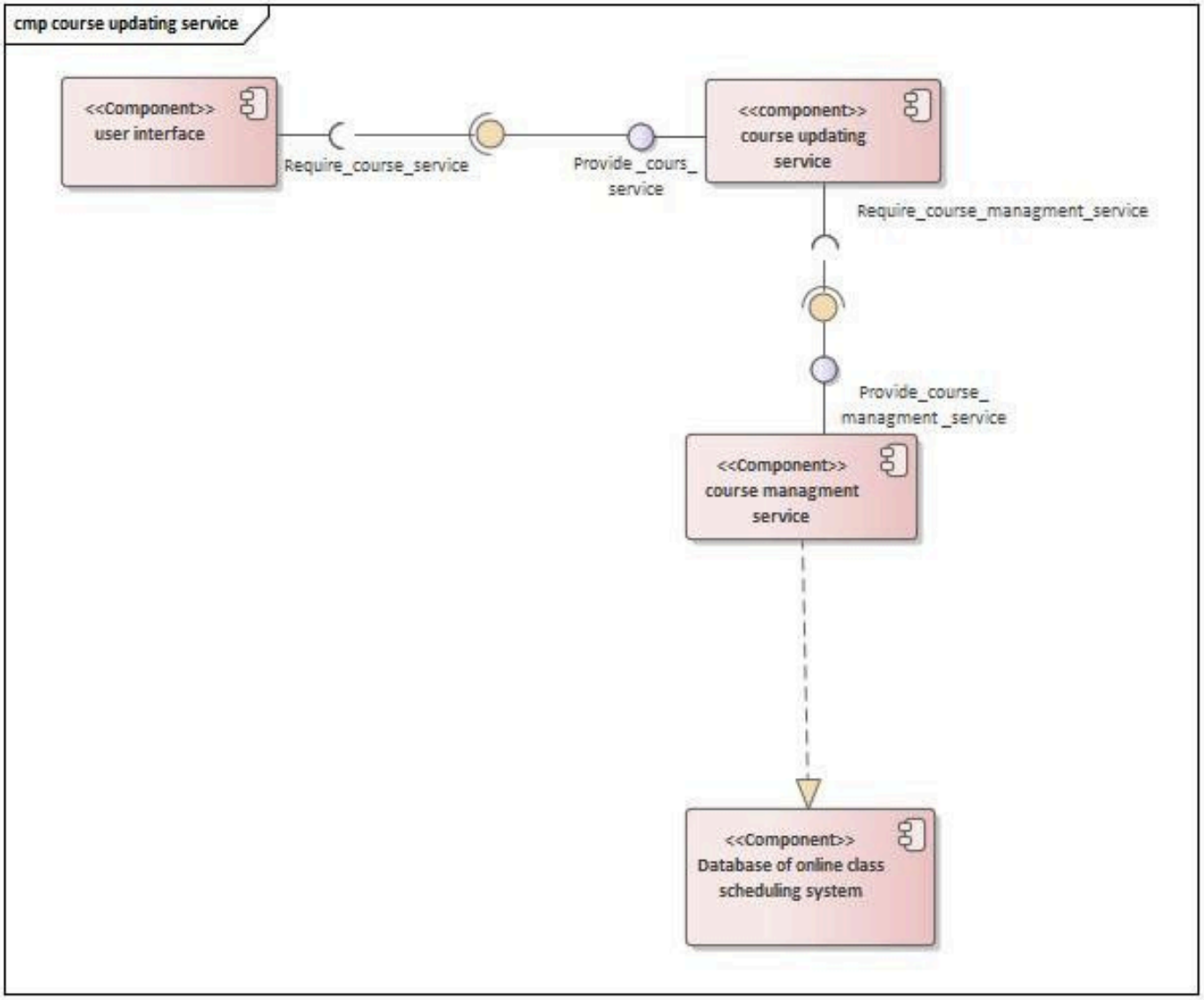


FIGURE 4.3


cmp class assigning service
<<Component>> use interface
Require class assigning service
Provide clas assigning service
<<Component>> class assigning service
Require class managment service
Provide class managment service
<<Component>> class managment
<<Component>> Database of online class scheduling system

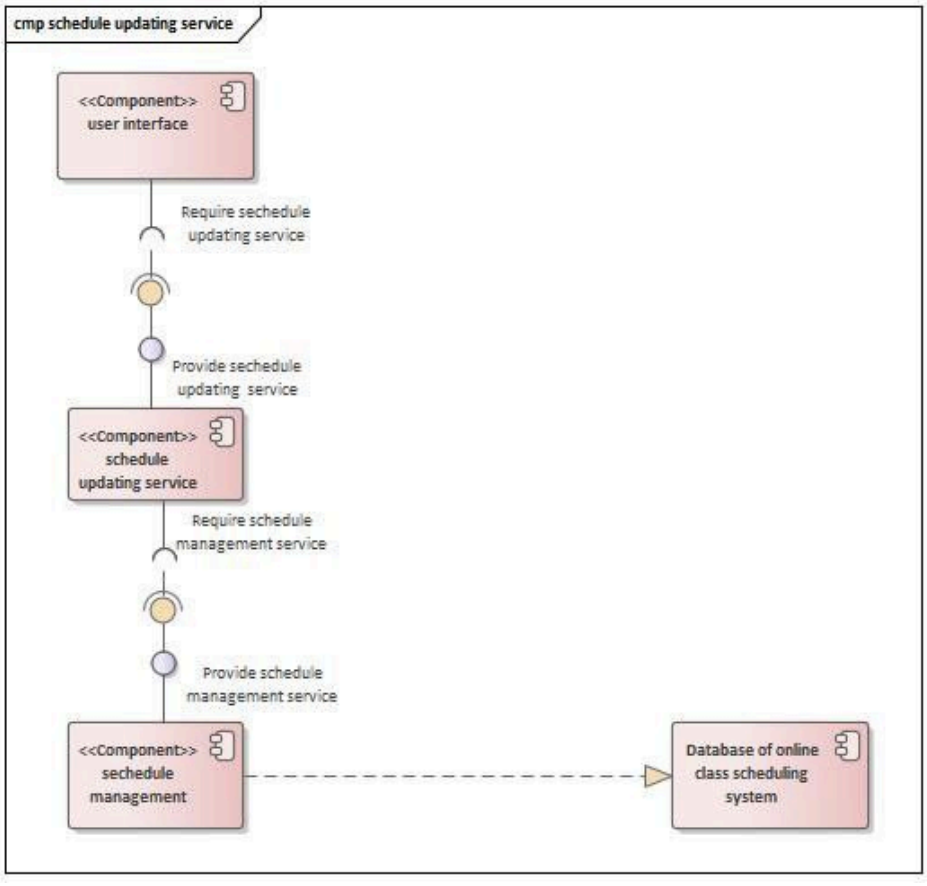
## COURSE UPDATING SERVICE

FIGURE 4.4




## SCHEDULE ASSIGNING SERVICE

FIGURE 4.5



## 4.5.3 DEPLOYMENT DIAGRAM

The system will be deployed as:

- Client Side: Computers used by students and staff to access the system.
- Server Side: Hosts the application and database.
- Database Server: MySQL database storing all system data.

Users connect to the server over the local network or internet.

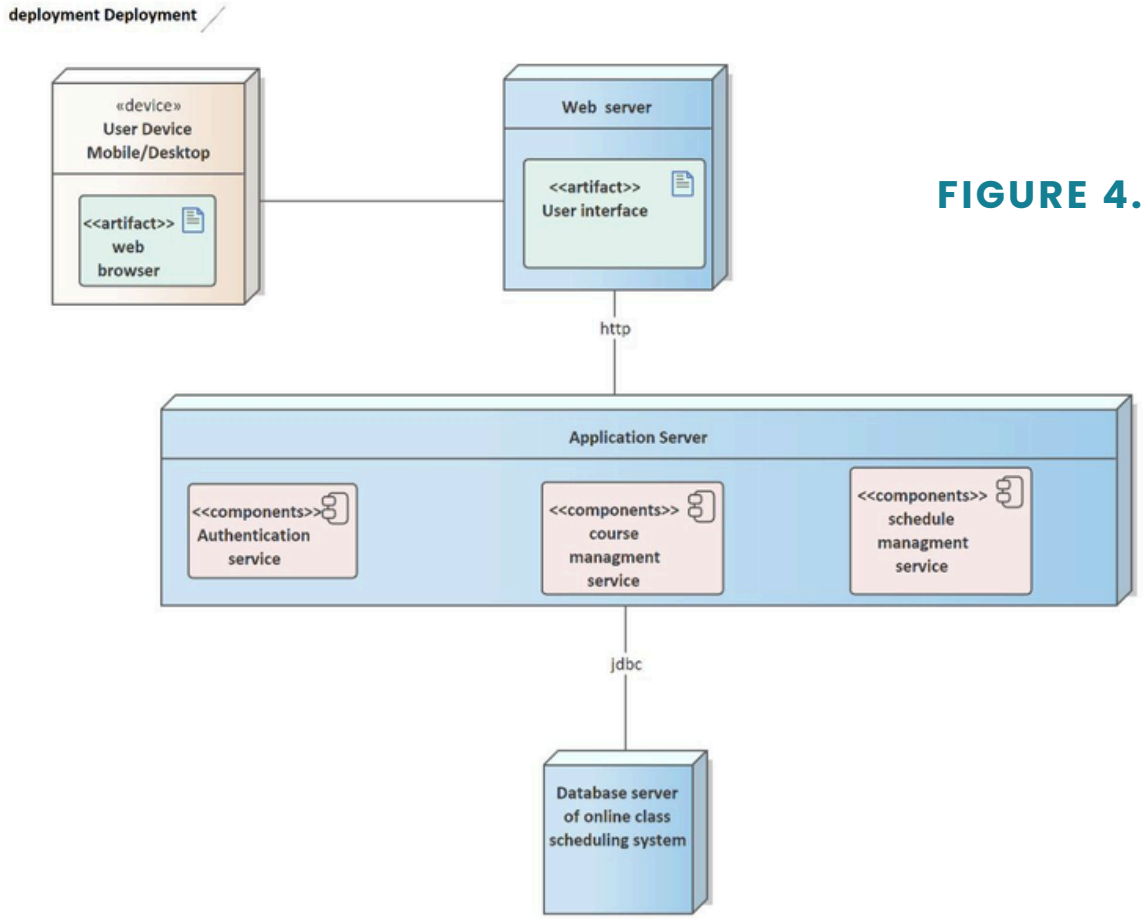


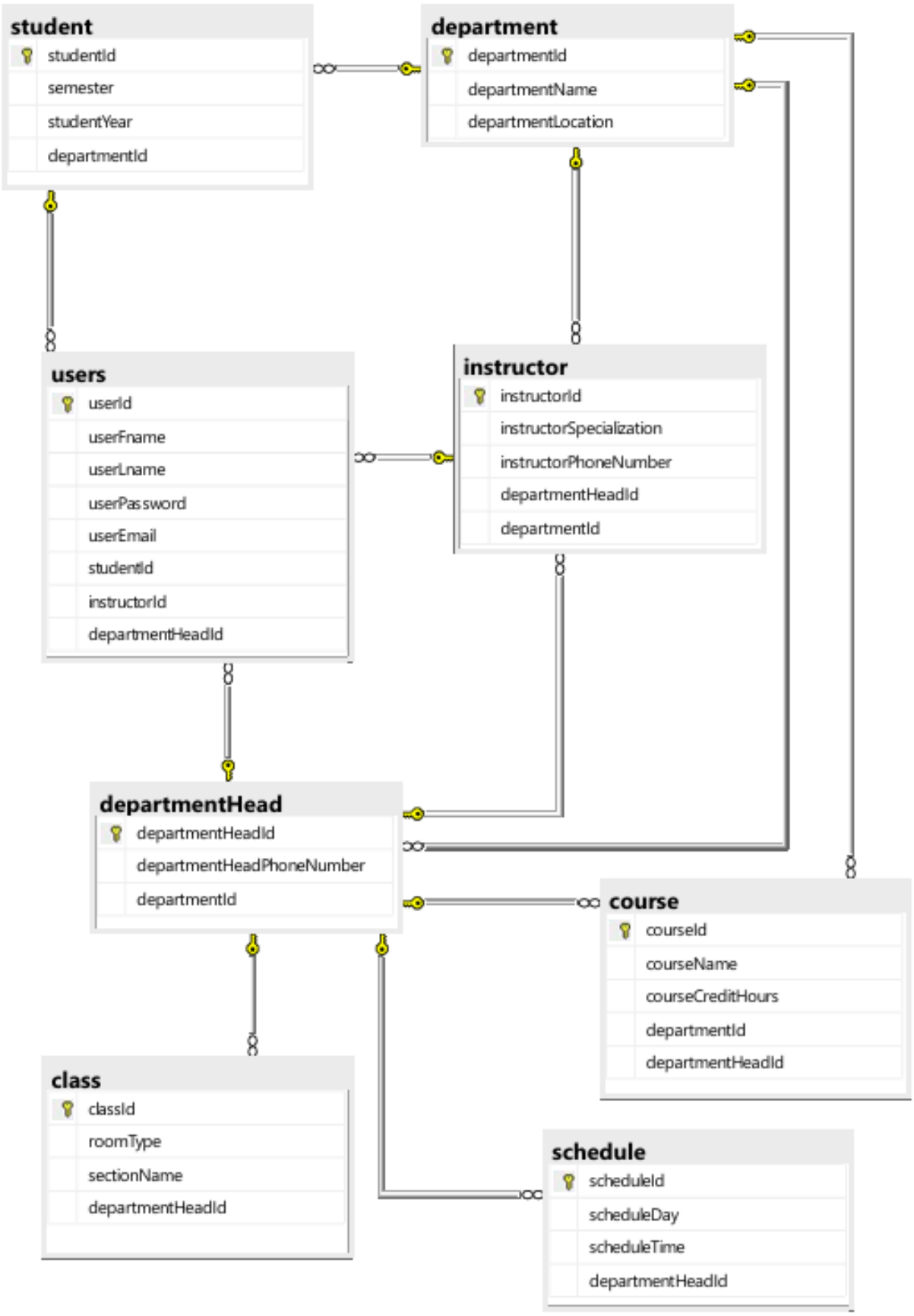

**FIGURE 4.6**

## 4.5.4 DATABASE DIAGRAM

The database design is based on the ER diagram and includes tables such as:

- Users
- Student
- Instructor
- Department
- Department_Head
- Course
- Class
- Schedule

Relationships and foreign keys ensure data consistency and integrity.


student
studentId
semester
studentYear
departmentId
department
departmentId
departmentName
departmentLocation
users
userId
userFname
userLname
userPassword
userEmail
studentId
instructorId
departmentHeadId
instructor
instructorId
instructorSpecialization
instructorPhoneNumber
departmentHeadId
departmentId
departmentHead
departmentHeadId
departmentHeadPhoneNumber
departmentId
course
courseId
courseName
courseCreditHours
departmentId
departmentHeadId
class
classId
roomType
sectionName
departmentHeadId
schedule
scheduleId
scheduleDay
scheduleTime
departmentHeadId

### 4.5.5 PERSISTENT DATA MANAGEMENT

The system uses MySQL to store persistent data.
Features include:

- Primary and foreign keys for relationships.
- Normalized tables to reduce redundancy.
- Backup support to prevent data loss.
- All user actions that change data are saved directly to the database.

### 4.5.6 ACCESS CONTROL AND SECURITY

Security is handled through:

- Username and password authentication.
- Role-based access:
  - Students: view schedules.
  - Instructors: view assigned schedules.
  - Department Head: manage schedules and classes.
  - Admin: manage user accounts.
- Basic validation to prevent unauthorized access.

### 4.5.7 GLOBAL SOFTWARE CONTROL

The system follows a menu-driven control flow:

- User logs in.
- System checks role.
- Appropriate menu and functions are displayed.
- User performs actions.
- System processes and updates the database.
- User logs out.

This ensures organized flow and easy navigation.

### 4.5.8 BOUNDARY CONDITIONS

Boundary conditions define system behavior at start and end:

- Startup: System loads, connects to the database, and shows login screen.
- Normal Operation: Users interact based on roles.
- Shutdown: User logs out and system closes connections safely.
- Error Handling: Invalid inputs or failures show clear messages.

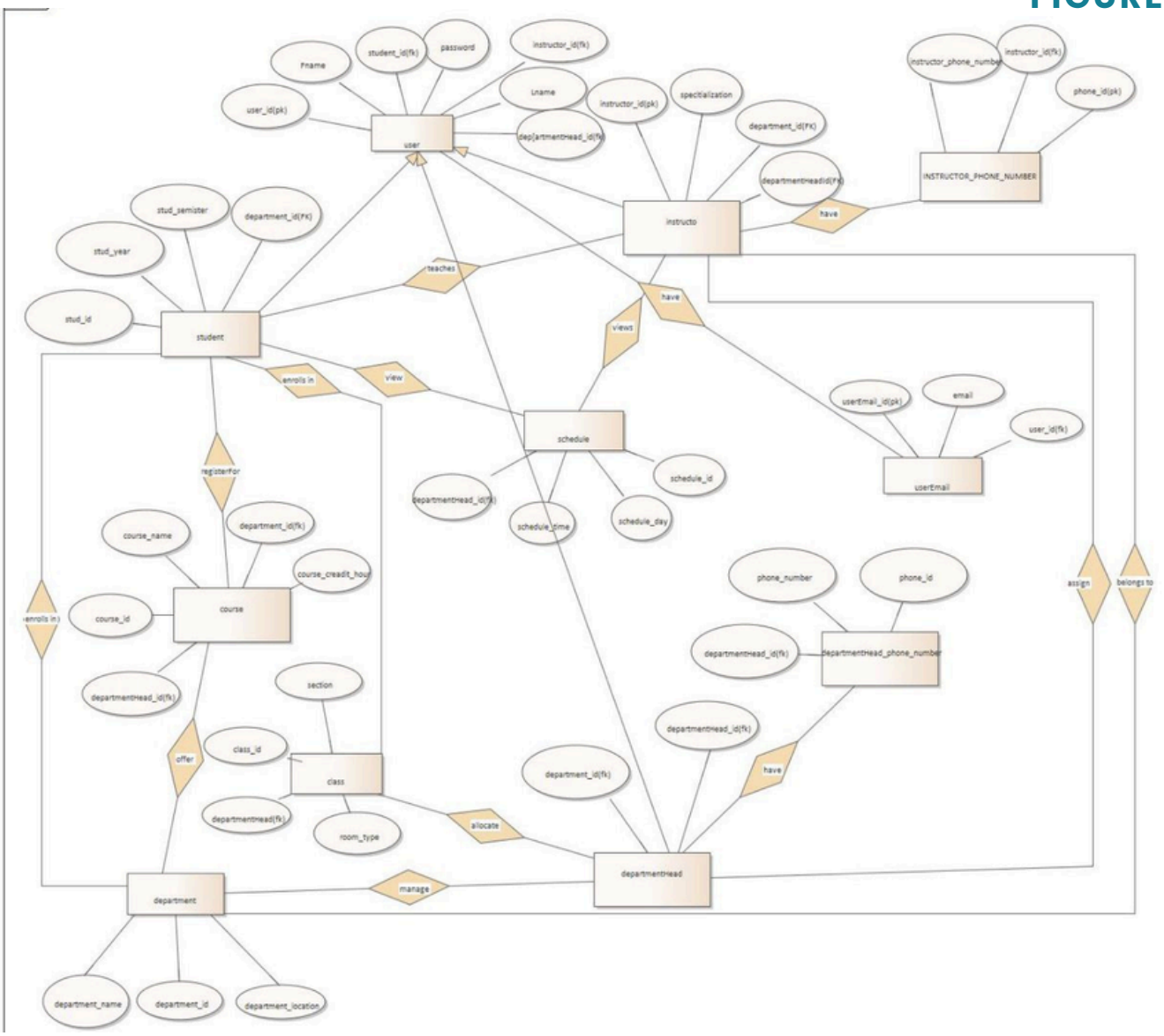
## 4.6 REFINED ER DIAGRAM

Based on analysis and normalization in Chapter Three, the refined ER diagram represents:

- Clear entities: Users, Student, Instructor, Course, Class, Schedule, Department, Department Head.
- Proper relationships such as enrolls, teaches, assigns, manages, and allocates.
- Well-defined attributes and keys to support system functions.

This refined ER diagram serves as the foundation for database implementation.

**FIGURE 4.8**



# CHAPTER FIVE: IMPLEMENTATION AND TESTING

## 5.1 TEST PROCEDURE

Testing was carried out to ensure that the Online Class Scheduling System functions correctly and meets the specified requirements. The testing process focused on verifying system functionality, correctness of outputs, and system stability.
The following testing approaches were used:

- Unit Testing
- Each module of the system was tested individually. Core components such as user login, schedule creation, course assignment, and schedule viewing were checked to confirm that they work as expected.
- Integration Testing
- After unit testing, related modules were combined and tested together. This ensured that interactions between modules such as students viewing schedules, instructors accessing assigned classes, and department heads managing schedules work properly.
- Functional Testing
- Functional requirements were tested against the system behavior. For example:
  - Students can view their schedules
  - Instructors can access assigned schedules
  - Department heads can create, update, and manage schedules
- Error Handling Testing
- The system was tested with invalid inputs to ensure it handles errors properly, such as incorrect login credentials or missing schedule data.

All detected errors were corrected before moving to the final stage of the project.

## 5.2 USER MANUAL PREPARATION

A simple and user-friendly user manual was prepared to guide users in operating the system. The manual explains system usage step by step using clear language.
The user manual includes:

- How to log in to the system
- How students view class schedules
- How instructors access assigned schedules
- How department heads create, update, and manage schedules
- Basic troubleshooting tips

The manual was designed so that users with basic computer knowledge can easily understand and use the system.

## 5.3 TRAINING AND INSTALLATION

Before deployment, basic training was planned for intended users, mainly department heads and instructors. The training focuses on:

- System navigation
- Schedule creation and management
- Viewing and updating schedules

The system installation process includes:

- Setting up the MySQL database
- Configuring the C++ application
- Connecting the system to the database
- Testing the system after installation to ensure proper functionality

## 5.4 STARTUP STRATEGY

The system startup strategy ensures smooth adoption and use of the Online Class Scheduling System. Initially, the system will be used in parallel with the existing manual scheduling process to minimize risks.

After successful validation:

- The system will fully replace the manual scheduling process
- Continuous monitoring will be done to identify issues
- Minor updates and improvements will be applied based on user feedback

This gradual startup approach helps reduce operational problems and ensures user confidence in the system.

# CHAPTER SIX: CONCLUSION AND RECOMMENDATION

## 6.1 CONCLUSION

This project focused on the design and development of an Online Class Scheduling System for the Department of Software Engineering at Wachemo University. The main goal was to replace the existing manual scheduling process, which is time-consuming and prone to errors, with a computerized system that is faster, more accurate, and easier to manage.

Throughout the project, system requirements were collected from users, analyzed carefully, and modeled using different diagrams such as use case, class, and ER diagrams. Based on these, the system was designed and implemented using C++ for application logic and MySQL for database management. The system allows department heads to manage schedules, instructors to view their assigned classes, and students to easily access their class timetables.

Testing results showed that the system performs the required functions correctly and helps reduce schedule conflicts, data redundancy, and manual workload. Overall, the project achieved its objectives and demonstrated how information systems can improve academic administration and efficiency within the department.

## 6.2 RECOMMENDATION

Although the system meets its current objectives, there are several areas where future improvements can be made:

**1 Web or Mobile Interface**

The system can be extended into a web-based or mobile application to allow access anytime and anywhere.

**2 Support for Other Departments**

The system is currently designed for the Software Engineering department. In the future, it can be customized to serve other departments and colleges in the university.

**3 User Roles Expansion**

Additional roles such as registrar or faculty dean could be included to enhance system control and coordination.

**4 Advanced Scheduling Features**

Automatic conflict detection and optimization features can be added to generate better schedules.

**5 Backup and Security Enhancements**

Stronger data backup and security mechanisms should be implemented to ensure data safety.

**6 Training and Maintenance**

Continuous user training and regular system maintenance are recommended to keep the system effective.

These improvements will make the system more robust, scalable, and useful for long-term use.

## 6.3 REFERENCE


- Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill.
- Sommerville, I. (2016). Software Engineering. Pearson Education.
- Elmasri, R., & Navathe, S. (2016). Fundamentals of Database Systems. Pearson.
- Wachemo University Academic and Administrative Guidelines.
- Lecture Notes – Software Engineering and Database Systems, Wachemo University.


## 6.4 APPENDIX

- The appendix includes additional materials related to the project, such as:
- Sample source codes (C++ implementation)
- SQL scripts for database and tables
- Sample input/output screenshots
- User manual
- Test cases and results
- Additional diagrams and documents
- These materials support the design and implementation discussed in this project.

```sql
CREATE DATABASE Class_Scheduling;
USE Class_Scheduling;

CREATE TABLE department (
    departmentId VARCHAR(10) PRIMARY KEY,
    departmentName VARCHAR(30) NOT NULL,
    departmentLocation VARCHAR(7)
);

CREATE TABLE departmentHead (
    departmentHeadId VARCHAR(10) PRIMARY KEY,
    departmentHeadPhoneNumber VARCHAR(14),
    departmentId VARCHAR(10),
    FOREIGN KEY (departmentId) REFERENCES department(departmentId)
);

CREATE TABLE student (
    studentId VARCHAR(10) PRIMARY KEY,
    semester INT NOT NULL,
    studentYear INT NOT NULL,
    departmentId VARCHAR(10),
    FOREIGN KEY (departmentId) REFERENCES department(departmentId)
);

CREATE TABLE instructor (
    instructorId VARCHAR(10) PRIMARY KEY,
    instructorSpecialization VARCHAR(25) NOT NULL,
    instructorPhoneNumber VARCHAR(14),
    departmentHeadId VARCHAR(10),
    departmentId VARCHAR(10),
    FOREIGN KEY (departmentHeadId) REFERENCES departmentHead(departmentHeadId),
    FOREIGN KEY (departmentId) REFERENCES department(departmentId)
);

CREATE TABLE users (
    userId VARCHAR(10) PRIMARY KEY,
    userFname VARCHAR(15),
    userLname VARCHAR(15),
    userPassword VARCHAR(20),
    userEmail VARCHAR(50),
    studentId VARCHAR(10),
    instructorId VARCHAR(10),
    departmentHeadId VARCHAR(10),
    FOREIGN KEY (studentId) REFERENCES student(studentId),
    FOREIGN KEY (instructorId) REFERENCES instructor(instructorId),
    FOREIGN KEY (departmentHeadId) REFERENCES departmentHead(departmentHeadId)
);

CREATE TABLE course (
    courseId VARCHAR(10) PRIMARY KEY,
    courseName VARCHAR(20) NOT NULL,
    courseCreditHours INT NOT NULL,
    departmentId VARCHAR(10),
    departmentHeadId VARCHAR(10),
    FOREIGN KEY (departmentId) REFERENCES department(departmentId),
    FOREIGN KEY (departmentHeadId) REFERENCES departmentHead(departmentHeadId)
);

CREATE TABLE class (
    classId VARCHAR(8) PRIMARY KEY,
    roomType VARCHAR(20) NOT NULL,
    sectionName CHAR(1) NOT NULL,
    departmentHeadId VARCHAR(10),
    FOREIGN KEY (departmentHeadId) REFERENCES departmentHead(departmentHeadId)
);
```

```sql
CREATE TABLE schedule (
    scheduleId VARCHAR(20) PRIMARY KEY,
    scheduleDay VARCHAR(25) NOT NULL,
    scheduleTime VARCHAR(15) NOT NULL,
    departmentHeadId VARCHAR(10),
    FOREIGN KEY (departmentHeadId) REFERENCES departmentHead(departmentHeadId)
);

INSERT INTO department VALUES ('D001','SE','B001');
select*from department
INSERT INTO departmentHead VALUES ('H001','+25151234567','D001');
select*from departmentHead

INSERT INTO student VALUES
('S001',1,2,'D001'),
('S002',1,2,'D001'),
('S003',1,2,'D001'),
('S004',1,2,'D001'),
('S005',1,2,'D001');
select * from student
INSERT INTO instructor VALUES
('I001','Teacher','+251945673278','H001','D001'),
('I002','Teacher','+251989765432','H001','D001');
select * from instructor

INSERT INTO users VALUES
('S001','Hikma','Mustefa','Uh21@Hi','fuia@email.com','S001',NULL,NULL),
('S002','Dagmawi','Lencho','Da34_lic','dagi@email.com','S002',NULL,NULL),
('S003','Tsion','Aklilu','Ts21#AKL','tsi@email.com','S003',NULL,NULL),
('S004','Bekalcha','Nuredin','Be23@nu1','beki@email.com','S004',NULL,NULL),
('S005','Kamil','Ersido','27!KPer2','kamil@email.com','S005',NULL,NULL),
('I001','Alemu','Kasa','19@alex2','kg7@email.com',NULL,'I001',NULL),
('I002','Solomon','Adena','so4$Ad','soli@email.com',NULL,'I002',NULL),
('H001','Belayneh','Kasahun','212$De32','belay@email.com',NULL,NULL,'H001');
select* from users
INSERT INTO course VALUES
('Math2011','DIS Maths',3,'D001','H001'),
('Seng5021','FSE',4,'D001','H001');
select* from course

INSERT INTO class VALUES
('C001','Lecture','A','H001'),
('C002','Lecture','B','H001');
select * from class
INSERT INTO schedule VALUES
('SCH01','Monday','2AM-5AM','H001'),
('SCH02','Tuesday','8AM-11AM','H001');
SELECT *FROM student s
INNER JOIN department d
ON s.departmentId = d.departmentId;
SELECT *FROM student s
LEFT JOIN department d
ON s.departmentId = d.departmentId;
SELECT *FROM student s
RIGHT JOIN department d
ON s.departmentId = d.departmentId;
SELECT* FROM student,department select * from student s
FULL OUTER JOIN department d ON s.departmentId=d.departmentId;
SELECT * FROM  users WHERE userFname like'H%';
SELECT userFname,userLname,userId FROM  users WHERE userFname like'B%';
SELECT userFname,userLname,userId FROM  users WHERE userId IN ('S001','S002','S003','S004','S005');
SELECT userFname,userLname,userId FROM  users WHERE userId BETWEEN 'S001' AND'S005';
SELECT userFname,userLname,userId FROM  users WHERE userFname like'B%' AND userId='S003';
SELECT userFname,userLname,userId FROM  users WHERE userFname like'B%' OR userId ='S003';
SELECT * FROM users  order by userFname,userLname;
SELECT * FROM users  order by userFname ;
SELECT * FROM users  order by userLname ;
```