Software Requirement Specifications
CSC4351 Capstone I

Team #: 10
Team Name: Team 10
Members: Yasaswi Kompella
Mubashar Mian
Divya Patel
Krish Patel
Hari Thavittupalayam Manivannan

Section: Wednesday

Table of Contents

Introduction / Overview

To build a schedule management app focused on accessibility for people with ADHD. The app simplifies task management by allowing users to focus on immediate tasks (daily or half-day), reducing cognitive load and anxiety, and providing AI-powered suggestions to make task execution easier.

General description

Tiny Tasks is a schedule management app designed for accessibility and usability, particularly for people with ADHD. The app reduces cognitive overload by allowing users to focus on smaller, actionable tasks through AI-powered decomposition. It integrates with multiple calendar APIs, like Google Calendar, to enhance scheduling and supports task prioritization, reminders, and task categorization.

The app's machine learning models adapt to user behavior over time, personalizing suggestions and improving task decomposition accuracy. By focusing on simplicity and accessibility, Tiny Tasks helps users stay organized and productive while reducing stress.

Functional Requirements

FR1: User Authentication and Profile Management

- Description: Users must be able to create an account, log in, and manage their profile. This ensures that task data and preferences are saved for each user.
- Inputs:
    - Username/Email: String input, source: user entry.
    - Password: String input, source: user entry.
    - Profile Data (e.g., name, preferences): String/Boolean/Integer inputs, source: user entry.
- Units of Measure: N/A (standard text fields for email and password, selection options for preferences).
- Range of Valid Inputs:
    - Username/Email: Valid email format (e.g., "example@domain.com").
    - Password: Minimum 8 characters, with a mix of letters, numbers, and symbols.
    - Profile Data: Choices for preferences (e.g., task view, notification settings).

FR2: Task Creation and Management

- Description: Users should be able to add, edit, and delete tasks. Tasks can be categorized and assigned priority levels.
- Inputs:
    - Task Title: String input, source: user entry.
    - Task Description: Optional string input, source: user entry.
    - Due Date/Time: DateTime input, source: user entry or Google Calendar sync.
    - Priority Level: Enum (high, medium, low), source: user selection.
    - Task Category: Enum (School, Work, Personal), source: user selection.
- Units of Measure: Date format (e.g., MM/DD/YYYY), time in 24-hour or 12-hour format, priority/categorization as enums.
- Range of Valid Inputs:
    - Task Title: Up to 100 characters.
    - Due Date/Time: Valid date and time entries within a 1-year range.
    - Priority Level: Selection from pre-defined choices (high, medium, low).
    - Task Category: Selection from pre-defined choices.

FR3: Daily and Half-Day Task View

- Description: Users should be able to toggle between viewing a list of tasks for the entire day or just the first or second half of the day.
- Inputs:
    - View Selection: Enum (daily view, half-day view), source: user selection.
    - Current Date/Time: DateTime input, source: system clock.
- Units of Measure: Date format (MM/DD/YYYY), time as hours/minutes.
- Range of Valid Inputs:
    - View Selection: Choices between full-day view or half-day view (AM or PM).
    - Current Date/Time: Automatically sourced from the system.

FR4: Google Calendar Integration

- Description: Users can sync tasks and events from Google Calendar. The app should fetch events from the user's calendar, allowing them to select which events to add to the task list.
- Inputs:
    - Google Calendar Authorization Token: String input, source: Google API (user provides consent).
    - Calendar Events: Task data from Google Calendar, source: Google API.
- Units of Measure: Date format (MM/DD/YYYY), time as hours/minutes.
- Range of Valid Inputs:
    - Authorization Token: Must be a valid OAuth token.

    - Calendar Events: Valid date, time, and title for each event pulled from Google Calendar.

FR5: Task Prioritization and Filtering

- Description: Users can filter tasks by priority (high, medium, low) and task category (school, work, personal). This helps them focus on the most important tasks.
- Inputs:
    - Filter Criteria (Priority): Enum (high, medium, low), source: user selection.
    - Filter Criteria (Category): Enum (School, Work, Personal), source: user selection.
- Units of Measure: N/A (Enum selection).
- Range of Valid Inputs:
    - Priority Level: Must be one of the defined levels (high, medium, low).
    - Category: Must be one of the pre-defined categories.

FR6: Notifications and Reminders

- Description: The system should send notifications to remind users of upcoming tasks and deadlines. Users can customize when they receive these reminders (e.g., 30 minutes before, 1 hour before).
- Inputs:
    - Notification Time Preference: Enum (15 min, 30 min, 1 hour, etc.), source: user selection.
    - Task Data (Due Date/Time): DateTime input, source: task details.
- Units of Measure: Date format (MM/DD/YYYY), time as hours/minutes.
- Range of Valid Inputs:
    - Notification Time Preference: Pre-defined values (e.g., 15 min, 30 min, 1 hour, 1 day).
    - Task Data (Due Date/Time): Must be a valid date and time format.

FR7: AI-Powered Tast Decomposition

- Description: The system utilizes AI to break down user-added tasks into smaller, actionable steps to improve focus and productivity. The AI adapts to user behavior over time and provides personalized suggestions.
- Inputs:
    - Task Title: String input, source: user entry.
    - Task Description: Optional string input, source: user entry.
    - Task Category: Enum (School, Work, Personal), source: user selection.
    - Priority Level: Enum (High, Medium, Low), source: user selection.
    - Feedback Rating: Integer input (1-5), source: user feedback.
- Units of Measure:
    - Task Title/Description: Length in characters.

    - Task Category: Predefined categories.
    - Priority Level: Predefined levels.
    - Feedback Rating: Numeric scale (1-5).
- Range of Valid Inputs:
    - Task Title: Minimum 1 character, maximum 100 characters.
    - Task Description: Minimum 0 characters (optional), maximum 500 characters.
    - Task Category: Must match one of the predefined enums (School, Work, Personal).
    - Priority Level: Must match one of the predefined enums (High, Medium, Low).
    - Feedback Rating: Must be an integer between 1 and 5.

Interface Requirements

- Description: The user interface must be simple, clean, and accessible, designed with users who have ADHD in mind. The UI should minimize cognitive load and provide a clear display of tasks.
- Components:
    - Task List Interface: Displays tasks for the day or half-day based on user preference.
    - Input Forms: For adding and managing tasks, including text fields, date pickers, and dropdowns for categories and priorities.
    - Navigation: Intuitive buttons and menus for switching between different views (e.g., full day, half-day, settings, etc.).
- Platform: Mobile application built using Flutter, ensuring cross-platform support for both iOS and Android devices.

Performance Requirements

PR1: Task Load and Display Speed

- Description: The app must quickly load and display tasks, ensuring a smooth user experience, especially for users with ADHD who may become frustrated by delays.
- Requirement:
    - The app must be able to load and display a user's daily task list in under 2 seconds after login or upon switching between views (daily view, half-day view).
    - When adding or editing tasks, changes must be reflected in real-time or within 500 milliseconds if using a backend like Firebase.
- Conditions:
    - Normal Load: The app must maintain performance with up to 50 tasks displayed per day.

    - High Load: The app should still load within 3 seconds under heavy task loads (e.g., up to 100 tasks).

PR2: Google Calendar Sync Time

- Description: The app should synchronize with Google Calendar efficiently, ensuring that users' tasks and events are up to date without significant delays.
- Requirement:
    - The app must synchronize with Google Calendar and update the task list in under 5 seconds for up to 10 events.
    - Any changes made to tasks in the app should be reflected on Google Calendar within 2 seconds after submission.
- Conditions:
    - Sync must occur in real-time or near real-time while ensuring that the app remains responsive, even with intermittent internet connections.

PR3: Notification Delivery

- Description: The app should reliably send reminders and notifications at the appropriate time, especially when users have critical tasks or deadlines approaching.
- Requirement:
    - Notifications must be delivered within ±30 seconds of the user-defined time (e.g., if a reminder is set for 3:00 PM, it must be delivered between 2:59:30 PM and 3:00:30 PM).
    - The system should be able to handle sending up to 10 notifications per day without performance degradation.
- Conditions:
    - Notifications should be delivered accurately in low-network conditions or when the app is running in the background.

Design Constraints

Mobile Platform Compatibility

- Constraint: The app must be compatible with both iOS and Android devices.
- Explanation: To reach a broad audience, the app should function seamlessly on both major mobile operating systems. This necessitates the use of a cross-platform framework like Flutter. Developing separate native apps for each platform would increase development time and cost.
- Limitation: The design must consider platform-specific UI components (such as the back button on Android) and system-level features (e.g., push notifications), ensuring a consistent user experience across both platforms.

Accessibility Standards

- Constraint: The app must comply with WCAG 2.1 (Web Content Accessibility Guidelines) to ensure it is accessible to people with disabilities, particularly ADHD.
- Explanation: Since the app is designed for users with ADHD, special attention must be given to accessibility features such as large, readable fonts, high-contrast themes, and simple navigation to minimize distractions.
- Limitation: The user interface (UI) must avoid clutter and complex navigation patterns. Developers must also ensure that screen readers and other assistive technologies can interact with the app effectively.

Limited Budget and Development Resources

- Constraint: The development team has limited time and resources, restricting the number of features and the complexity of the system.
- Explanation: Given constraints on project timelines, the app must prioritize essential features like task management and Google Calendar integration while minimizing feature creep.
- Limitation: More advanced features like AI-based task suggestions, integration with third-party apps, or extensive offline capabilities may need to be postponed or scaled back for a future release.

AI and Machine Learning Integration

- Constraint: The app must effectively utilize AI and machine learning to deliver personalized task suggestions and break tasks into smaller, manageable steps.
- Explanation: AI should enhance user productivity by generating subtasks dynamically, and machine learning should adapt these suggestions based on user behavior and feedback.
- Limitation: Training machine learning models requires computational resources and may need periodic updates to remain relevant to user needs. The design must optimize AI processes to function efficiently on mobile devices.

API Integration and Usage Limits

- Constraint: The app must integrate multiple calendar APIs, including Google Calendar, while adhering to their usage limits.
- Explanation: Calendar APIs have rate limits for requests per minute/hour. Efficient usage is necessary to avoid exceeding these thresholds and maintain synchronization reliability.
- Limitation: The app must batch API requests, minimize unnecessary syncs, and cache task data locally to reduce the frequency of API calls.

User Data Privacy and Security

- Constraint: The app must comply with GDPR, CCPA, and other privacy regulations to ensure secure handling of sensitive user data.
- Explanation: As the app manages personal schedules and potentially mental health-related data, robust encryption and secure authentication are critical.
- Limitation: All data, including AI-generated subtasks and sync data, must be securely stored (encrypted) and transmitted (via HTTPS). Implementing additional security layers like two-factor authentication (2FA) may require extra development effort.

Performance on Low-End Devices

- Constraint: The app must perform smoothly on low-spec devices with limited processing power, memory, and storage.
- Explanation: The target audience includes students who may rely on budget devices. The app must minimize resource-intensive operations to ensure wide accessibility.
- Limitation: AI and machine learning features must be optimized to run efficiently on-device or rely on lightweight backend services.

Non-Functional Attributes

Security

- Description: The system must ensure the confidentiality, integrity, and availability of user data.
- Details:
    - User Data Protection: All user data (tasks, personal settings, etc.) must be securely stored in an encrypted format, both in transit (using HTTPS) and at rest (using encryption on storage systems like Firebase).
    - Authentication: User authentication must use secure methods such as OAuth 2.0 or Firebase Authentication. Multi-factor authentication (MFA) can be added to improve security for users who wish to enable it.
    - Data Access Control: Only authorized users can access their own data, and proper security protocols must be in place to prevent unauthorized access.

Portability

- Description: The system must be able to run on multiple platforms without significant modifications.
- Details:
    - Cross-Platform Support: The app must be built using a cross-platform framework like Flutter to ensure compatibility with both iOS and Android devices.

    - UI Adaptability: The user interface must adapt to different screen sizes and resolutions, ensuring the app functions well on both phones and tablets.

Reliability

- Description: The system must be dependable, with minimal downtime and accurate task management.
- Details:
    - Availability: The system should be available at least 99.9% of the time to ensure users can access their tasks whenever needed. Downtime must be minimal, and any scheduled maintenance should be announced in advance.
    - Fault Tolerance: The app should handle errors (such as network disruptions or failed API calls) gracefully, ensuring that users do not lose data or experience crashes. If the app loses connection, it should work in offline mode and sync once a connection is reestablished.
    - Backup and Recovery: In case of data loss or corruption, the app should have mechanisms in place to recover data from cloud storage or provide recent backups.

Usability

- Description: The app must be easy to use and provide a smooth user experience, especially for users with ADHD.
- Details:
    - Simple Navigation: The interface should focus on simplicity, with minimal distractions, large buttons, and intuitive navigation that caters to users who may have attention challenges.
    - Minimal Cognitive Load: The app should present information in a way that reduces mental strain. For example, displaying only today's tasks or half-day tasks helps users focus on manageable segments of time.
    - User Feedback: The system must provide immediate feedback to user actions (e.g., when adding or completing a task) to confirm that the app has successfully processed the user's inputs.

Scalability

- Description: The system must be able to handle an increasing number of users and tasks without a decline in performance.
- Details:
    - User Base Growth: The app architecture (including backend services like Firebase) should be designed to scale as the user base grows from dozens to potentially thousands of users without significant refactoring.

    - Task Data Volume: The system must handle a growing number of tasks per user without slowing down. For instance, even if a user adds hundreds of tasks, the app should remain responsive and load tasks efficiently.
    - Cloud Infrastructure: Using scalable cloud infrastructure like Firebase ensures that as demand increases, the system can automatically scale to meet the needs without requiring manual intervention.

Data Integrity

- Description: The system must ensure the accuracy and consistency of data throughout its lifecycle.
- Details:
    - Task Synchronization: When syncing with Google Calendar or other external services, the system must ensure that no tasks are lost or duplicated. Any updates should be reflected correctly on both the app and Google Calendar.
    - Data Validation: Input validation must be enforced (e.g., valid dates for tasks, required fields for new entries) to prevent incorrect or incomplete data from being entered into the system.
    - Consistency Across Devices: When using the app on multiple devices, task lists, settings, and notifications must remain consistent and synchronized in real time.

Maintainability

- Description: The system must be easy to maintain, update, and modify over time.
- Details:
    - Modular Design: The app's architecture must follow a modular design pattern, making it easier to add new features, fix bugs, or make improvements without affecting the overall system.
    - Code Readability: Code should be well-documented and follow best practices, ensuring that future developers can easily understand and work on the project.
    - Versioning: Updates and patches must be managed effectively, ensuring backward compatibility and avoiding issues when rolling out new versions.

Application Compatibility

- Description: The system must be compatible with other third-party services or applications, especially those that users commonly used for task and calendar management.
- Details:
    - Google Calendar Integration: Seamless integration with Google Calendar API is essential for syncing tasks and managing events across platforms.

- Push Notifications Compatibility: The app must work with native notification services on both Android and iOS, ensuring that reminders and alerts are delivered reliably.
- Third-Party Tools: The system should be built with flexibility in mind, so that future integrations with other productivity tools (e.g., task management apps or note-taking tools) can be easily implemented.

Reusability

Description: The components and modules developed for Tiny Tasks should be reusable in other projects or future updates.

- Details:
  - Component-Based Architecture: The UI components (e.g., task list, task input forms, etc.) should be designed in a modular way so they can be reused or adapted in future versions of the app or in different applications.
  - API Integration Modules: The logic for Google Calendar integration and other external API calls should be developed as separate modules, making them reusable in other systems.

Preliminary Schedule and Budget

Project Kickoff and Requirements Gathering

- Date: Week 1 (September 15-21, 2024)
  - Description: Initial team meeting, discuss project objectives, gather functional and non-functional requirements, and assign roles.

Design Phase

- Date: Week 2-3 (September 22 - October 5, 2024)
  - Description: Design the app architecture, create wireframes and UI mockups, and finalize technical specifications.

Development Phase

- Date: Week 4-8 (October 6 - November 2, 2024)
  - Description: Develop the core features, including user authentication, task creation and management, and Google Calendar integration.

Testing and Debugging

- Date: Week 9-10 (November 3 - November 16, 2024)

- Description: Conduct thorough testing, fix bugs, and refine the user interface based on feedback.

Final Integration and Optimization

- Date: Week 11 (November 17 - November 23, 2024)
  - Description: Integrate all features, optimize performance, and ensure cross-platform compatibility.

Final Testing and User Feedback

- Date: Week 12 (November 24 - November 30, 2024)
  - Description: Final round of testing, gather user feedback, and make final adjustments.

Project Presentation and Documentation

Date: Week 13-14 (December 1 - December 14, 2024)
- Description: Prepare project presentation, finalize documentation, and submit the final project.

Preliminary Budget

- Development Tools and Software
  - Estimated Cost: $0
  - Details: Utilizing open-source tools like Flutter, Firebase (free tier), and Google Calendar API. No cost anticipated for development tools.
- Cloud Services and API Costs
  - Estimated Cost: $0 - $50
  - Details: Utilizing Firebase for backend services. If usage exceeds the free tier, a minimal cost for additional usage may be incurred.
- Testing Devices and Emulators
  - Estimated Cost: $0
  - Details: Use of existing personal devices and free emulators for testing. No additional cost expected.
- Miscellaneous Expenses
  - Estimated Cost: $50
  - Details: Potential small expenses for documentation tools, user surveys, or contingency costs.
- AI Integration Costs
  - Estimated Cost: $50
  - Details: API usage for AI and ML services
- Calendar API Costs

    - Estimated Cost: $20
    - Details: if usage exceeds free tier
- AI Integration Costs
    - Estimated Cost: $50
    - Details: API usage for AI and ML services

Total Estimated Budget: $150 - $200

Appendices

Acronyms and Abbreviations
ADHD, API, CCPA, GDPR, MFA, OAuth, PR, UI, ADHD, API, GDPR, CCPA, MFA, AI, ML.
Definitions
Cognitive Load, Cross-Platform Support, Flutter, Firebase, Google Calendar Integration, Minimalistic Design, OAuth Token, Task Prioritization.
References
Google Calendar API Documentation, WCAG 2.1 Guidelines, GDPR Compliance Information, CCPA Compliance Guide.
Assumptions
Users' familiarity with Google Calendar, primary use on smartphones and tablets, varying degrees of technical skill.
Glossary of Terms
Task Management, Task View, User Authentication.
PR: Performance Requirements – specifications related to the system's performance, such as speed, reliability, and efficiency.
UI: User Interface – how the user and a computer system interact, particularly the use of input devices and software.

- Definitions
    - Cognitive Load: The mental effort required to learn or complete a task, often considered in software design to minimize stress and improve usability.
    - Cross-Platform Support: The capability of software to run on multiple operating systems, such as iOS and Android, without needing modifications.
    - Flutter: An open-source UI software development kit created by Google used to develop cross-platform applications for Android, iOS, and more.
    - Firebase: A platform developed by Google for creating mobile and web applications that provides cloud-based backend services like real-time databases, authentication, and analytics.
    - Google Calendar Integration: The process of syncing or connecting an application with Google Calendar to allow users to access or manipulate calendar events from within the app.

    - Minimalistic Design: A design approach that focuses on simplicity by reducing visual and functional complexity, emphasizing the essentials.
    - OAuth Token: A string of characters that acts as a security credential for authorization, granting a user or system access to a resource.
    - Task Prioritization: The process of organizing tasks by importance, urgency, or other factors to help users focus on what's most critical.
    - Task Decomposition: Breaking a large task into smaller, actionable steps.
    - AI: Artificial Intelligence used for task suggestion and automation.
    - Machine Learning (ML): Training models to adapt to user preferences and improve task suggestions over time.
- References
    - Google Calendar API Documentation: Google Calendar API Documentation
    - WCAG 2.1 Guidelines: Web Content Accessibility Guidelines that define how to make web content more accessible to people with disabilities. WCAG Guidelines
    - GDPR Compliance Information: Official documentation and resources on GDPR compliance for handling personal data in the European Union. GDPR Information
    - CCPA Compliance Guide: Information and resources for complying with the California Consumer Privacy Act. CCPA Guide
    - Google Calendar API Documentation
    - WCAG 2.1 Guidelines
- Assumptions
    - Users have basic familiarity with Google Calendar and mobile applications.
    - The app will primarily be used on smartphones and tablets, and users will generally have access to reliable internet connections for syncing data with Google Calendar.
    - Users may have varying degrees of technical skill and attention management, hence the focus on a minimalistic, easy-to-navigate user interface.
- Glossary of Terms
    - Task Management: The process of managing a task through its life cycle, including planning, testing, tracking, and reporting.
    - Task View: A mode in the app that displays tasks either for the full day or for half of the day (morning or afternoon).
    - User Authentication: The process of verifying the identity of a user before granting them access to the system or its resources.