Tutor Management System

Madeline Abbot and Ben Hogan

3/15/2025

INFO-C451

Professor Jabin

Table of Contents

Proposal

Title: Tutor Management System

Madeline Abbott and Benjamin Hogan

Problem Statement

We are looking to connect students to tutors in a way that is functional and easy to understand for users who may not be used to computer systems. By allowing tutors and students to see open schedule times to sign up for they will be able to work around their own schedules and match with another student or tutor easily.

System Functionality

The Tutor Management System (TMS) will allow tutors/students to:

- Check Their Current Schedule: Any previously scheduled appointments will be seen here.
- Schedule Appointments: Set up and manage tutor/student appointments.
- Track Grades and Assignments: Students able to update their own grades to track progress.

Objectives

- Centralized Tutoring System: Provide a single platform for tutors and students to get matched together to provide students with help.
- Efficient Scheduling: Develop a more streamlined approach to appointment scheduling and management to improve efficiency.
- Grade Tracking: Enable easy tracking of medical histories and treatments.
- Data Security: Ensure that patient data is stored securely and access is controlled.

System Requirements

- User Capabilities:
  - Tutors: Access their schedule, see students looking for a tutor, and adjust their schedule if they want to add a student.
  - Student: Access their schedule, see tutors who match academic needs, and add appointments if they need help.
  - Administrators: Manage user accounts and control access levels.

● Functional Requirements:

○ Frontend: A user-friendly, engaging, and appealing interface for interacting with records and scheduling.

○ Backend: A reliable and capable system for handling data operations.

○ Database: A relational database to store patient information, appointment schedules, and medical histories.

Typical Customers

● University Students: Any students who are struggling in a class at the college level.

● Tutors: Current or former students who want to be able to help other students succeed.

Project Planning

Abbott and Hogan 2

● Software:

○ Frontend: HTML, CSS, JavaScript, as well as jQuery for easier DOM manipulation.

○ Backend: PHP for server-side scripting and Java for complex operations with the database.

○ Database: MySQL for relational data storage.

● Hardware: PCs or servers for development and deployment.

● Network: Standard internet connectivity for accessing the system.

Development Approach

● Frontend: Use HTML and CSS for layout and design, and implement interactive features with JavaScript and jQuery.

● Backend: Use PHP to handle server-side logic and interactions with the database.

● Database: Design a relational schema in MySQL for managing patient records, appointments, and medical histories.

● Security: Implement user authentication and authorization in PHP, and use prepared statements in SQL to prevent SQL injection.

Development Plan

● Weeks 1-2:

○ Set up a development environment (PHP, MySQL, HTML/CSS/JS).

○ Design database schema and set up MySQL database.

● Weeks 3-4:

○ Develop the frontend interface with HTML, CSS, and JavaScript.

○ Implement basic PHP scripts for CRUD operations on patient records.

○ Create simple Java classes representing the classes in the program.

● Weeks 5-7:

○ Develop the ability to schedule appointments.

○ Add features for managing medical histories and submitting reports.

● Weeks 8-10:

○ Record midterm report.

○ Implement user authentication and authorization.

○ Ensure data security and integrity.

● Weeks 11-13:

○ Test the system for bugs and usability issues, including adding in test cases.

○ Refine and optimize features based on feedback.

● Week 14:

○ Prepare final documentation and presentation.

● Week 15:

○ Record demo and finalize project for submission.

Title: System Sequence Diagram and Activity Diagram

Madeline Abbott and Benjamin Hogan

Sequence Diagrams

Use Case 1: Updating Student Records

Actor:

User: The person who interacts with the system (ex. tutor, admin, etc.),

Object:

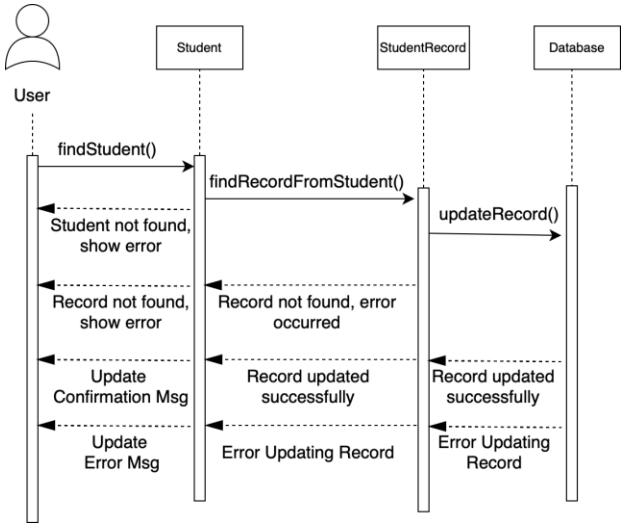
Student: the student whose records are being managed.

StudentRecord: the record of the student.

Database: the storage system for student records.

Steps of Updating a Student Record:

1. The User opens the “Manage Student Records” section in the interface.
2. The User identifies the Student object based on the input of StudentID.
3. The system sends the request to the server and finds the corresponding StudentRecord.
4. The server returns a response finding the record (either a success or error) to the interface.
5. The system processes the request and communicates with the Database by either inserting, updating, or deleting a StudentRecord.
6. The server returns a response updating the record (either a success or error) to the interface.
7. The changes are saved to StudentRecords via the Database.



Use Case 2: Scheduling an Appointment

Actor:

User: The person who interacts with the system (ex. tutor, admin, etc.),

Object:

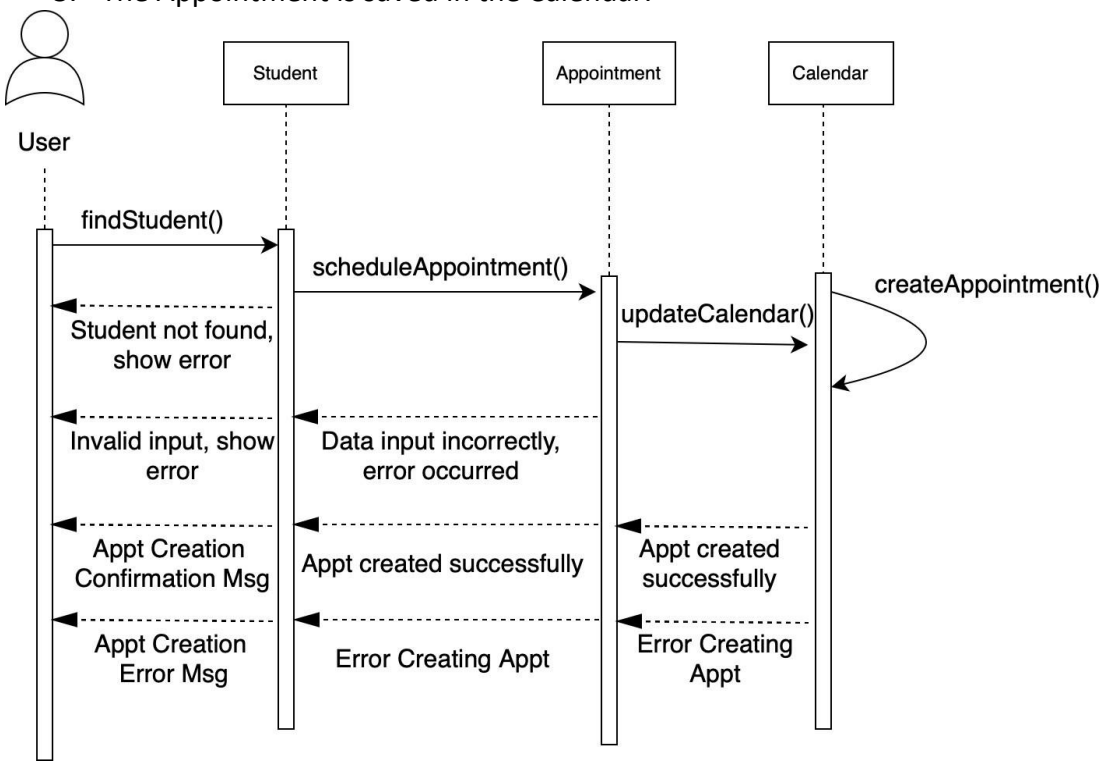
Student: The person for whom the appointment is being scheduled.

Appointment: Represents the transaction of scheduling an appointment including the time, date, and subject.

Calendar: The tool used to represent dates and times, as well as schedule appointments.

Steps of Scheduling an Appointment:

1. The User selects the Student which will have an Appointment by using their StudentID.
2. The User selects a time and date for the Appointment.
3. The system checks the Calendar to validate the time and date for the Appointment.
4. The system either returns an error or confirmation message for the Appointment.
5. The Appointment is saved in the Calendar.



Activity Diagrams

Use Case 1: Creating a Student and Student Record

States:

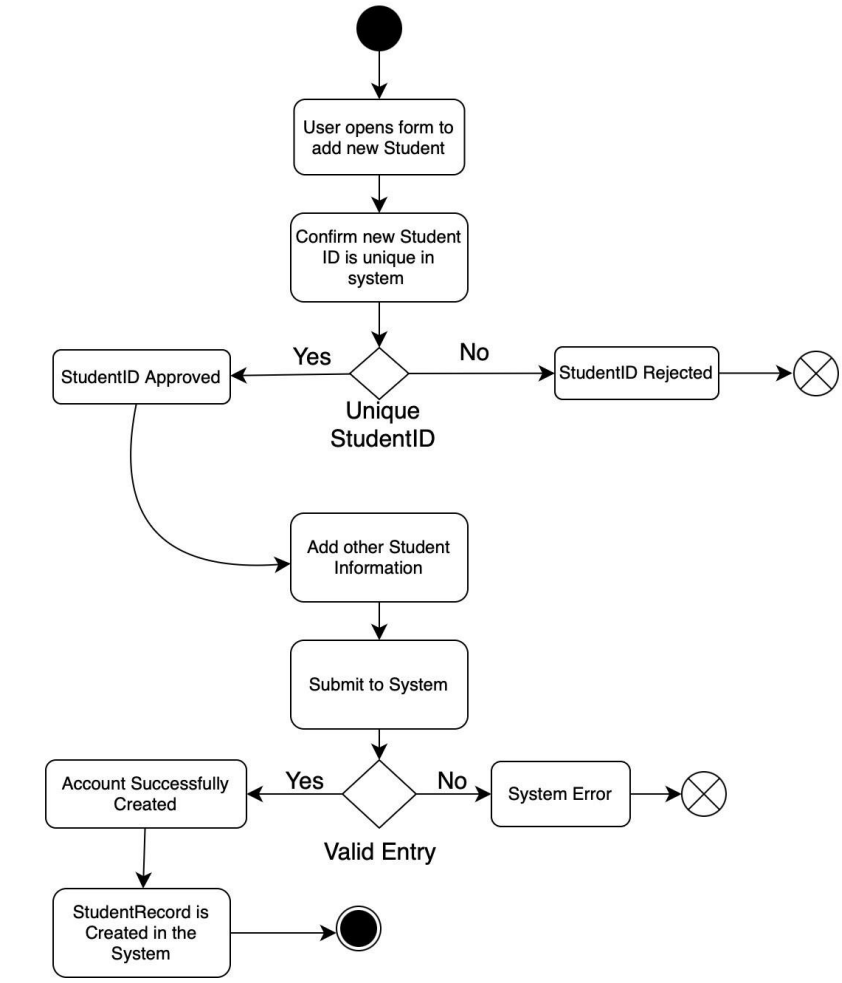
Initial State: The user opens the system and begins the student creation process.

Final State: 1. The user receives confirmation of the new student creation, and a new student is added to the database. 2. The user receives an error message with an explanation, and a new student is not added to the database.

Actions:

The user is logged in and selects the “Add New Student” option. The user is given a form and inputs student information including name, student ID, date of birth, major, address,

contact information, tutoring concerns, and other notes. User submits Student details. System validates the entered information. If the validation is successful, the system creates a StudentRecord. System saves the new StudentRecord to the database. System returns a success or error message to the interface accordingly.


User opens form to add new Student
Confirm new Student ID is unique in system
Yes
No
StudentID Approved
StudentID Rejected
Unique StudentID
Add other Student Information
Submit to System
Account Successfully Created
Yes
No
System Error
Valid Entry
StudentRecord is Created in the System