Software Requirement Specification

ECE651 Project Evolution 3

Team 5

Haolou Sun, Haoyuan Geng, Hiep Nguyen, Hunter Shen, Yuyu Hsieh

Introduction

The attendance management system is designed to facilitate the attendance taking process and make attendance management easier for the professors. This system will help to automate the process for taking attendance and managing student enrollment. It aims to help the professor save time and energy, and ensure accuracy and security of attendance records. It also allows students to easily access their own attendance record, and customize how they want to receive notification for any attendance record changes.

There are 3 separate applications working together to make the attendance management system work. Faculty and student users need to first be registered by an admin user through the **User admin app**. The admin user will then create courses via the **Course Management app** and set up the course instructor/enroll students to relevant courses. After these initial set up, the faculty and student users will then be able to log into the **Attendance Management system** to perform various operations related to attendance management.

For the **User Admin app** and **Course Management app**, we utilized JavaFX to implement the user interface, enhancing the user experience for administrators in managing personnel and course registrations with greater convenience and intuitiveness. As for the **Attendance Management System**, we implemented a web-based GUI, making our program more user-friendly and enabling instructors to quickly familiarize themselves with our application.

Functional Requirement

The Functional requirements are described in detail below for each of the 3 applications.

1. **User admin app**
   User admin app is a standalone application which is responsible for all the user related operations like user creation, modification and deletion. There are 2 main types of users (students and faculty), and the user admin app governs both of them. Only the admin super users are able to access and log in to the user admin app. The detailed functional requirements are listed below:
   1. The user admin app will run on a text based terminal and allow the admin user to interact with the system via typing inputs.

2. The admin super user shall be able to securely log in to the system via a predefined username and password that was set up ahead of time by the development team.
    a. Only one super user account is required.
    b. The super user information (username and password) need to be stored in the database.
3. Super users shall be able to register students in the system by either manually input the student information one by one, or by uploading a csv file containing the required information for bulk student registration.
    a. information required for student registration:
        i. netid: must be in valid netid format (english characters + digits)
        ii. default password
        iii. legal name: must include both first name and last name or middle name (optional), separated by a single space
        iv. preferred name (optional): must include both first name and last name, separated by a single space
        v. email
        vi. phone (optional): must contain exactly 10 digits without any other characters
    b. The provided csv file must contain the corresponding header in the exact same wording for the user admin app to recognize the different information. (header names: netId, password, legalName, email, phone, preferredName)
    c. User admin app shall perform pattern checking for the input information/file and prompt the super user for a new input if the supplied information does not meet the required format
    d. User admin app shall create a user account in the attendance management system for every student registered via the admin app. Students shall be able to log into their accounts in the attendance management system using their netid and the default password.
4. Super users shall be able to register faculty in the system by either manually input the faculty information one by one, or by uploading a csv file containing the required information for bulk faculty registration.
    a. information required for faculty registration:
        i. netid: must be in valid netid format (english characters + digits)
        ii. default password
        iii. legal name: must include both first name and last name, separated by a single space)
        iv. email
        v. phone (optional): must contain exactly 10 digits without any other characters

    b. The provided csv file must contain the corresponding header for the user admin app to recognize the different informations
    c. User admin app shall perform pattern checking for the input information/file and prompt the super user for a new input if the supplied information does not meet the required format
    d. User admin app shall create a user account in the attendance management system for every faculty registered via the admin app. Faculty shall be able to log into their accounts in the attendance management system using their netid and the default password.
5. The user passwords should be salted and hashed with security standard equivalent or higher than SHA-2 before storing in the database.
6. Admin super users shall be able to use the user admin app to update user information on behalf of students and faculties. The information that can be edited shall cover and only cover below information:
    a. for students: preferred name, email, phone, password
    b. for faculties: email, phone, password
7. The admin app shall allow the admin user to remove a student account or a faculty account by manually selecting the account they want to delete.
    a. The user admin app shall display a warning message when the admin user tries to delete an account. Admin users need to confirm their intention again before the deletion would take place.
    b. When a student account is deleted, the deletion will not cascade to delete the student's enrollment record, their attendance record and their notification preference.
    c. When a professor account is deleted, the deletion will not cascade to delete the instructor information for the courses/sections they teach and the courses/sections itself.
8. The user admin app shall allow the admin super user to perform operations on different users by selecting the users they want to modify/delete.
9. The admin app should utilize the JavaFX framework to implement graphical user interface interaction with users.
    a. It should feature a graphical login interface where users can input their username and password, then click a login button to authenticate.
    b. After logging in, it should have two main sections. One section is responsible for manually inputting student and professor information for creation or update. The other section is responsible for importing people's information from files.
    c. In the manual input section, clicking the "Create Student" or "Create Professor" buttons allows us to input relevant information to create a student or a professor, respectively. Clicking the "Update Student" or "Update Professor" buttons enables us to select the information to update,

input the new details, and update the student or professor information accordingly.

d. In the manual input section, clicking the "Create Student" and "Create Professor" buttons allows us to input relevant information to create a student or a professor, respectively. Clicking the "Update Student" and "Update Professor" buttons enables us to select the information to update, input the new details, and update the student or professor information accordingly.

10. Standalone app: The standalone application should be executable everywhere with the connection to the setup database.

2. **Course management app**

Course management app is a standalone application responsible for all the course related operations like course/section/session creation, student enrollment, modification and deletion. Only the admin super users are able to access and log in to the course management app. The detailed functional requirements are listed below:

1. The course management app will run on a text based terminal and allow the admin user to interact with the system via typing inputs.
2. The admin super user shall be able to securely log in to the system via a predefined username and password that was set up ahead of time by the development team.
    a. Only one super user account is required.
    b. The super user information (username and password) need to be stored in the database.
3. Super users shall be able to create courses/sections, set up the course instructor, and enroll students to the course/section through the course management app.
    a. One course should contain one or more sections. The system will not support adding additional sections to a course after the course has been created.
    b. One section should have one instructor who is an existing professor in the system created by the user admin app. The behavior of entering a professor that does not exist in the system is undefined. The admin user should not attempt to enter a professor unknown to the system.
    c. One section should have one or more sessions (lectures). Each session is tied to one specific time when that session will take place.
    d. All the sessions for one section are created via the course management app during the initial course set up. Once set up, the super user will not be able to add/modify/remove sessions for a specific section.
    e. One section will have one or more students enrolled in it. The enrolled students must be an existing student in the system created by the user admin app. The behavior of entering a student that does not exist in the

system is undefined. The admin user should not attempt to enroll an unknown student.

f. The same student can be enrolled in multiple courses (if any course has more than one section, which section is enrolled needs to be specified)
g. One instructor can teach multiple courses/sections.
h. Courses can be created by either manually input the course information one by one, or by uploading a csv file containing the required information for bulk courses creation.
    i. information required for courses creation:
        1. course id: must be in the format of <Department name><course number> (eg ECE651)
        2. course name
        3. section id: must be a numeric number (eg 1, 2)
        4. instructor netid: must be in valid netid format, and the instructor must been created in the system via the user admin app
        5. lecture time: must be in YYYY-MM-DD HH:MM:SS format
        6. student netid (for the enrolled students): must be in valid netid format, and the student must have been created in the system via the user admin app
        7. status: ENROLLED
i. For bulk course creation, there should be 2 input csv files (one for course creation and student enrollment, one for session creation). Both of the csv should contain a header. The detailed csv format is described below:
    i. course creation and student enrollment file (instructor_netid must show up before student_netid)

course_id	course_name	section_id	instructor_netid	student_netid	status
ECE651	software engineering	1	Af123	Haolou123	ENROLLED
ECE651	software engineering	1	Af123	yuyu123	ENROLLED
ECE651	software engineering	2	Af123	hunter123	ENROLLED

    ii. session creation file

course_id	section_id	lecture time
ECE651	1	2024-04-12 15:45:00
ECE651	1	2024-04-15 15:45:00
ECE651	2	2024-04-12 15:45:00
ECE651	2	2024-04-15 15:45:00

    iii. course management app shall perform pattern checking for the input file and prompt the super user for a new input if the supplied information does not meet the required format

    j. For manual creation, users will be prompted to choose one instructor out of all the professors existing in the system so instructor netid is not required.
    k. course management app shall create the courses/sections in the attendance management system and link the instructor and enrolled student's account to it. After this set up, students and faculty should be able to log into their accounts in the attendance management system and see the relevant courses they are enrolled/teaching.
4. After courses are created, admin users shall still be able to enroll more students to the courses-sections by manually selecting students to enroll. If they want to batch enroll students via csv file, they must do it in the same step when creating the course. The behavior for providing a new csv file with courses that have already been registered and new students is undefined and should be avoided.
5. The course management app shall allow the super user to remove a course by selecting one course out of all the existing courses. A warning message must be displayed to warn the user of the destructive action. The super user has to confirm their intention before the deletion will take place.
    a. Upon deletion of the course, the change will be cascaded to delete all the related sections, sessions, instructor relationship (but not the professor account), enrollment record (but not the student account), attendance record, and notification preferences
6. The course management app shall allow the super user to remove a section by selecting one course and selecting one section. A warning message must be displayed to warn the user of the destructive action. The super user has to confirm their intention before the deletion will take place.
    a. Upon deletion of the section, the change will be cascaded to delete all the related sessions, instructor relationship (but not the professor account), enrollment record (but not the student account), attendance record, and notification preferences
7. The course management app shall allow the super user to update courses/sections by changing the course name and the section instructor.
8. The course management app shall allow the super user to modify student enrollment status from enrolled to dropped by selecting one course-section and one currently enrolled student
    a. the system does not support enrollment edits from dropped to enrolled
    b. Once a student is dropped from a course, their past attendance record will not be deleted in the system, but their name will not show up in the attendance taking process, nor will it show up in the students browsing / searching results.
9. The Course Management app should utilize the JavaFX framework to implement graphical user interface interaction with users.

a. Upon running the program, it should initially display a login interface. Administrators can input their account and password, then click the login button to authenticate and access the system.
b. After logging in, there should be a column on the left side of the screen displaying the names of currently existing professors. Clicking on a professor's name should lead to the professor's course page. On the right side of the screen, there should be a list of currently existing course names. Clicking on a course name should lead to the respective course page.
c. On the homepage after logging in, there's a button labeled "Create a Course." When clicked by the administrator, they can follow the prompts to create a new course, set the course instructor, determine the lectures included, and register students.
d. After selecting a course, the administrator can choose to add lectures to the course, change the course instructor, or enroll students in the course. The administrator also has the option to delete the course or the current section, but will receive a system warning. If they click the confirmation button, the deletion will be successful.

10. Standalone app: The standalone application should be executable everywhere with the connection to the setup database.

3. Attendance Management System

The attendance management system is the main application for the faculty to take attendance and manage attendance records, and for students to view their attendance record and manage notification preference. Only faculty and students who have been registered via the user admin app will be able to access the attendance management system via their netid and predefined password. The detailed functionality are described below:

1. The attendance management system will run on a text based terminal and allow the faculty and student users to interact with the system via typing inputs.
2. Users shall perform authentication to gain access to the attendance management system. They will be able to log into the system using their netid and a password preset by the user admin app.
3. **Faculty Access:** faculty users will be able to perform below actions once they login to the system:
    a. Faculty users shall be able to take attendance for the course/section they teach. The detailed process is specified below:
        i. They shall be prompted to select a course and section from a menu option of all the courses/sections they teach.
        ii. After course and section selection, they shall be prompted to select one session out of all the sessions for the course/section. Sessions will be displayed in terms of the time which the session is supposed to take place.

iii. If the session which was selected by the faculty user already had an attendance record before, the new attendance record will overwrite the old record in the system.

iv. Once the course/section and session has been selected, the attendance taking process will start. All the enrolled student's names will show up at the console one by one. If the student's preferred name has been set up via the user admin app, the system will display their preferred name. If not, the legal name will be used for display. The instructor will be prompted to select one attendance status out of below options:

1. present
2. absent
3. tardy

Instructor should type in one option, and the system will record the attendance for that student and move on to the next student.

v. Once all the students' attendance records have been recorded, the system will end the attendance taking process and prompt the professor for the next action they want to perform.

b. Faculty users shall be able to modify the attendance for a student for a particular session in a class. The detailed process is specified as below:

i. The faculty user needs to first select one course/section out of all the courses and sections they teach.

ii. After that, they can choose to either browse all the enrolled students of the course/section, or search for a student using their legal name.

1. If they choose to browse, the system will display all the enrolled students with an option number before it. The User can then type in the option number to choose the corresponding student.
2. If the user chooses to use the search functionality, the system will display a list of all students with the matching name so that the faculty can choose one student out of the options. If no students were found by the search keyword, the system shall prompt the user for a new action.

iii. After one student is selected, the user will then be prompted to choose one session out of all the sessions for the course/section.

iv. After session is selected, the user will then be prompted to choose one attendance status for the student among the below options:

1. present
2. absent
3. tardy

        v. The user will type in the number before each option into the system, and the attendance record will be updated in the system.
    c. Faculty users shall be able to select one class-section they teach and export attendance report.
        i. The report should cover all of the enrolled students, each with one class participation score.
        ii. Class participation score is calculated as the average score for each of the lectures from day 1 to present. Students who were “present” for a lecture will get 100% of participation for that lecture. “Tardy” will count towards 80% of participation, and “absent” would get 0%. Score will be shown in an absolute score out of 100.
        iii. The faculty user shall be able to choose whether they want to export the attendance report into json format or pdf format.
4. **Student Access:** student users will be able to perform below actions once they login to the system:
    a. Student users shall be able to select one of the course-sections they are enrolled in and change the notification preference.
        i. The default notification preference is email notification for every course/student.
        ii. The notification preference shall influence the status update notification whenever their attendance record was updated, but not the weekly attendance report. (weekly attendance report will be sent no matter what their notification preference is)
    b. Student users shall be able to select one of the course-sections they are enrolled in and generate a summary and detailed attendance report (1 file containing both of them per class)
        i. The attendance report will contain a summary of their average participation score for the course from the first day to the current moment.
        ii. The attendance report will contain their detailed participation record, which would have one participation status for each lecture of the course.
        iii. The attendance report will be exported as json/pdf
5. This program should utilize the Spring framework to implement a web-based GUI, providing professors and students with a more convenient and intuitive way to use the application, thereby reducing their learning curve. Below are the detailed description of the specific functionalities:
    a. Professors and students should be able to input their NetID and password, then click the login button to authenticate and access the system.
    b. After professor logging in, the program should navigate to the dashboard page, where all the courses taught by the professor are displayed. By clicking on a link, the professor can access the corresponding course.

Then, the professor should be presented with three possible options to choose from: "Take Attendance", "Search Student", and "Generate Section Report".

i. If the professor selects "Take Attendance", the system should display all possible lectures for the current course. The professor can then choose one of the lectures to begin attendance. Upon starting the attendance, the screen should show the names of all students currently in the class. The professor can then click on each name to select the attendance status (such as present, absent, late) for each student and submit the attendance record.

ii. If the professor chooses "Search Student", they will be directed to a page where they can input the student's name (partial names are also acceptable, as long as they are consecutive parts of the name) and click "Search" to display results. Then, they can select the searched name and the lecture to update the attendance status for that student in the selected lecture.

iii. If the professor selects "Generate Section Report," the system will navigate to a report page, displaying the cumulative attendance scores of all students in the current section. In this interface, the professor can also choose to export the report in different formats such as PDF, JSON, or XML by selecting the corresponding options for "Export PDF," "Export JSON," or "Export XML."

c. After logging in, the program should direct students to their dashboard interface, displaying two possible options: "Change Notification Preference" and "Get Attendance Report."

i. If the student selects "Change Notification Preference," they will be directed to a page displaying all the courses they are currently enrolled in. From this list, the student can choose one course. Then, they can select a notification method from a dropdown menu and click the submit button to make the change.

ii. If the student selects "Get Attendance Report," they will be directed to a page displaying all the courses they are currently enrolled in. From this list, the student can choose one course. Then, they can view the overall attendance score for that course and choose to export the report in PDF, JSON, or XML format.

Non-functional Requirements

1. **Usability:** The system shall pass a usability test from 10 users with no programming background by at least 90% of all users being able to perform the tasks with no expert guidance.
2. **Reliability:** The system shall ensure the reliability and accuracy of the attendance records.
3. **Concurrency**: The system shall be able to handle multiple students and faculty users to login and perform different operations at the same time. It has to ensure the data stays consistent even when multiple users try to modify the same data concurrently.
4. **Performance**: There are no performance requirements from the client as long as the system can allow multiple users to access the system concurrently.
5. **Security**: Users have to authenticate themselves before being able to access the system. Plain text will be sent between client and server, but passwords shall be salted and hashed before storing in the database.
6. **Data Management:** The system shall use DAO pattern to implement data management. The data should be stored in a RDBMS.
7. **System Implementation:** The system shall be implemented in Java with no framework or pre-existing solutions.
8. **System Architecture:**
    a. Attendance management system should follow a Client/Server architecture
        i. The repository of student attendance should be stored in a centralized server which provides services to standalone console applications.
        ii. The server will run on a linux server (VM)
        iii. Clients will be light-clients. I.e., no data or major processing should happen on the client side (input processing is ok).
    b. User administration app and course management app should both be standalone application (no client/server architecture)