Team Design Report

Group - V

Team Member Names
Mugdha Atul Kulakarni
Anirudha Kapileshwari
Vijay Krishna Raghav Kesanapalli
Naga Sai Shyam Suhas Komaravolu
Thirumalesh Kurukundha

Spring 2024 Software Engineering I (CS-487-02)
30 March 2024

Section 1

Summary

A thorough grasp of the different requirements and difficulties students encounter on a daily basis is necessary while developing a "Student Assist" app. With a full toolkit designed to improve learning, productivity, organisation, and well-being, this software seeks to maximise the student experience. An overview of the application's suggested design may be found below:

App Overview

"Student Assist" is going to be a web and mobile application that helps students at every stage of their education, from college to high school. It will combine a number of features to handle social, academic, and personal facets of student life, giving them a one-stop shop.

Core Features:

- A. Academic Organizer:
- **Timetable of Classes:** An easy-to-use calendar that allows students to enter the times, locations, and contact details of their professors.
- **Assignment Tracker**: An application with customisable reminders for keeping track of exam dates, project milestones, and assignment due dates.
- **Grade Tracker**: This tool enables students to measure their academic progress by recording and keeping track of their grades throughout the course of the semester.

B. Research Aids:

- **Flashcards**: An application that facilitates the creation, sharing, and use of flashcards for study and memorization.
- **Study Timer**: Uses programmable clocks or the Pomodoro Technique to assist efficiently organise study sessions.
- **Making Notes:** An adaptable note-taking portion arranged by subject or course that allows text, voice notes, and photos. C. Resource Hub:
- **Library Access**: Straightforward connections to databases, journals, and school library resources for scholarly purposes.
- **Tutor Connect**: A peer-to-peer tutoring platform where students can offer or receive subject-matter expertise.

D. Wellness Centre:

Mental Health services: Provides quick access to contacts and services, including those particular to the university, for mental health support.

Fitness Tracker: Works with well-known fitness applications to promote physical health and wellness, including alerts to continue exercising.

Budget Manager: An easy-to-use programme for goal-setting, tracking expenses, and managing funds.

3.Social Integration:

- Campus Events: This tool helps students get involved in their community by allowing them to learn about clubs, events, and other activities on campus.
- Study Groups: The ability to start and join study groups, which facilitates group project collaboration and exam preparation.

4.Customization and Accessibility:

Personalization : Users have the ability to alter the application to suit their own tastes in terms of theme, layout, and notification settings.functions that make the programme accessible to all users
Accessibility : text-to-speech, high contrast modes, and other functions.

5.Privacy and Technology:

Cloud Sync: All information will be safely synchronised between all devices, guaranteeing accessibility from any location while preserving data security and privacy.
Offline Mode: To accommodate locations with inadequate internet connectivity, core features will be accessible offline.

6.Development of the Prototype:

Figma will be used for interactive prototyping, and Sketch will be used for UI/UX design. Since these features make up the majority of the app's functionality, the first phase will concentrate on the study tools and academic organiser features. A small group of students will be given access to a beta version for testing and suggestions, as user feedback will be very important.

7.Future Expansion:

AI-powered study assistants, integration with academic institutions for direct communication and resource sharing, and more sophisticated wellness tracking to provide all-encompassing support for students are some examples of future improvements.

Section 2

System/Context Model

It is important to consider how the "Student Assist" app will interface with different internal and external components when designing it within a larger educational environment. To do this, a system/ context model outlining these interactions and the information flow must be created. We go over the model and how it interacts below.

Overview of the System/Context Model: The "Student Assist" app's system/context model revolves around the app's internal components, third-party service providers, educational institutions, and students. The approach is made to guarantee secure and private communication while promoting interoperability.

External Interactions:

1.Academic Establishments:

Data exchange: Safe APIs that allow the app to directly exchange data with the user, such as class schedules, grades, and activities happening on campus.
Authentication: Integration for access control and verification with institutional login systems.
Resource Access: Straightforward connections to course materials, institutional announcements, and library databases.

2.Providers of Third-Party Services:

Physical fitness App Integration: To import activity data into the wellness centre, connect to fitness applications (like Fitbit and MyFitnessPal).
Financial Tools: Real-time financial tracking is provided by integrating the budget manager tool with banking APIs or budgeting apps.
Cloud Services: For data syncing, backups, and cross-device availability, make use of cloud computing and storage services.

3.Students (Users):

Device Integration: The application ought to be able to operate in unison with the user's whole device ecology, which encompasses computers, tablets, and smartphones.
Social Networks: Facilitate the exchange of knowledge, educational materials, or event specifics via widely used social media channels.

Internal Components:

1.user interface (UI):

Customisation and Accessibility: UI components that adapt to accommodate personalisation options and accessibility features.

2.Backend Services:

Schedules, notes, academic records, and user data are handled, stored, and processed securely through data management.Notification System: Personalised alerts for tasks, occasions, and individual notes.

3.Analytics Engine:

Usage Analytics: Gather de-identified information about app usage to enhance functionality and user experience.

Performance tracking: Resources to monitor students' long-term development on both the academic and personal fronts.

Interaction Flow :

1.With Academic Institutions: Academic institutions grant "Student Assist" access to their APIs so that it can retrieve and update academic data.

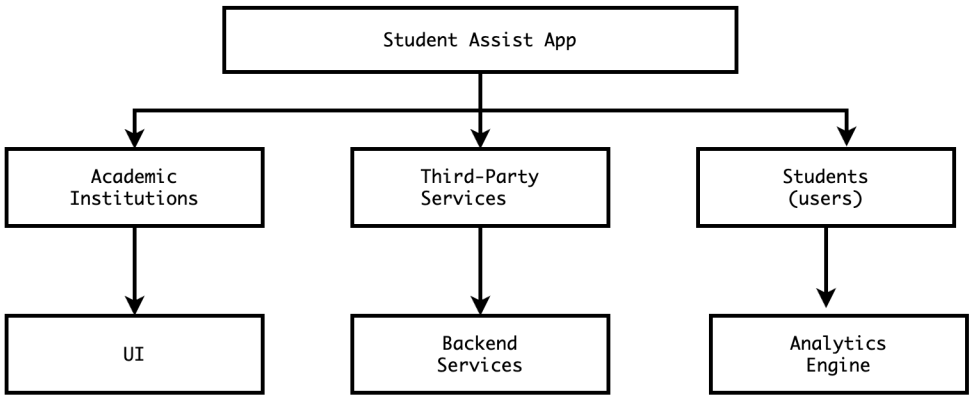
For up-to-date information on events and class schedules in real time, the app links with the school calendar.

2.With Third-Party Service Providers: "Student Assist" links to third-party APIs with the authorization of the user in order to retrieve pertinent data (e.g., daily steps from a fitness app).

The software uses secure APIs to retrieve financial tools in order to deliver information about budgeting.

3.With Students: Students use the app to directly enter their schedule, homework, and personal objectives. After processing this data, the app offers wellness advice, study recommendations, and personalised notifications. Through the app, users can choose to participate in study groups or share their study materials, promoting a collaborative learning environment.

Internal Components: To improve the user experience, the UI adapts to user choices and accessibility settings. Backend services handle user authentication and device syncing to guarantee data security and integrity. Over time, the analytics engine enhances the effectiveness and personalisation of the app by using user interaction data to provide insights and recommendations.



Section 3

UI sketches and Depictions of User Navigation

3.1 User Interface (UI) Sketches

Registration Page: A simple form with fields for username, password, and email, along with a submit button.

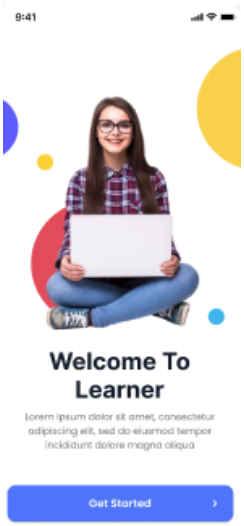
Personalized Calendar: A calendar view showing the user's classes and deadlines, with options to add or modify events.

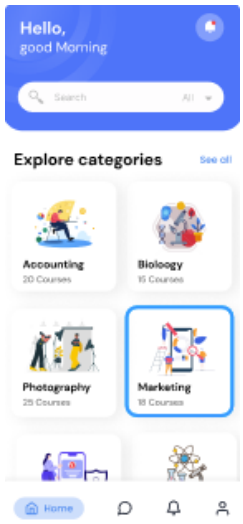
Goal Adjustment Mechanism:An interface allowing users to set, view, and modify their academic goals.

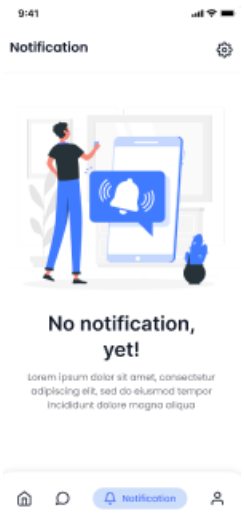
Direct Messaging:A chat interface enabling users to send and receive messages with other users.

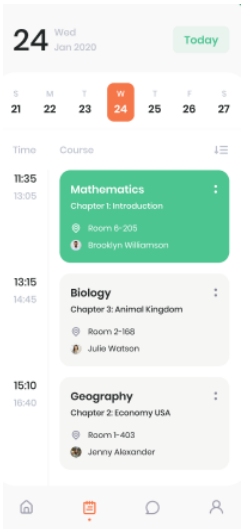
Discussion Board: A forum-like interface where users can post topics and reply to others' posts.

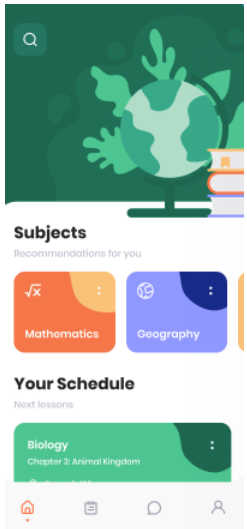
UI References are as Follows







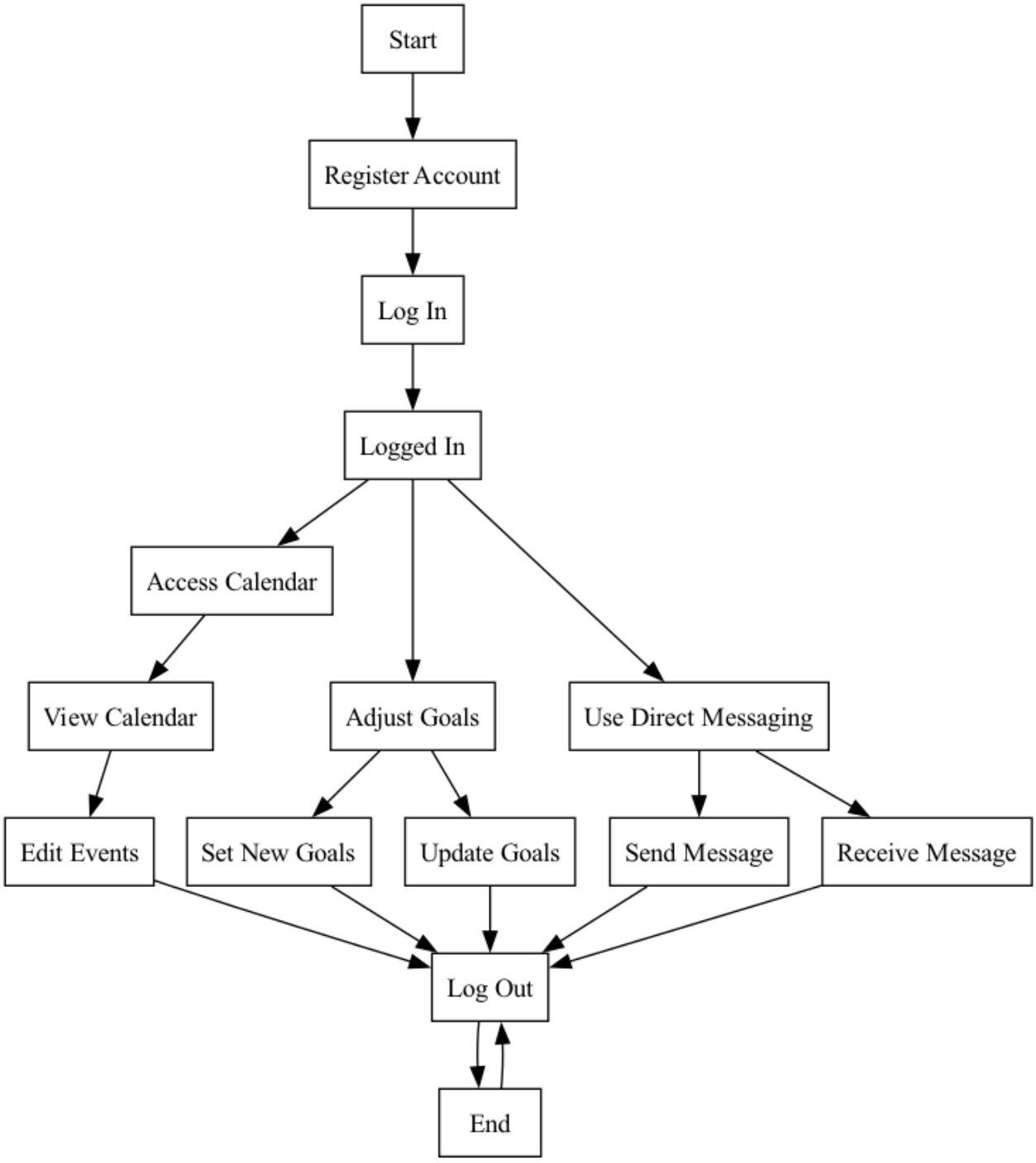


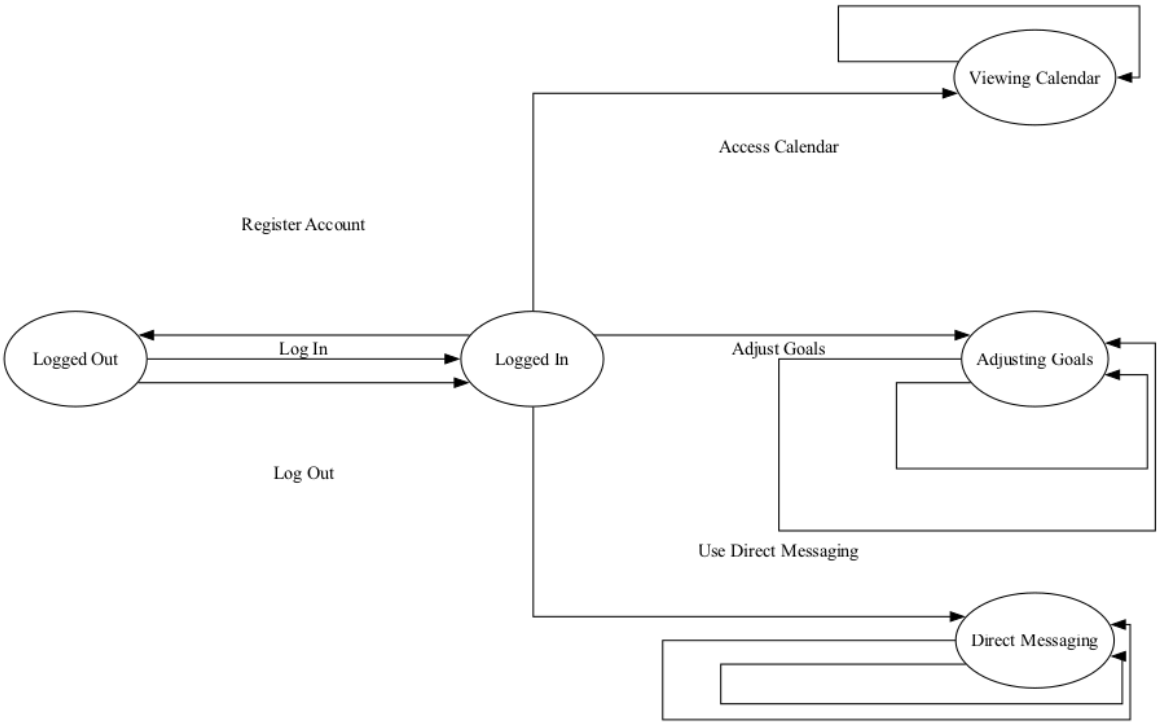


3.2 User Navigation

Workflow Diagram: A flowchart depicting the steps a user takes to register, log in, access their personalized calendar, adjust goals, and use direct messaging.

State-Transition Diagram: A diagram showing the various states of the application (e.g., logged out, logged in, viewing calendar, adjusting goals) and the transitions between these states based on user actions.





Section 4

Algorithmic Perspective with Pseudocode

In this section, we'll delve into the algorithmic perspective of the key functionalities of the application, using pseudocode to illustrate the logic and processes involved

User Registration

```
function registerUser(username, password, email):
    if not isValidUsername(username):
        return "Invalid username"
    if not isValidPassword(password):
        return "Invalid password"
    if not isValidEmail(email):
        return "Invalid email"
    if userExists(username):
        return "Username already exists"
    createUser(username, password, email)
    return "User registered successfully"
```

Log In

```
function logIn(username, password):
    if not userExists(username):
        return "User not found"
    if not isCorrectPassword(username, password):
        return "Incorrect password"
    setUserLoggedIn(username)
    return "User logged in successfully"
```

Access Personalized Calendar

```
function accessCalendar(userID):
    calendar = getCalendarForUser(userID)
    if calendar is empty:
        return "No events in calendar"
    return calendar
```

Adjust Goals

```
function adjustGoal(userID, goalID, newDetails):
    if not goalExists(userID, goalID):
        return "Goal not found"
    updateGoal(userID, goalID, newDetails)
    return "Goal updated successfully"
```

Use Direct Messaging

```
function sendMessage(senderID, receiverID, message):
    if not userExists(receiverID):
        return "Recipient not found"
    createMessage(senderID, receiverID, message)
    return "Message sent"
```

Data Integration with University Systems

```
function syncUserData(userID):
    universityData = getUniversityDataForUser(userID)
    if universityData is not empty:
        updateUserData(userID, universityData)
```

```
        return "User data synced with university systems"
    return "No new data to sync"
```

View Discussion Board

```
function viewDiscussionBoard():
    posts = getAllPosts()
    if posts is empty:
        return "No posts found"
    return posts
```

Post to Discussion Board

```
function postToDiscussionBoard(userID, postContent):
    if not isValidPost(postContent):
        return "Invalid post content"
    createPost(userID, postContent)
    return "Post created successfully"
```

Submit Feedback

```
function submitFeedback(userID, feedbackContent):
    if not isValidFeedback(feedbackContent):
        return "Invalid feedback content"
    createFeedback(userID, feedbackContent)
    return "Feedback submitted successfully"
```

Section 5

Data Perspective

Based on a thorough data and object model, the EduConnect program makes use of a complex information architecture. " The architecture is intended to facilitate the main features of the application, such as goal modification, direct messaging, customized calendars, user registration, and system connection with the institution. A few main entities form the framework of the architecture:

User Profile: At the core of the app is the user profile, which holds data like login credentials, preferences, courses they are registered in, and educational objectives. Personalized calendars and goal monitoring are only two examples of how this architecture facilitates user experience customization.

Calendar Events: Associated with the User Profile, these items stand in for personal events, class schedules, and deadlines for coursework. The integrated planning tool is enabled by their dynamic generation, which is dependent on the user's inputs and courses.

Direct Messages: This entity supports real-time data exchange by facilitating user communication. Messages are safely transmitted and saved thanks to the messaging system's efficient and private design.

Goals and Progress Tracking: This model's objects let users establish, track, and modify their professional and personal objectives. The technology gives users insights and reminders while monitoring their progress toward these objectives.

Integration Points: EduConnect integrates with university systems to deliver precise and current information. This involves transferring grades, assignments, and course data to the user's EduConnect profile.

Through the data model's performance and scalability optimizations, the application can support many users at once without experiencing any speed or functionality issues. Sensitive user data and system integrity are safeguarded by security mechanisms that are integrated into the data architecture.

Section 6

Design Approach for Non-Functional Requirements

EduConnect was developed with the goal of meeting a wide range of non-functional requirements, guaranteeing the application's functionality as well as its dependability, usability, and security.

Security: The program uses secure hash techniques to store passwords, end-to-end encryption for messaging, and HTTPS for data transmission in order to safeguard user data and guarantee secure

communication. The security posture of the application is further strengthened by routine security audits and adherence to data protection laws (such as GDPR).

Performance: EduConnect is built with great performance in mind. To guarantee fast response times even during periods of high demand, it makes use of caching, load balancing, and database optimization techniques. Without sacrificing the user experience, the application's backend architecture expands dynamically to meet demand.

Ease of Use: EduConnect's user interface (UI) was designed with ease of use and simplicity in mind. An interface that makes it easier to navigate and lowers the learning curve for new users has been created through the combination of a minimalist design philosophy and user feedback loops.

Accessibility: In order to make sure that EduConnect is useable by those with a variety of disabilities, it complies with accessibility standards (such as WCAG), acknowledging the different needs of its user base. The application's design is centered around features like keyboard navigation, high contrast settings, and text-to-speech.

Compatibility: EduConnect is thoroughly tested on a range of browsers, operating systems, and screen sizes to guarantee that it is usable on a wide range of devices and platforms. Regardless of the device —a desktop, tablet, or smartphone—a consistent experience is guaranteed by responsive design principles.

Privacy: EduConnect gives users control over their data and is built with privacy as a top priority. Users can control what information is shared and with whom by putting up clear privacy settings. The application has a transparent privacy policy that clearly outlines the rights, storage, and use of data.

Through an emphasis on both functional and non-functional criteria, this all-encompassing design approach guarantees that EduConnect provides its academic community with a safe, effective, and user-friendly platform.

Section 7

Risk assessment

1. Data Security Breach:

- Potential Failure: Unauthorized access to user data, data leakage, or exposure.
- Risk Exposure: High likelihood due to the sensitive nature of user information (academic aspirations, personal preferences).

- Risk Mitigation Actions: Implement robust encryption protocols, conduct regular security audits, adhere to data protection laws.

2. System Downtime:
- Potential Failure: Application downtime disrupting user access and functionality.
- Risk Exposure: Moderate to high, as downtime could impact user experience and satisfaction.
- Risk Mitigation Actions: Implement redundancy measures, deploy automated monitoring systems, establish rapid response protocols.

3. Poor User Adoption:
- Potential Failure: Lack of user engagement and adoption.
- Risk Exposure: Moderate to high, as it could hinder the achievement of application objectives.
- Risk Mitigation Actions: Conduct thorough user testing, launch targeted marketing campaigns, continuously iterate based on user feedback.

4. Inaccurate Recommendations:
- Potential Failure: Providing irrelevant or inaccurate recommendations to users.
- Risk Exposure: Moderate, as it could diminish user trust and satisfaction.
- Risk Mitigation Actions: Continuously refine machine learning algorithms, validate recommendations, gather user feedback.

5. Data Integration Issues:
- Potential Failure: Issues with syncing user data from university systems.
- Risk Exposure: Moderate, as it could affect the accuracy of personalized recommendations.
- Risk Mitigation Actions: Establish robust data integration protocols, ensure consistency between application and university databases.

6. Privacy Concerns:
- Potential Failure: Breach of user privacy due to inadequate security measures.
- Risk Exposure: High, as it could lead to legal and reputational repercussions.
- Risk Mitigation Actions: Implement strong authentication mechanisms, data anonymization strategies, adhere to data protection laws.

7. Dependency on Third-Party Services:
- Potential Failure: Disruption of services provided by third-party platforms (e.g., calendar integration).
- Risk Exposure: Moderate, as it could impact application features and usability.
- Risk Mitigation Actions: Monitor third-party service status, implement fallback mechanisms, establish communication channels for updates.

Exception handling

1. Data Security Breach:

- Detection Mechanisms: Implement intrusion detection systems (IDS) to monitor for unauthorized access attempts or abnormal behavior within the application.
- Handling Mechanisms: Immediately isolate affected systems upon detection of a breach, activate incident response protocols, and notify affected users and stakeholders about the breach.

2. System Downtime:

- Detection Mechanisms: Deploy continuous monitoring tools to track application performance metrics and system health indicators.
- Handling Mechanisms: Upon detection of downtime, initiate incident response procedures to identify the root cause of the issue, implement temporary workarounds or failover mechanisms, and communicate transparently with users about the issue and expected resolution timelines.

3. Privacy Concerns:

- Detection Mechanisms: Monitor user data access logs and audit trails to detect any unauthorized or suspicious activities that may compromise user privacy.
- Handling Mechanisms: Upon detection of privacy breaches, invoke incident response protocols to assess the scope and impact of the breach, notify affected users and regulatory authorities, and implement corrective measures to prevent future incidents.

4. Dependency on Third-Party Services:

- Detection Mechanisms: Monitor service status and uptime of third-party providers through their status pages or API endpoints.
- Handling Mechanisms: Upon detection of service disruptions from third-party providers, switch to alternative service providers or fallback mechanisms, and communicate with users about the impact and expected resolution timelines.

Section 8

HCI PROTOCOL

1. User Interface Design:

- Ensure the interface is intuitive and user-friendly, with clear navigation and visually appealing design.

- Prioritize key features such as personalized calendars, task lists, and reminder settings for easy access.
- Implement responsive design to accommodate various screen sizes and devices.

2. Calendar Integration:
- Integrate with popular calendar applications such as Google Calendar or Apple Calendar to sync user schedules seamlessly.
- Provide options for users to import existing calendars or manually add events and deadlines within the app.
- Enable users to view their personalized calendar within the app, with color-coded events and intuitive layouts.

3. Task Management and Reminders:
- Allow users to create and manage tasks, assignments, and deadlines within the app.
- Implement reminder settings for upcoming tasks and events, with options for customizable notifications.
- Provide users with the ability to set priority levels, due dates, and recurring tasks for better organization.

4. Personalization and Customization:
- Offer customization options to tailor the scheduling and reminder settings according to individual preferences.
- Allow users to choose notification preferences, time intervals, and notification tones.
- Provide recommendations for optimal scheduling based on user input and preferences.

5. Accessibility and Inclusivity:
- Ensure the app is accessible to users with disabilities by following accessibility guidelines and standards.
- Provide options for font size adjustments, color contrasts, and screen reader compatibility.
- Conduct usability testing with diverse user groups to identify and address any accessibility barriers.

6. Feedback and Iterative Improvement:
- Incorporate feedback mechanisms within the app to gather user input on the scheduling and reminder features.
- Regularly collect user feedback through surveys, ratings, and user interactions to identify areas for improvement.
- Iteratively refine the scheduling and reminder functionalities based on user feedback and usage patterns.

7. Performance and Responsiveness:

- Ensure the app responds quickly to user interactions, with minimal loading times and smooth transitions between screens.
- Optimize performance by minimizing resource consumption and efficiently handling data synchronization processes.
- Conduct performance testing to identify and address any bottlenecks or performance issues.

8. Security and Privacy:

- Implement robust security measures to protect user data, including encryption of sensitive information and secure authentication mechanisms.
- Adhere to data protection regulations and privacy laws to safeguard user privacy and confidentiality.
- Provide transparency to users about data handling practices and obtain consent for data collection and processing.

CCI PROTOCOL

1. API Integration:

- Establish APIs (Application Programming Interfaces) for seamless integration with external calendar services, such as Google Calendar or Apple Calendar.
- Define endpoints for data synchronization, event creation, and retrieval to enable communication between the app and external calendar platforms.

2. Push Notification Service:

- Implement a push notification service to deliver reminders and alerts to users' devices.
- Integrate with push notification platforms (e.g., Firebase Cloud Messaging for Android, Apple Push Notification Service for iOS) to send real-time notifications to users based on their scheduling preferences.

3. Database Management:

- Design and manage a database schema to store user schedules, tasks, and reminder settings securely.
- Implement database CRUD (Create, Read, Update, Delete) operations for efficient data management and retrieval within the app.

4. Authentication and Authorization:

- Implement authentication mechanisms, such as OAuth or JWT (JSON Web Tokens), to verify user identities and secure access to scheduling and reminder features.
- Define authorization rules to control user access levels and permissions within the app, ensuring data privacy and security.

5. Background Task Execution:
- Implement background task execution mechanisms to handle automated processes, such as data synchronization and reminder notifications, without impacting app performance.
- Utilize background processing frameworks (e.g., Android WorkManager, iOS Background Tasks) to execute scheduled tasks efficiently and reliably.

6. Error Handling and Logging:
- Implement error handling mechanisms to capture and log errors occurring during scheduling and reminder processes.
- Utilize logging frameworks (e.g., Log4j for Java, CocoaLumberjack for Swift) to record detailed information about errors, exceptions, and system events for troubleshooting and debugging purposes.

7. Data Encryption and Security:
- Implement data encryption techniques, such as SSL/TLS encryption for network communication and AES encryption for sensitive data storage, to protect user information.
- Ensure compliance with security standards and best practices, such as OWASP (Open Web Application Security Project) guidelines, to mitigate security risks and vulnerabilities.

8. Cross-Platform Compatibility:
- Ensure compatibility with multiple platforms (e.g., iOS, Android) by implementing platform-agnostic code and using cross-platform development frameworks (e.g., React Native, Xamarin).
- Test and optimize app performance and functionality across different operating systems and device types to provide a consistent user experience.

Section 9

Boundary Value Analysis: Added a test case for the user registration component to check if the system validates user-input information, for example, the username, password, and email, with the appropriate minimum and maximum length.
Integration Testing: In addition, a test case was created for the system's integration with university systems to be smooth and the same data consistency to be exhibited.

Functional Testing: New test cases must be provided that will check for accessing personalized calendars and the same with sending direct messages to ensure all functions work properly by all users.
Usability Testing: Implement a user testing scenario that relates to the goal adjustment mechanism so that it is understandable and appropriate.
Security Testing: Installed a test instance for due board publish to ensure the feature fends off SQL injection and cross-site scripting attacks.

Section 10

Remaining Work Needed

Prototype Development:
Create a working model of EduConnect in which you incorporate a final test plan's core functionality item that are indicated by user registration, calendar, personalized goal, direct messages, forum, and access to course information.
Make sure that the interface of the prototype has a user-friendly version, which complies with the visual design and its mock-ups.
Integration with University Systems:
Implement and conduct the interfacing of EduConnect with the university's automatic data system to allow smooth data flow, which is particularly on the user records, courses, and assignments.
Security Implementation:
Security elements like data protection for users and prevention of typical web weaknesses such as SQL injection and cross-site scripting, should be given special attention to discussion board features.
Cross-Platform Compatibility:
Test and fine-tune the prototype model for compatibility across all devices and browsers to provide a seamless user experience.

Load Testing:
Perform the load testing of the prototype to observe how well it operates and how many simultaneous users it can handle without the system crashing or getting slower.
User Feedback Collection:
Make it possible for the users to give their feedback by putting a feedback collection mechanism in the prototype and using this to incorporate their insights and suggestions for improvement.

Documentation:

Prepare compact documentation about the prototype with user guides, technical specifications, and security protocols necessary.

Ethical and Privacy Considerations:

Ensure that the prototype conforms to ethical standards and data regulations and with the policies of data usage and user consent as clear as possible.

Preparation for Final Presentation:

Furthermore, make the preparations for the prototype's final presentation demonstrating the features, the testing process summary discussing the feedback, and making improvements.