**Analysis and Test Planning Report**

**Index**

**Section 1:** Summary

**Section 2:** System/Context Model

**Section 3:** State Transition Diagrams

**Section 4:** Pseudo Code

**Section 5:** Information Architecture

**Section 6:** Non-Functional Requirements

**Section 7:** Risk Assessment

**Section 8:** H-C-I protocols and C-C-I protocols

**Section 9:** Changes & Additions to the Deliverable 2

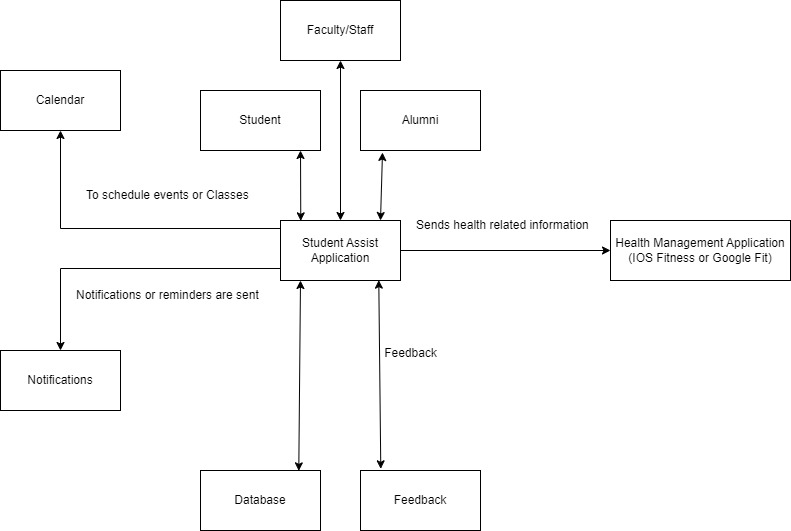
**Section 1: Proposed Design Summary:**  
  
Our team methodically designed the "Student Assist" application with a user-centric approach, keeping IIT students' and other users' requirements and concerns in mind. We developed the application interface to be straightforward, with an emphasis on establishing a seamless user experience, making it easy for students to easily manage their academic obligations, schedules, and health-related activities.

Our program strives to enable students to take proactive actions toward academic achievement by providing a complete range of features like assignment tracking, timely deadline reminders, and easy access to crucial course resources. We've also included health management tools that gently encourage users to take breaks, establish good study habits, and access resources that promote physical and mental well-being.  
   
Our staff recognizes the significance of balancing academic excellence and personal well-being. With this in mind, we have worked hard to establish an environment that encourages students to emphasize their entire health and personal growth as well as academic accomplishment.

We diligently created the user interface to guarantee seamless navigation, clear information display, and an appealing visual design, consequently increasing user engagement and productivity. Users may easily access campus resources, remain updated about forthcoming events and activities, and efficiently manage their academic duties by using the "Student Assist" program.

We will continue to improve the application based on user input and developing requirements, with a strong commitment to establishing an inclusive and friendly community that enhances academic achievement and overall well-being.

**Section 2: System/Context Model and Interaction Description:**  
  
The "Student Assist" application seamlessly integrates with several key components within the broader system, including the database, notification services, health management, and feedback mechanism. Our application interacts securely with these elements, ensuring robust data privacy, seamless information dissemination, and efficient communication channels. The interconnectedness of these components facilitates a cohesive user experience, enabling smooth data exchange and streamlined functionality within the application.



**Interactions of the Student Assist Application with the following are described.**

**Database:** The database, a critical component of our service, saves and manages critical user information, allowing students to establish and maintain individualized profiles with their academic records, preferences, and hobbies. We have put in place strong data security procedures to protect user privacy and the confidentiality of sensitive information, creating trust and confidence among our users.

**Feedback Mechanism:** Our application has a strong feedback mechanism that encourages user involvement and participation to create ongoing improvement. This component provides an important route for users to share their ideas, comments, and concerns, allowing us to improve the program and better fulfill the needs of our user community.  
   
**Calendar:** This is used to access the student's calendar and set the class and event schedule.

**Notification Services:** The notification services in our program are critical in informing users about upcoming deadlines, campus activities, and other notifications. We've created an effective notification system that sends out timely notifications and updates, ensuring that students keep organized and informed without getting overcome by an overload of information.  
   
**Health Management Application:** Third-party programs, such as IOS Fitness and Google Fit, are utilized to collect the data and gives health suggestions accordingly.

**Section 3:**

Login Screens:

A screenshot of a login form

Description automatically generatedA screenshot of a login screen

Description automatically generated

My Tasks:  
  
A screenshot of a phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Courses selected by student and related tasks and score logs:

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

User Dashboard:

A screenshot of a computer

Description automatically generatedA screenshot of a course report

Description automatically generated

**Score Logs:**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Application Menu:

A screenshot of a cell phone

Description automatically generated

User’s Calendar integrated with our application in Agenda section:

A screenshot of a cell phone

Description automatically generated

Collaboration with other users in same group using email, chat or chatter:

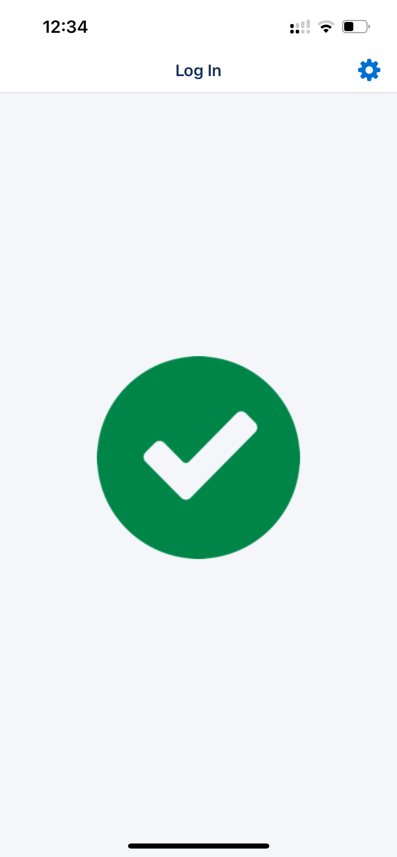
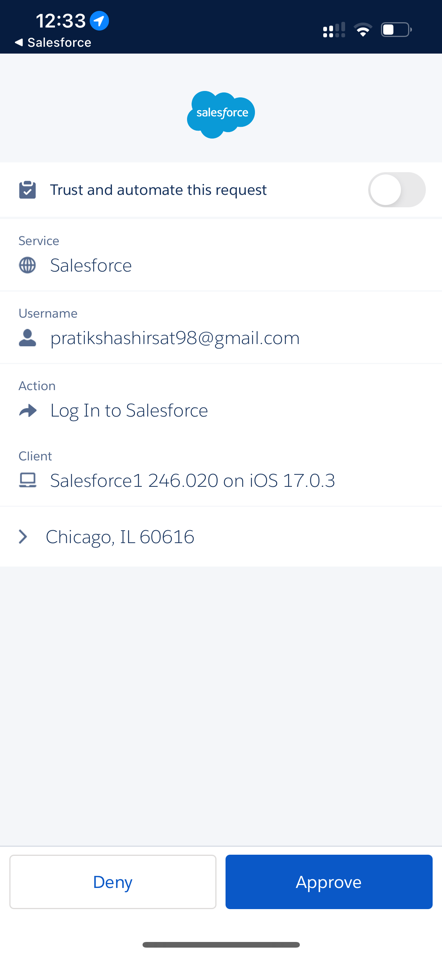
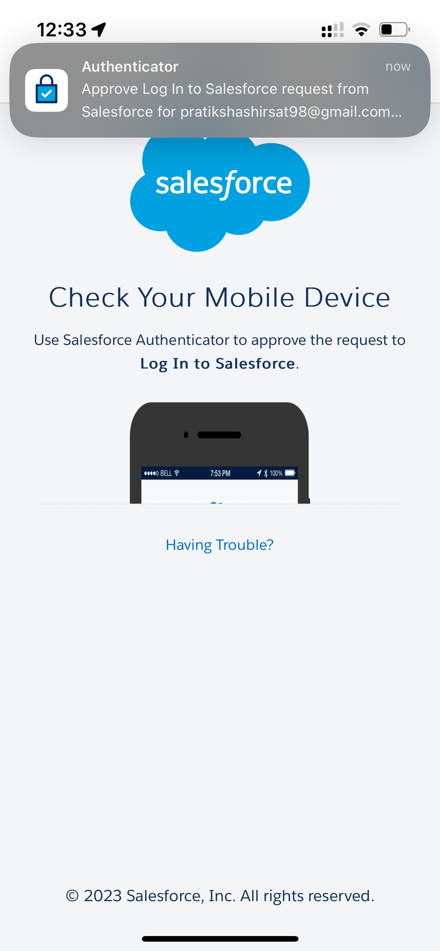
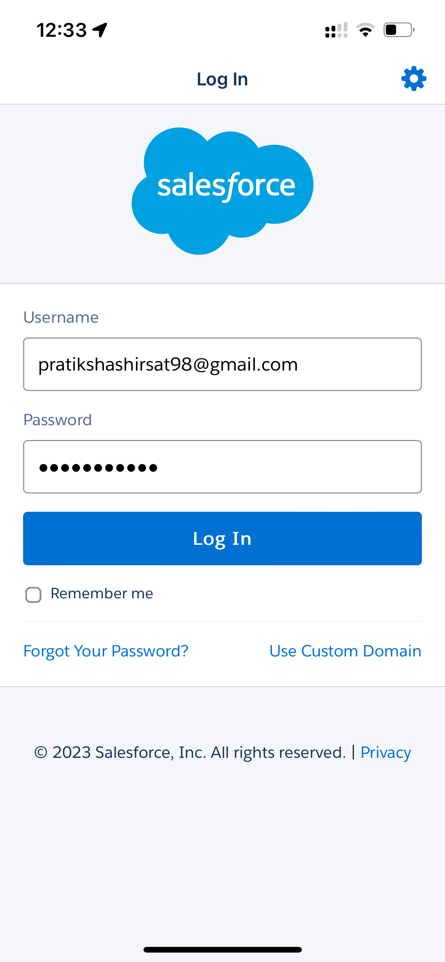
A screenshot of a list

Description automatically generatedA screenshot of a phone

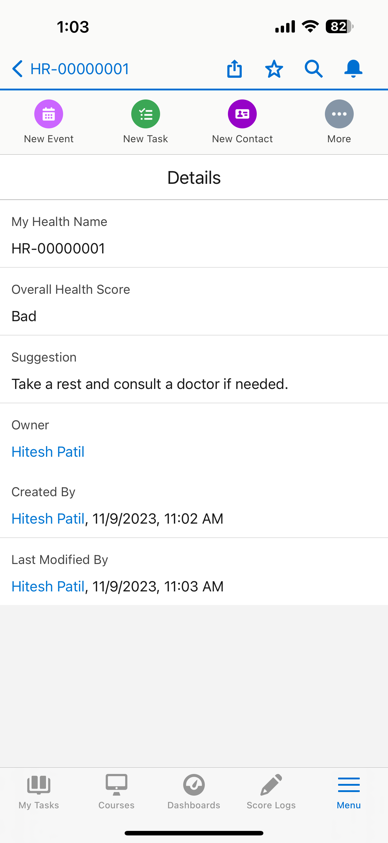
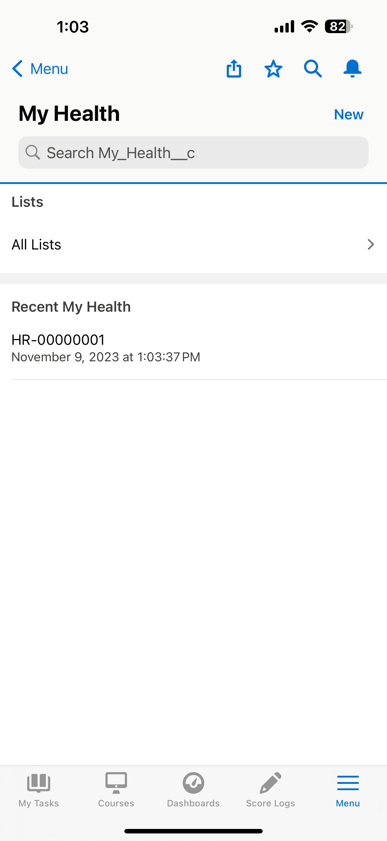
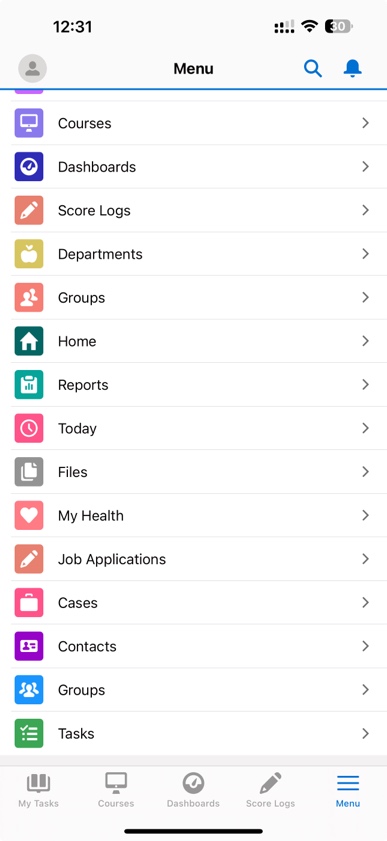
Description automatically generated

A screenshot of a phone

Description automatically generated



**My Health Section:**



**Flow Chart:**

**A diagram of a company

Description automatically generated**

**Section 4:**

**1. Login Functionality:**

public void login(Credentials credentials) {

String identifier;

// Identify the user based on credentials

if (credentials.hawkId != null) {

identifier = credentials.hawkId;

} else if (credentials.email != null) {

identifier = credentials.email;

} else {

System.debug('Error: Please check your username and password. If you still can’t login, contact your Salesforce Administrator.');

return;

}

// Authenticate the user

if (authenticate(identifier)) {

System.debug('Welcome!');

} else {

System.debug('Please check your username and password. If you still can’t login, contact your Salesforce Administrator');

}

}

private Boolean authenticate(String identifier) {

// Implement authentication logic

// Return true if authentication is successful, false otherwise

return true; // Placeholder, replace with actual logic

}

**2. Task management functionality:**

public class MyTasksController {

// Method to create a new My Task

public static void createMyTask(String taskName, String priority, String relatedCourse, Date dueDate, String owner) {

My\_Task\_\_c newTask = new My\_Task\_\_c();

newTask.My\_Task\_Name\_\_c = taskName;

newTask.Priority\_\_c = priority;

newTask.Related\_Course\_\_c = relatedCourse;

newTask.Due\_Date\_\_c = dueDate;

newTask.Status\_\_c = 'Not started'; // Default status

newTask.Owner\_\_c = owner;

insert newTask;

}

// Method to update the status of a My Task

public static void updateTaskStatus(String taskId, String newStatus) {

My\_Task\_\_c task = [SELECT Id, Status\_\_c FROM My\_Task\_\_c WHERE Id = :taskId LIMIT 1];

// Check if the task is not already completed

if (task.Status\_\_c != 'Completed') {

task.Status\_\_c = newStatus;

// If the new status is "Completed", lock the task

if (newStatus == 'Completed') {

task.Locked\_\_c = true;

}

update task;

} else {

// Task is already completed, handle accordingly (e.g., throw an exception or log a message)

}

}

}

**3. Score Logging Functionality:**

public class CourseController {

// Method to create a new Course record

public static void createCourse(String courseCode, String courseName, String section) {

Course\_\_c newCourse = new Course\_\_c();

newCourse.Course\_Code\_\_c = courseCode;

newCourse.Course\_Name\_\_c = courseName;

newCourse.Section\_\_c = section;

insert newCourse;

}

// Method to create a new Score Log record for a specific Course

public static void createScoreLog(String scoreLogName, String courseName, Decimal totalMarks, Decimal obtainedMarks) {

// Query the Course record based on the Course Name

Course\_\_c course = [SELECT Id FROM Course\_\_c WHERE Course\_Name\_\_c = :courseName LIMIT 1];

// Create a new Score Log record

Score\_Log\_\_c newScoreLog = new Score\_Log\_\_c();

newScoreLog.Score\_Log\_Name\_\_c = scoreLogName;

newScoreLog.Course\_\_c = course.Id;

newScoreLog.Total\_Marks\_\_c = totalMarks;

newScoreLog.Obtained\_Marks\_\_c = obtainedMarks;

// Calculate and set the Your\_Score\_\_c field (percentage)

if (totalMarks != 0) {

newScoreLog.Your\_Score\_\_c = (obtainedMarks / totalMarks) \* 100;

}

insert newScoreLog;

}

}

**4. My Course Dashboard functionality:**

public class ReportController {

// Method to generate and display the report

public static void generateScoreReport() {

// Query the necessary fields from Course and Score Log objects

List<Course\_\_c> coursesWithScores = [SELECT Course\_Code\_\_c, Course\_Name\_\_c,

(SELECT Score\_Log\_Name\_\_c, Your\_Score\_\_c FROM Score\_Logs\_\_r)

FROM Course\_\_c];

// Iterate through the results and display the report

for (Course\_\_c course : coursesWithScores) {

System.debug('Course Code: ' + course.Course\_Code\_\_c);

System.debug('Course Name: ' + course.Course\_Name\_\_c);

// Iterate through Score Logs related to the course

for (Score\_Log\_\_c scoreLog : course.Score\_Logs\_\_r) {

System.debug('Score Log Name: ' + scoreLog.Score\_Log\_Name\_\_c);

System.debug('Your Score: ' + scoreLog.Your\_Score\_\_c);

}

System.debug('------------------------');

}

}

}

**5. Student health monitor system functionality:**

public class HealthDataProcessor {

// Salesforce REST API endpoint for receiving health data and sending recommendations

public static String REST\_ENDPOINT = 'https://your-salesforce-instance/services/apexrest/healthdata';

// Method to process health data and send recommendations

public static void processHealthData(String studentId, Double heartRate) {

// Implement logic to analyze health data and generate recommendations

String recommendation = generateRecommendation(heartRate);

// Send recommendation to Salesforce using REST API

sendRecommendationToSalesforce(studentId, recommendation);

}

public static String generateRecommendation(Double heartRate) {

// Implement logic to analyze heart rate and generate a recommendation

// For example, recommend rest if heart rate is too high

if (heartRate > 100) {

return 'Take a rest and consult a doctor if needed.';

} else {

return 'Your health is good. Keep it up!';

}

}

public static void sendRecommendationToSalesforce(String studentId, String recommendation) {

// Implement logic to send recommendation to Salesforce using REST API

HttpRequest request = new HttpRequest();

request.setEndpoint(REST\_ENDPOINT);

request.setMethod('POST');

request.setHeader('Content-Type', 'application/json');

request.setBody('{"studentId": "' + studentId + '", "recommendation": "' + recommendation + '"}');

Http http = new Http();

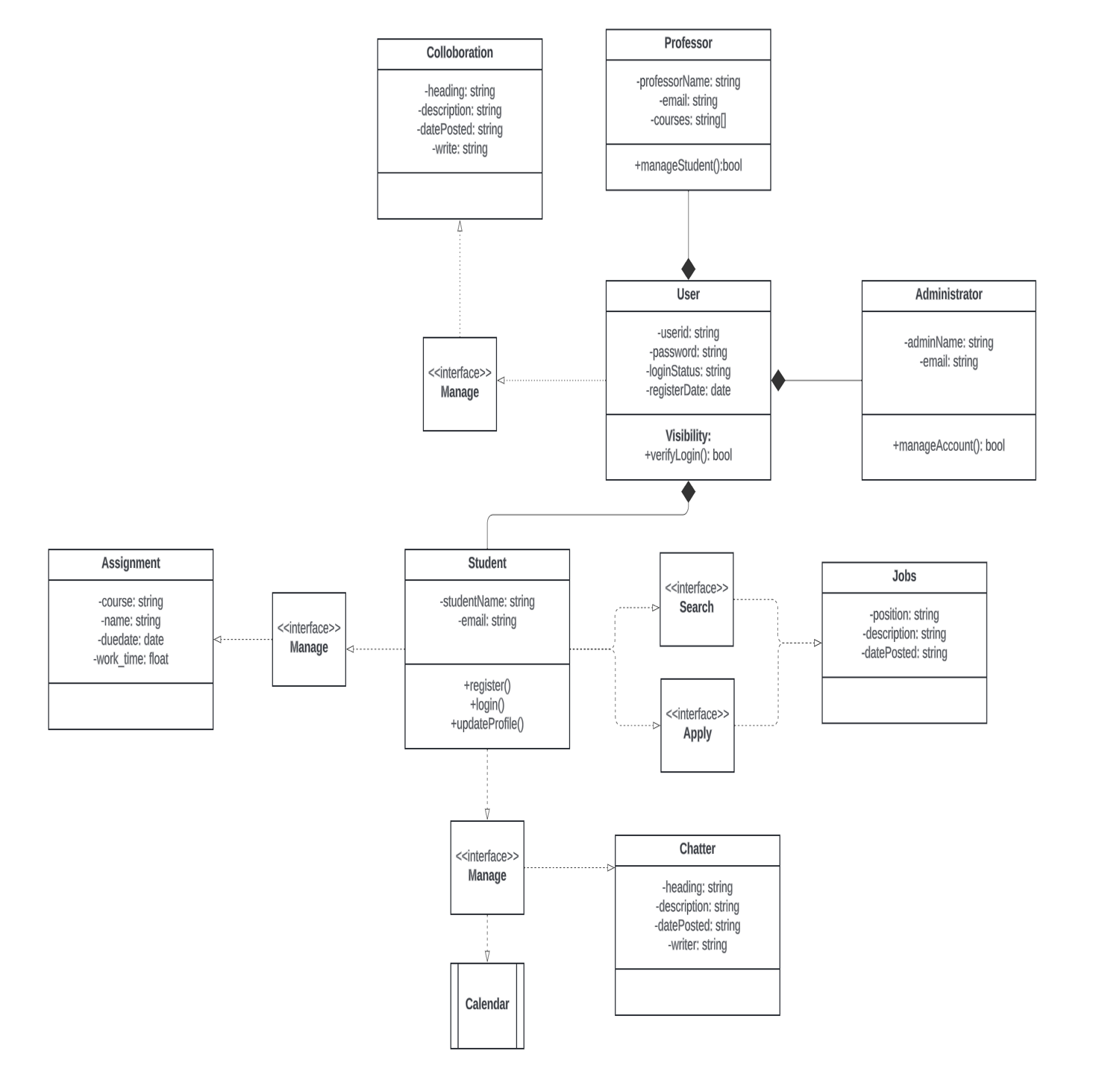
HttpResponse response = http.send(request);

// Handle the response as needed

}}

**Section 5: Information Architecture**

This section presents a UML Diagram illustrating the data perspective within the information architecture of the Student Assist App.



**Section 6: Non-Functional Requirements**

This section will describe the design approach used to satisfy each non-functional requirement.

**Performance:**

Ensuring the app runs smoothly and responds fast is crucial. We aim to make it perform well, even when many people are using it. Our target is to manage up to 20,000 users (about the seating capacity of Madison Square Garden) every hour. When you start the app, the main page with all the information and pictures should appear in less than 4 seconds. Plus, when you tap on something, the app should react in about 1 second. If we meet these goals, it'll make using the app feel smooth and speedy, just the way we want it to be.

**Scalability:**

Scalability is about figuring out how much our system can handle and still do an excellent job. In our app, we've made sure that even when 1000 people are using it all at once, it stays smooth and works at its best.

**Security:**

Security is a set of guidelines that protect the app's information from harmful software and unauthorized users. By utilizing Google Authentication, the app prevents access from those who should not be there, ensuring overall safety. Users are limited to viewing only their personal information to maintain privacy. The system adheres to security rules to ensure everything remains secure and meets the required standards.

**Availability:**

Availability is about how likely it is for users to use the system when they want to. It is like a percentage that tells us how often the system is up and running or how many times things work successfully. For example, the aim for the app is to be working and ready for users 98% of the time every month. Keeping the system available is crucial for the business.

**Reliability**:

The reliability of a system or component indicates the likelihood that it will function properly for a predetermined amount of time under specific circumstances. This probability is typically stated as a percentage. The application is designed to have a threshold of 90% reliability for every month. This demonstrates that the system must perform without failure for 90% of its usage time during the month.

**Maintainability:**

The applications should be updated very frequently, and all the bugs and exceptions are continuously updated. And the latest releases must also support new features and functionalities. While under maintenance, the application should be restored very quickly within 24 hours for the user.

**Section 7:**

Conducting a thorough risk assessment on the student assistance app is crucial. It will help us identify potential flaws in the design or development system and determine where we are most vulnerable from a security perspective. A comprehensive risk assessment is essential to ensure the safety and security of the app users' data.

**Identification of potential failures:**

Losing user data can have severe consequences as it can lead to user dissatisfaction, which in turn can result in reputational damage, loss of revenue, and legal implications. Therefore, it is crucial to consider the potential risks associated with data loss and take all necessary measures to prevent it.

Another issue that needs to be considered is user engagement. If the user of our app has difficulty understanding its features, it can lead to problems. To prevent this, it's essential to develop the user interface of our app in a much more interactive and easy-to-understand way.

In addition, there may be performance issues with the app due to high user traffic, leading to crashes. To prevent such failures, it's crucial to ensure that our systems are scalable. Another issue that may arise is when new code is integrated with the existing codebase during the deployment of a new feature in the production environment. This integration can cause problems, which can be avoided by conducting thorough testing prior to deployment in the production environment.

**Assessment of Risk Exposure:**

One of the highest risks in software development is the loss of user data, which can result in severe consequences such as user disappointment and legal action. To mitigate this risk, it is crucial to have a contingency plan in place beforehand. In addition, performance issues and integration issues are also significant categories of risk. Performance issues can impact user experience by reducing app usage due to slow response times, while integration issues can impede the functionality of certain features. Finally, issues with the user interface, while relatively low risk in terms of legal consequences, can still have a significant impact on user engagement and adoption. A poor UI can make it difficult for users to understand the features and functionality of the app, which may result in lower user counts.

**Exception Management:**

**Detection and Handling Mechanism:**

To ensure the seamless functioning of an application, it is essential to have detection and handling mechanisms in place for various types of issues. For instance, if a security issue arises, it is imperative to detect the problem and immediately notify the app management team. The team should then inform the users of the issue and work on regaining their data. Additionally, they should delete any data that may have been uploaded on dark web platforms.

In the case of performance issues, continuous monitoring is necessary to detect any potential problems. If any issues are detected, the system should be immediately scaled, and the developed code should be optimized to handle the performance issues effectively. In order to detect issues that may arise due to integration activities, it is important to have an error-tracking mechanism in place. Automated rollback of newly deployed features is the preferred mode of choice to handle the issues. When it comes to user adoption issues, user feedback plays a critical role in identifying them. In order to handle this better, UI development should be focused on making it understandable to all sections of society, and continuous communication with users on how to use our developed features is the best way to address the issue.

**Risk Mitigation Activities:**

To mitigate risks, it's important to perform continuous and thorough security audits to identify potential vulnerabilities. Our technology should be scalable enough to meet the needs of all users and handle high traffic without any issues. All new functionalities should be tested rigorously before deployment to reduce integration risks. Regular updates should be released to patch any loopholes and cater to the needs of different user groups.

**Section 8: HCI and CCI protocols:**

**H-C-I Human-Computer Interaction Protocols:**

HCI protocols facilitate communication between users and student assist apps, enhancing user experience.

The User Login/Registration Protocol enables quick and easy access to the app. The user can either use their existing credentials or create a new account. The program ensures secure identification of the user and uses encryption to protect their data. This protocol enhances the user experience by simplifying and securing the login process.

The User Profile Management Protocol allows users to access and modify their demographic and educational information, ensuring that data is current and secure. This feature supports efficient information management, user satisfaction, and security.

The Task Management Protocol empowers users to establish, modify, and remove tasks, incorporating priority rankings and alerts. This enhances efficiency and alleviates stress.

The Collaboration Protocol enables users to communicate, exchange messages, and share resources, promoting a culture of collaboration and knowledge transfer, while maintaining secure data exchanges.

Users can easily access educational materials, such as lecture recordings and articles through the Resource Access Protocol. This app makes sure that you get the latest and most relevant resources for a super-efficient and enjoyable learning experience.

**C-C-I Component-Component Interaction Protocol:**

This protocol outlines how the student assistance app components work together to enhance user experience.

For an educational application to function efficiently, its different components need to work together seamlessly. One of the most important modules is the task management system, which connects tasks to specific users by communicating with user profiles. Additionally, it works in tandem with the notification system to send timely reminders to users about upcoming tasks.

The resource access component enables users to efficiently navigate the database and access instructional materials. It collaborates with the user interface to ensure that resources are displayed clearly and are easily accessible.

The collaboration feature is an important part of the system that enables users to interact and establish relationships with peers. It works with user profiles to provide a seamless and secure messaging and peer-linking experience. Additionally, it interfaces with the security infrastructure to ensure the confidentiality and safety of all communications.

The user authentication system is a crucial component that verifies the identity of the user. It collaborates with the user profile management system to ensure that only authorized users can access the application's features. Furthermore, it interacts with the task management and collaboration components to restrict access to specific features as required.

The database module is a critical component of the application, interfacing with other modules to perform data operations, ensuring data uniformity, and preserving the overall integrity of the application.