Team Final 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

Summary

The Student Assist App, developed by Group V, is an innovative mobile application designed to enhance the educational experience of students by offering personalized advice and support for daily decision-making. Built with machine learning algorithms, the app adapts to the user's goals over time to optimize recommendations. Key features include personalized guidance based on individual academic aspirations and personal preferences, a goal-setting and measurement system, smart scheduling with custom plans and reminders, and resource recommendations catered to users' interests. It also facilitates social and community integration, connecting users with mentors and peers, and incorporates feedback loops for continuous improvement.

The app emphasizes security and privacy, employing robust authentication, data anonymization, and encryption protocols. It is designed to be valuable, reliable, portable, and scalable, addressing a wide range of student needs.

For instance, general students can use it to track their academic progress and schedule, while applicant students can navigate application processes. Non-traditional students benefit from personalized scheduling and career services, and job/internship-seeking students receive tools for resume building and interview preparation. Tutors and mentors can use the platform to connect with and guide students. Functional requirements include user profile management, goal tracking, resource libraries, and personalized recommendations. Non-functional requirements focus on usability, performance, scalability, reliability, security, and privacy.

The prototype will feature a user-friendly interface, scheduling tools, and resource libraries, supported by a cloud-based architecture for iOS and Android platforms. It will undergo iterative testing and revision based on user feedback. The app's system/context model shows its interaction with internal components and external entities like educational institutions and third-party service providers. It is designed to ensure secure and private communication and promote interoperability. In conclusion, the Student Assist App by Group V aims to be a comprehensive tool that supports students in achieving their academic and career objectives by providing a smart, adaptable, and secure platform for managing various aspects of student life.

Functional Requirements

User Profile Creation and Management: Allow users to create and customize their profiles, including academic interests, career goals, and personal preferences. Support for different user roles, such as general students, applicant students, non-traditional students, job/internship-seeking students, and tutors/mentors.

Personalized Guidance and Recommendations: Implement machine learning algorithms to analyze user data and provide personalized academic, career, and personal development recommendations. Feature to set and track personal and academic goals, with progress indicators and suggestions for improvement.

Smart Scheduling and Reminders: Custom schedule creation based on user's academic obligations, deadlines, and preferences. Integration with calendar apps and push notifications for upcoming tasks and deadlines.

Resource Library: Curate and recommend resources tailored to the user's academic interests and career goals, including study guides, articles, videos, and external courses. Enable access to online workshops, tutoring sessions, and courses relevant to the user's field of study or interest.

Social and Community Integration: Features to connect with peers, mentors, and alumni for collaboration, networking, and information sharing. Study groups, discussion boards, and mentorship program functionalities. Feedback and Improvement Loop: Collect user feedback through surveys, ratings, and interaction data to refine recommendations and services. Machine learning models that adapt and improve based on user feedback and behavior.

Career Services: Tools for building resumes, cover letters, and interview preparation. Job and internship search functionalities with filters for industry, location, and role.

Security and Privacy Management: Strong authentication mechanisms, data anonymization, and encryption protocols to protect user data. Compliance with relevant data protection laws and transparency about data usage.

Non-Functional Requirements

Usability: User-friendly interface with intuitive navigation and accessible design for diverse user groups. Personalized dashboards that present relevant information and recommendations clearly.

Performance: Fast response times for user interactions and data processing. Efficient handling of large datasets and user requests to provide timely recommendations and updates.

Scalability: Cloud-based architecture to easily scale resources in response to fluctuating user numbers and data volumes. Modular design to facilitate the addition of new features and user categories.

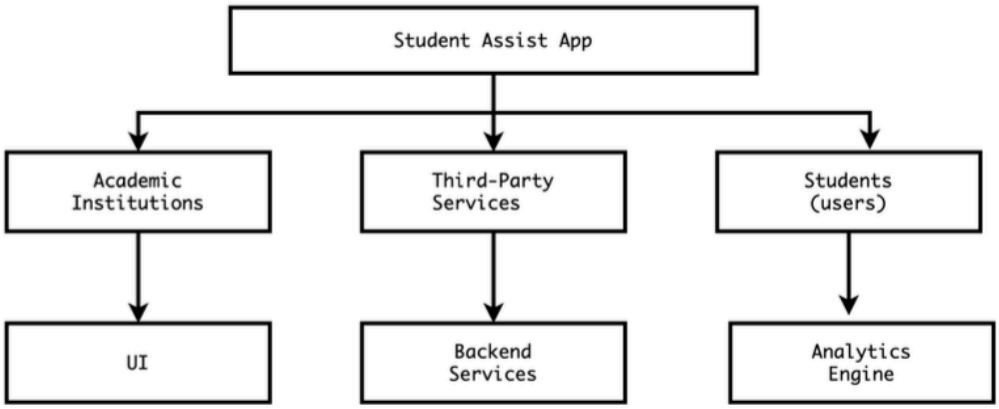
Reliability: High availability of the application with minimal downtime. Robust error handling and data backup mechanisms to prevent data loss.

Security: Implementation of industry-standard security practices to safeguard user information and interactions. Regular security audits and updates to address emerging threats and vulnerabilities.

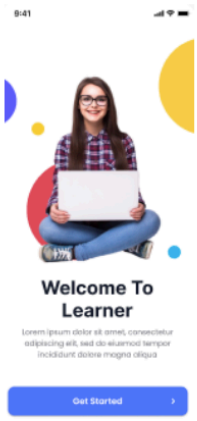
Privacy: Strict adherence to privacy laws and regulations, with clear user consent forms and privacy notices. Features that allow users to control their personal data and preferences, including data deletion options.

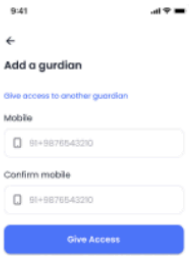
Portability: Compatibility with both iOS and Android platforms to cater to a wide user base. Responsive design to ensure usability across different devices and screen sizes. These requirements will serve as the foundation for the development, testing, and iteration of the Student Assist application, aiming to create a comprehensive tool that supports students in achieving their academic and career objectives.

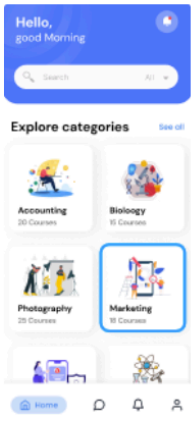
Context Model

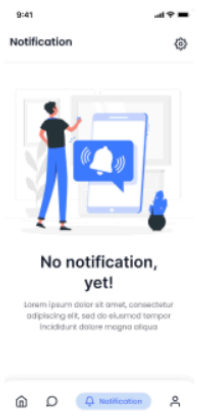


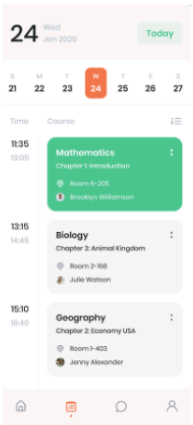
UI sketches

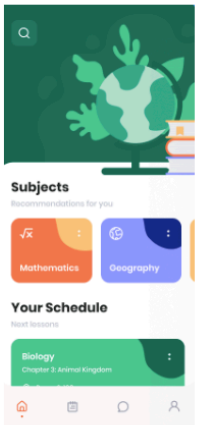












Algorithmic Perspective

Algorithmic Views For Functionalities

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"
```

Algorithm for Student Academic and Social Integration

```
function ManageStudentAcademicAndSocialIntegration(studentID):
    studentProfile = retrieveStudentProfile(studentID)
    personalizedSchedule = generatePersonalizedSchedule(studentProfile)
    academicResources = recommendAcademicResources(studentProfile)
    connectWithPeers(studentProfile)

    if studentProfile.preferences.enableSocialFeatures:
        joinStudyGroups(studentID)
        participateInForums(studentID)

    displayPersonalizedDashboard(personalizedSchedule,
academicResources)
    return updateStudentProfile(studentID, personalizedSchedule,
academicResources)

function joinStudyGroups(studentID):
    availableGroups = fetchAvailableStudyGroups(studentID)
    for group in availableGroups:
        if checkGroupCompatibility(group, studentID):
            joinGroup(group, studentID)

function participateInForums(studentID):
    forumPosts = fetchRecentPosts()
    for post in forumPosts:
        if isRelevant(post, studentID):
            displayPost(post)
            if userWantsToRespond():
                response = getUserResponse()
                submitResponse(post, response, studentID)
```

Algorithm for Time Management System in Student Assist App

Initialize Time Management System

```pseudo
function initializeTimeManagement(studentID):
    schedule = loadStudentSchedule(studentID)
    reminders = loadStudentReminders(studentID)
    return schedule, reminders
```

Generate Schedule

```pseudo
function generateSchedule(studentProfile, currentCourses, personalEvents):
    schedule = new Schedule()
    for course in currentCourses:
        classSessions = getScheduledSessions(course)
        schedule.addSessions(classSessions)
    for event in personalEvents:
        schedule.addEvent(event)
    optimizeSchedule(schedule, studentProfile.preferences)
    return schedule
```

Optimize Schedule

```pseudo
function optimizeSchedule(schedule, preferences):
    if preferences.preferMorningStudy:
        prioritizeTimeBlocks(schedule, "Morning")
    elif preferences.preferEveningStudy:
```

```
        prioritizeTimeBlocks(schedule, "Evening")
    applyConflictResolution(schedule)
    return schedule
```

Set Reminders for Tasks and Deadlines
```pseudo
function setReminders(tasks):
    reminders = []
    for task in tasks:
        deadline = task.deadline
        reminderTime = calculateReminderTime(deadline, task.urgency)
        reminders.append(createReminder(task, reminderTime))
    return reminders
```

Calculate Reminder Time
```pseudo
function calculateReminderTime(deadline, urgency):
    if urgency == "High":
        return subtractTime(deadline, Duration(hours=48))
    elif urgency == "Medium":
        return subtractTime(deadline, Duration(hours=24))
    else:
        return subtractTime(deadline, Duration(hours=12))
```

Create Reminder
```pseudo
function createReminder(task, time):
    return new Reminder(task.description, time)
```

```

#### Apply Conflict Resolution
```pseudo
function applyConflictResolution(schedule):
    conflicts = detectConflicts(schedule)
    for conflict in conflicts:
        resolveConflict(schedule, conflict)
    return schedule
```

Detect and Resolve Conflicts
```pseudo
function detectConflicts(schedule):
    # Check for overlapping time blocks
    return conflicts

function resolveConflict(schedule, conflict):
    # Logic to adjust conflicting sessions
    # This might involve rescheduling less important events
    adjustSchedule(schedule, conflict)
```

Adjust Schedule
```pseudo
function adjustSchedule(schedule, conflict):
    # Move or split events according to priority and availability
```
```

Design Perspective

To provide an in-depth design overview for the EduConnect app, we'll explore essential elements including its architectural framework, user interface design, security measures, and its capability for integration. This overview outlines the structure and operational mechanisms of the app to effectively meet user requirements.

1. **Client-Server Framework**
   - The app is based on a client-server model where the server manages the bulk of data processing, and the client presents an interactive interface to the users.
   - **Clients**: Includes both mobile and web applications accessible to students and faculty.
   - **Server**: Manages backend services, handles requests, and ensures secure data storage.

2. Microservices Strategy
   - The app employs a microservices approach for various services such as user management, messaging, and calendar events, enhancing scalability and simplifying updates.

3. API Gateway Implementation
   - An API gateway interfaces the clients with the microservices, managing request routing, load distribution, and adding an extra security layer.

User Interface Design

1. Adaptive Design

The application is designed to be fully functional on a variety of devices including smartphones, tablets, and desktops, ensuring uniform user experience.

2. User-Friendly Navigation

Features straightforward menus and clear pathways for users to seamlessly toggle between functionalities like messaging, calendar, and goal tracking.

3. Accessibility Considerations

Incorporates features such as high contrast modes, options for text enlargement, and screen reader compatibility to serve users with diverse accessibility needs.

Security Features

1. Secure Authentication and Permissions

- Utilizes OAuth for secure, flexible user authentication.
- Employs role-based access control to ensure users access only appropriate features based on their roles (e.g., student, faculty, admin).

2. Data Protection Measures

- Employs HTTPS for secure transmission of data.
- Encrypts sensitive data both at rest and during transfer using established encryption standards.

3. Proactive Security Practices

Regularly performs security audits and vulnerability checks to identify and address potential security issues.

Integration with Educational Systems

1. Standard APIs

Develops and utilizes standardized APIs for smooth integration with varied university systems, allowing for efficient real-time data updates.

2. Ensuring Data Uniformity

Applies robust mechanisms to maintain data uniformity across the platform and integrated systems, effectively managing data conflicts and redundancies.

3. Flexible Integration Modules

- Designs integration modules to be easily adaptable, facilitating straightforward additions or modifications with new university systems or third-party services.

Performance and Scalability

1. Efficient Load Management

Implements load balancers to effectively distribute requests across several server instances, enhancing response times and availability.

2. Effective Caching

Uses caching techniques to decrease server load and accelerate response times for frequently requested data, improving performance.

3. Dynamic Resource Allocation

Leverages cloud services capable of adjusting resources dynamically based on application demand, ensuring optimal performance during peak usage.

This comprehensive design overview emphasizes the robust, intuitive, and secure nature of the EduConnect app, equipped to integrate seamlessly with educational institution systems, providing a well-rounded and efficient solution.

Functional Testing

Methods:
Unit Testing: Test individual components for correctness.
Integration Testing: Ensure that the app's modules work together seamlessly.
System Testing: Validate the complete and integrated software product.
Usability Testing
Purpose: To ensure the app is user-friendly, intuitive, and accessible to all users, including those with disabilities.

Methods:
User Interface Testing: Check for graphical user interface elements like layouts, buttons, and navigation flows.
Accessibility Testing: Ensure compliance with accessibility standards such as WCAG for users with disabilities.
User Acceptance Testing (UAT): Conduct testing with real users to ensure the app meets their needs and expectations.
Performance Testing
Purpose: To ensure the app performs well under various conditions, particularly under high load.

Methods:
Load Testing: Determine how the system behaves under normal and peak loads.
Stress Testing: Identify the limits at which the app fails and how it recovers from failure.
Scalability Testing: Verify the app’s ability to scale up or down based on user demand.
Security Testing
Purpose: To identify vulnerabilities within the app and ensure that user data is protected from unauthorized access.

Methods:
Penetration Testing: Simulate attacks on the app to find exploitable vulnerabilities.
Vulnerability Scanning: Regular scans of the app’s infrastructure and codebase to detect security weaknesses.

Security Audits: Comprehensive reviews of the app's security architecture and compliance with security standards.
Integration Testing
Purpose: To verify that the app integrates effectively with external systems like university databases and third-party services.

Methods:
API Testing: Ensure that all API integrations work as expected, both sending and receiving data correctly.
Third-Party Integration Testing: Test the app's functionality with third-party services to ensure compatibility and reliability.
Regression Testing
Purpose: To ensure that new updates, bug fixes, or enhancements do not introduce new faults into existing working features.

Conclusion

The EduConnect app is designed to enhance the educational experience by providing robust functionalities like goal setting, direct messaging, and personalized calendars, integrated seamlessly with university systems. Its sophisticated architecture ensures scalability and performance, while a comprehensive testing strategy guarantees functionality, usability, and security. By leveraging advanced technology, EduConnect offers a user-friendly platform that supports students in effectively managing their academic and personal commitments, making it a quintessential tool for modern educational needs.

Future Enhancement

To propel its utility and appeal forward, future enhancements for the EduConnect app could focus on incorporating artificial intelligence for personalized learning experiences, extending integration capabilities with educational and professional development platforms, and enhancing user interactivity through gamification. Additionally, improving offline functionalities, expanding language support, and bolstering security measures will ensure that the app remains a robust, accessible, and secure resource for students globally. By continuously adapting to technological advancements and user feedback, EduConnect aims to offer an increasingly supportive and enriched educational environment.