## Team Design Report

Group - I

**STUDENT ASSIST APP**

Team members:

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME OF THE STUDENT** | **CWID** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

Course: **CS-487 Software Engineering I**

Date:

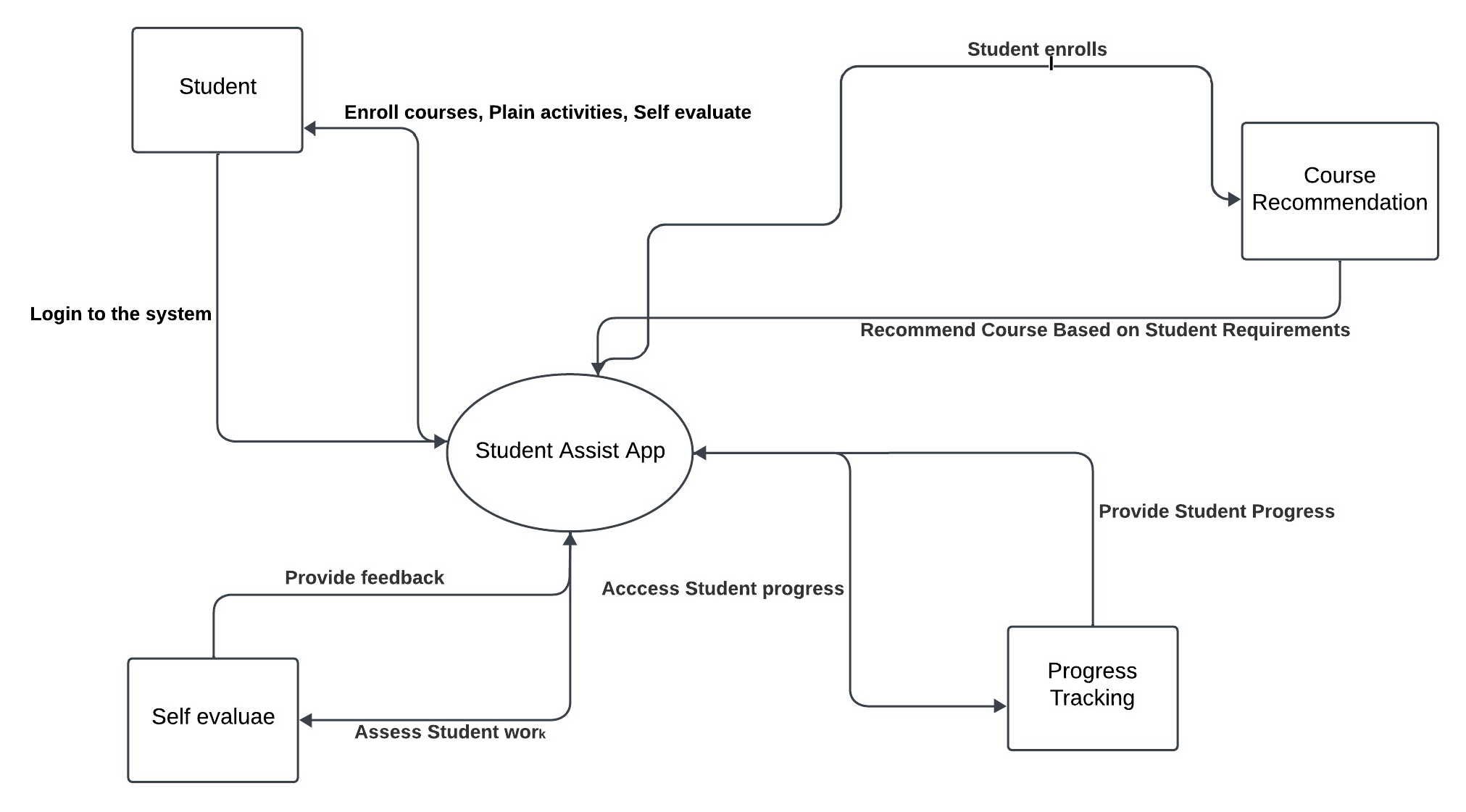
# Section 1: Proposed design of the application:

The "Student Assist" app provides a transformative approach to improving the student experience. Its primary purpose is to support students in their academic journeys by providing real-time coaching using powerful AI algorithms. These algorithms are not static tools rather they have dynamic components that continuously learn and adapt to each user's specific goals and preferences.

The app's primary goal is to improve academic performance. It diligently tracks the outcomes of each user's actions and decisions, creating a feedback loop which is tailored to their individual goals. This means that the app is more than just a passive aide; it is an active collaborator in helping students reach their academic goals.

The "Student Assist" software is geared to providing maximum utility to its users. It accomplishes this by providing features that are not only important but also extremely usable, allowing students to quickly incorporate them into their regular routines. Its value proposition goes beyond convenience and revolves around the overall enhancement of the student experience. The app's value proposition may be summarized as three main pillars: individualized support, continual innovation, and strong security. It listens to each user's specific requirements and goals and then provides individualized advice and support. It does not stop there; it actively attempts to improve its instructions in response to user feedback and evolving academic needs. Importantly, it protects user data with strict security measures, instilling trust in a safe academic support environment.

# Section 2: System/Context Model



## Application Interaction Description:

The Student Assist App interacts with other components of the larger system in the following ways:

User Profile Setup: The User Profile Setup component communicates with external systems or databases to validate and save user profile information. It may work with authentication systems to provide secure user login and access to the app. It interfaces with the User component to retrieve user information and update the user profile when necessary.

Daily Planner: The Daily Planner component communicates with the User component to retrieve user preferences and customizing options. It may work with calendar systems to sync and display user schedules and events. The Daily Planner component may also interface with other services or APIs to provide extra functionality, such as weather predictions or task reminders.

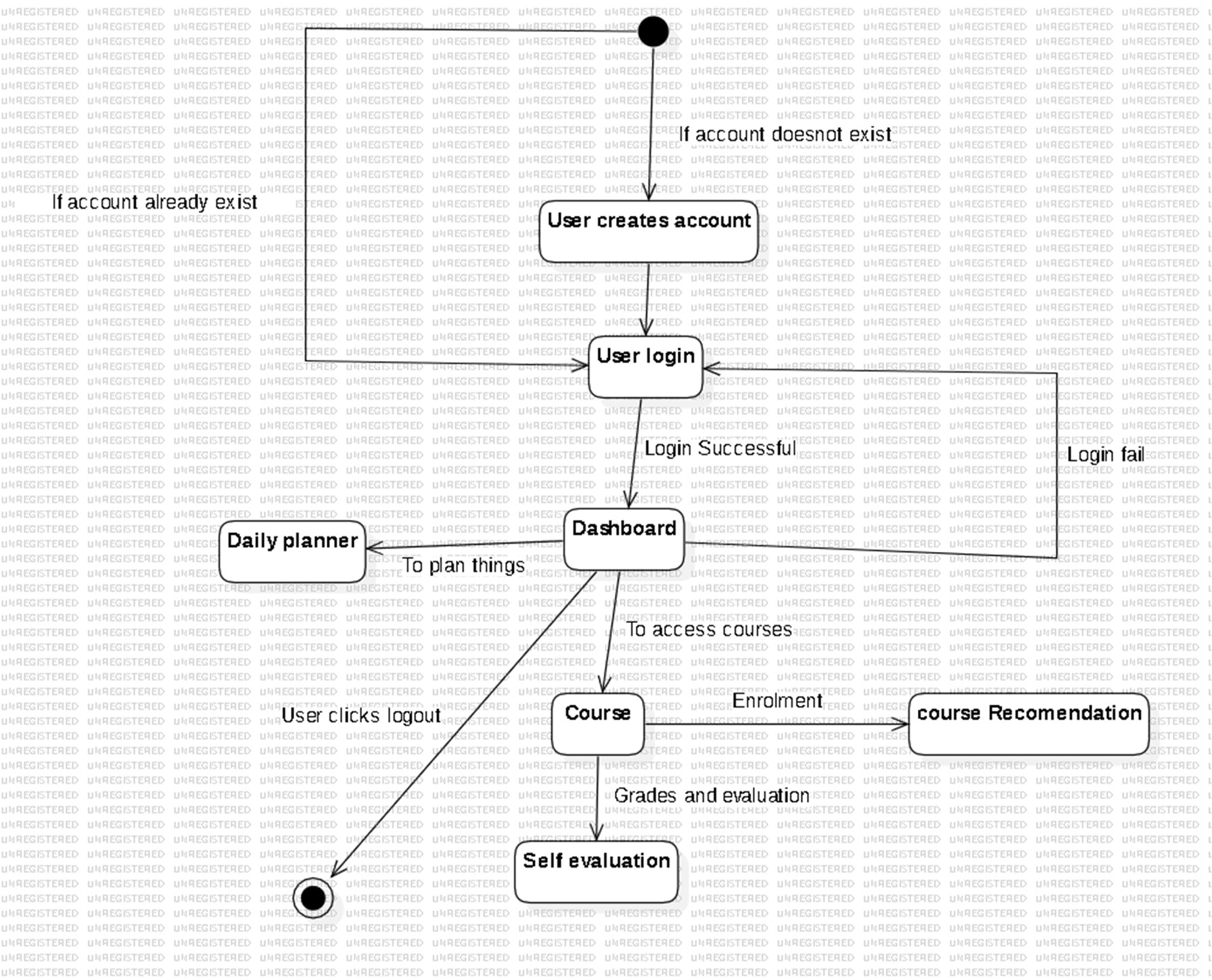
Course Recommendation: The Course Recommendation component works with the User component to collect user preferences, academic history, and career objectives. It may integrate with other educational platforms or databases to provide access to course catalogs, curriculum information, and user reviews. The Course Recommendation component generates individualized course suggestions based on user inputs and system data.

Progress Tracking: The Progress Tracking component communicates with the User component to retrieve user academic data such as grades, assignments, and exam results. It may work with learning management systems or educational databases to access and update user progress data. The Progress Tracking component offers visualizations, statistics, and reports to assist users track their academic achievement and discover areas for growth.

Self-evaluation Mechanism: The Self-evaluation Mechanism component works with the User component to collect user feedback on self-assessment tests, quizzes, and reflective exercises. It may be combined with external assessment tools or services to give a thorough assessment of user abilities and knowledge. The Self-Evaluation Mechanism component generates comments, recommendations, or learning materials based on the outcomes of the user's self-evaluation.

Overall, the Student Assist App interacts with other elements of the broader system through data exchange, integration with external systems or databases, and communication with various components to provide a seamless user experience and deliver the desired functionalities.

# Section 3: State-transition Diagram



**Section 4: Pseudo-code**

# Algorithm for User Profile Setup

function createProfile(username, email, password, major): user = new UserProfile()

user.setUsername(username) user.setEmail(email) user.setPassword(password) user.setMajor(major)

return user

# Example Usage

newUser = createProfile("JohnDoe", "[john.doe@email.com](mailto:john.doe@email.com)", "password123", "Computer Science")

# User Authentication

function login(username, password): if userExists(username):

if isValidPassword(username, password): return "Login successful"

else:

return "Invalid password"

else:

return "User not found" function userExists(username):

# Check if the user exists in the database

# Return true if user exists, false otherwise function isValidPassword(username, password):

# Check if the provided password matches the stored password for the user # Return true if the password is valid, false otherwise

# Algorithm for Daily Planner

function createTask(description, deadline, priority): task = new Task() task.setDescription(description) task.setDeadline(deadline) task.setPriority(priority)

return task

function addTaskToPlanner(userProfile, task): userProfile.getPlanner().addTask(task)

# Example Usage

newTask = createTask("Complete Assignment", "2023-12-01", "High") addTaskToPlanner(currentUser, newTask)

# Algorithm for Course Recommendation

function recommendCourses(userProfile, currentCourses): recommendedCourses = new

CourseRecommendationEngine().getRecommendations(userProfile, currentCourses) return recommendedCourses

# Example Usage

currentCourses = currentUser.getCurrentCourses()

recommendedCourses = recommendCourses(currentUser, currentCourses)

# Algorithm for Progress Tracking

function trackProgress(userProfile, course, completedUnits): userProfile.getProgressTracker().updateProgress(course, completedUnits)

# Example Usage

selectedCourse = "Introduction to Programming" completedUnits = 5

trackProgress(currentUser, selectedCourse, completedUnits)

# Algorithm for Self Evaluation

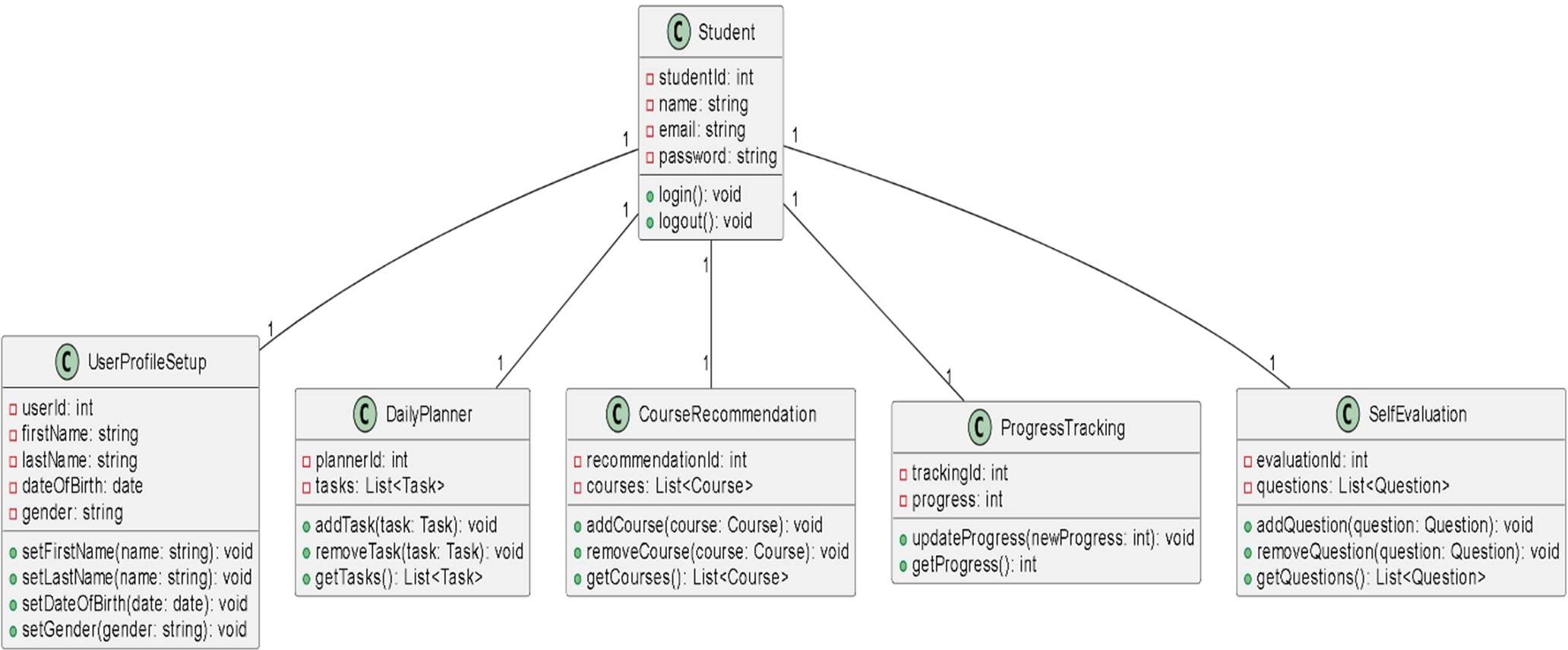
function evaluateSelf(userProfile, feedback, rating): selfEvaluation = new SelfEvaluation() selfEvaluation.setFeedback(feedback) selfEvaluation.setRating(rating) userProfile.setSelfEvaluation(selfEvaluation)

# Example Usage

feedbackText = "I need to improve time management." ratingValue = 3

evaluateSelf(currentUser, feedbackText, ratingValue)

# Section-5: Object models to explain the app's information architecture.



**Section 6: The design approach used to satisfy each non-functional requirement**

## Usability:

## Use clear and succinct language in the app's text and directions. Avoid employing jargon or technical terminology that your audience may not comprehend.

## Provide useful feedback and error notifications to users. Tell users what they're doing incorrectly and how to solve it.

## Test the app with users to obtain feedback and make any necessary modifications. This will assist guarantee that the software is simple to use for all users.

## Portability:

## Create the application using a cross-platform development framework. This allows you to build the app for many platforms using a single codebase.

## Use cloud-based services to store and deliver content. This increases the app's portability and accessibility to users from anywhere.

## Scalability:

## Implement a cloud-based architecture. This allows you to scale your infrastructure up or down based on the app's demand.

## Implement cache and load balancing. This will increase the app's performance and scalability. Choose a scalable database solution. Choose a database that can handle a high volume of concurrent users and transactions.

## Security:

Implement role-based access control (RBAC). This will allow you to restrict access to certain features of the app based on the user's role.

Use a secure communication protocol. Use HTTPS to encrypt all communication between the app and its servers.

Regularly update the app with security patches. This will help to protect the app from known vulnerabilities.

## Evolvability:

Implement a modular architecture. This will make it easier to introduce new features and resolve bugs in the future.

Implement version control and automated testing. This will allow you to better manage changes to the app's code and guarantee that new features do not introduce new bugs.

Release app updates regularly. This allows you to add new features and fix bugs rapidly.

# Section 7: The risk assessment and the exception management mechanisms.

## Risk Assessment:

**User Profile Setup**

Users may not set up their profiles completely or accurately, or they may not update their profiles regularly. The app will require users to complete certain profile fields before using the app, and it will send reminders to users to update their profiles regularly.

## Daily Planner

The auto-scheduling feature may not work correctly, or users may not be able to follow the scheduled plan. The app will allow users to customize the auto- scheduling feature and to manually override the schedule if needed. The app will also provide users with reminders about upcoming tasks and deadlines.

## Self-Evaluation Mechanism

The decision audit feature may not be accurate, or users may not find it to be helpful. The app will allow users to customize the decision audit feature to meet their individual needs. The app will also provide users with actionable insights from their self-evaluation data.

## Course Recommendation

The analysis-driven suggestions may not be accurate, or users may not be interested in the recommended courses. The app should will users to customize the course recommendation algorithm and to filter the results based on their interests and goals. The app will also provide users with information about the recommended courses, such as course descriptions, prerequisites, and reviews from other students.

**Progress Tracking :**

The visual dashboards and milestone markers may not be accurate, or users may not find them to be helpful. The app will allow users to customize the visual dashboards and milestone markers to meet their individual needs. The app will also provide users with actionable insights from their progress data.

## Exception1: A user falls behind in their studies.

Detection: The app could detect this by tracking the user's progress towards their goals and deadlines.

Handling: The app could send reminders to the user about upcoming tasks and deadlines. The app could also provide the user with additional support, such as study tips and resources.

## Exception2: A user is not interested in the recommended courses.

Detection: The app could detect this by tracking the user's engagement with the course recommendation feature.

Handling: The app could recommend courses that are more relevant to the user's interests and goals

## Section 8: H -C-I and C-C-I Protocols

Human-Computer Interaction (H-C-I) Protocols:

1. User Input:

Tokenization and sentiment analysis for understanding the false queries of the user. Personalize your services with smart dynamically generated responses.

1. Feedback Mechanism:

Provide instantaneous feedback through a user’s activities or individual preferences. Provide meaningful recommendations based on AI- based data.

Computer-Computer Interaction (C-C-I) Protocols:

1. Algorithmic Interaction:

Allow various programs for collaboration in which they learn from each other and evolve together. Use version management and update policies. Allow various programs for collaboration in which they learn from each other and evolve together. Use version management and update policies.

1. Data Security:

Ensure that use of end-to-end encryption and 2 factor authentication is incorporated to guard data security. Undertake security audit at regular intervals to enforce strict security standards.

# Section 9: Changes and Additions from Deliverable 1

1. Extended Feature Set:

Extending the project with widgets, one of the ideas is to be incorporated the dark mode function that is helpful during night studying session.

1. Refined User Interface

A user on-boarding experience that is well planned.

# Section 10: Remaining Work for the Final Prototype

1. Testing and Iteration:
   * Plan: Perform thorough alpha tests and beta tests on wide base of participants to collect the feedback and user experience.
   * Actions: Go through it with users and refine on the issues of usability, security, and finally general performance.
2. Scalability Planning:
   * Plan: Consider improvements that can be made to current architecture for better scaling and optimization.
   * Actions: Introduce auto-scaling techniques and optimize database query for higher number of users without inconsistencies.
3. Documentation:
   * Plan: Develop thorough user manuals and API documentations.
   * Actions: Provide video tutorials for end users, as well as developers, explaining exactly what their app is about and how they can integrate it into their system.
4. Integration with Academic Systems:
   * Plan: Standardize a common process of integration in collaboration with some educational institutions.
   * Actions: Check the functionality, Compatibility with pre-existing platforms such as Blackboard, Moodle, D2L, Sakai, etc.
5. Optimization for Performance:
   * Plan: Carry out a performance audit on identified bottlenecks. This detailed information outlines the changes, additions, and remaining work needed for the final prototype of the "Student Assist" app, considering the problem statement and requirements provided earlier.