Software Requirements Specification

for

Unbind

Version 1.0

Prepared by Aaron Yang

unbind-me

11.8.2024

Table of Contents

Table of Contents....1
Revision History....2
1. Introduction....3
1.1 Purpose....3
1.2 Document Conventions....3
1.3 Intended Audience and Reading Suggestions....3
1.4 Project Scope....3
1.5 References....3
2. Overall Description....3
2.1 Product Perspective....3
2.2 Product Features....3
2.3 User Classes and Characteristics....3
2.4 Operating Environment....3
2.5 Design and Implementation Constraints....4
2.6 User Documentation....4
2.7 Assumptions and Dependencies....4
3. System Features....4
3.1 Functional Requirements....4
4. External Interface Requirements....4
4.1 User Interfaces....4
4.2 Hardware Interfaces....4
4.3 Software Interfaces....4
4.4 Communications Interfaces....4
5. Nonfunctional Requirements....5
5.1 Performance Requirements....5
5.2 Safety Requirements....5
5.3 Security Requirements....5
5.4 Software Quality Attributes....5
6. Appendices....5
6.1 Glossary....5
6.2 Data Models....5
6.3 Process Flowcharts....5
6.4 Other Supporting Information....5

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This SRS document is to provide specifications, details, and technical requirements for the screen-time mobile based app “Unbind”. This is for those who want to know the underlying components of how the app works and what was used to create it.

1.2 Document Conventions

“Screen-time” essentially refers to the amount of time someone spends on an electronic device with a display and is often used to convey how much one uses their device in a given time period, such as “5 hours of screen time today”. Though screen-time can be used to refer to time on any device, this document will specifically focus on mobile device usage. “AI” refers to artificial intelligence, which is a general subtopic of computer science that deals with machine learning and neural networks (basically a computer trained on datasets and algorithms), which is used to create an “intelligent” model that is completely based off of the data it was given and the way it was processed through the algorithm used. This can be used to create tools such as LLMs (large language models), smart robots, etc.

1.3 Intended Audience and Reading Suggestions

Identify the intended readers and suggest how they should read the document.

1.4 Project Scope

This project is to create an app that limits a user’s screen-time but will differentiate itself from other well-known apps with its main feature: artificial intelligence. The objective of this project is to create a fully functional app that implements both basic screen-time blocking functionality as well as AI implemented into app to recognize patterns in user screen-time as well as give tasks based off user input, which is put through the AI to generate unique tasks and goals to fit with the desired input, e.g. “Calculus” would give you random calculus problems, or more a more specific scenario: “Integrals, Riemann sums” would test your knowledge on those specific topics.

1.5 References

List any documents or resources referenced in the SRS.

2. Overall Description

2.1 Product Perspective

Explain the context and origin of the system.

2.2 Product Features

Screen-time limits: Allows the user to pick out distracting apps, which are then controlled and monitored by the app (Unbind) to make sure the user cannot access them unless there is a specific event (such as “free time” or completing a task) that the app can be accessed.
AI Tasks: Uses AI to understand and interpret what the user inputs in order to generate tasks for the user to complete.
Proof system: Has you upload evidence, such as a photo, of your task, which is also processed through AI to verify you have finished your task and give feedback.
Parental Controls: Locks the app behind a passcode for parents to control in order to safeguard the app from their children from changing settings.

2.3 User Classes and Characteristics

Describe the different types of users and their characteristics.
This app is targeted towards young teens and adults. The people who seek this app likely struggle with controlling the time they spend on distracting apps and getting side-tracked from being productive.

2.4 Operating Environment

The app is intended to work on the latest versions of Android and iOS and should be functional on any device it is installed in provided it has up-to-date software. This app will also require an internet connection as it makes requests to servers to generate tasks, as the AI will *likely* be processed on server, though it is possible to cache some tasks on the app in specific cases, meaning a functional offline mode should be feasible.

2.5 Design and Implementation Constraints

One major constraint will be the token cost of GPT3.5 requests, as they cost upwards of $0.0080 per 1000 tokens. Though this is not a problem on a small scale, this app is meant to be shipped to millions of users and will therefore require the application to have monetization and subscriptions. We also have to pay App Store fees to ship the app to iOS, which is $100/year. The development of this app also requires emulation for iOS and Android to make sure the backend works on their respective operating systems. Android emulation is free but requires hardware acceleration and high-end hardware requirements. iOS testing will also require us to sideload the app to test it and requires an Apple device (unless we get a MacBook).

2.6 User Documentation

List the user documentation components (such as user manuals, online help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.
We will have a dedicated website for this app, which will have proper tutorials and demonstrations to teach users how to use the app. We will also host an FAQ which will address concerns and questions, as well as have a contact email for user support.

2.7 Assumptions and Dependencies

- Users will have consistent internet access when the app's AI functionality is running
- Necessary permissions will be granted by the user for the app to work
- The app will depend on the cross-platform functionality of the app framework "React Native", which will ensure smooth functionality across both platforms.
- The app will rely on an AI token (such as GPT3.5) to make AI requests to generate tasks for the user.

3. System Features

3.1 Functional Requirements

- Users must be able to add, edit, and delete tasks, including specifying task details such as title, due date, priority, and tags.
- Tasks should be grouped by category, priority, or due date for easy management.
- The app should support recurring tasks with customizable repeat intervals.
- Based on the user's input (e.g., location, time, preferences), the AI should suggest tasks and assist with prioritization.
- The AI should adapt over time to improve the relevance of suggestions based on user behavior.
- The app must track and analyze screen time data for both overall usage and usage by specific apps.
- Users must be able to set screen time limits, receive notifications when nearing limits, and view screen time trends.
- The app must generate weekly and monthly productivity reports, highlighting trends, completed tasks, and areas for improvement.
- Reports must include visual representations of data, such as graphs or charts, for easy interpretation.
- The app must provide timely notifications for upcoming tasks, deadlines, and goal achievements.
- Users must be able to customize notification preferences, including reminder intervals and specific task reminders.
- Users must be able to customize app themes, notification settings, and layout preferences to tailor the experience.
- The app should allow users to set productivity goals and track progress against these goals.

4. External Interface Requirements

4.1 User Interfaces

Home Dashboard

- A visually appealing, organized dashboard displaying an overview of today's tasks, calendar events, and productivity stats.
- Quick access to the most used features, such as adding a new task, viewing focus mode, and accessing recent notifications.

- A section showing AI-generated task suggestions based on location, time of day, and task priorities.

Task Management Interface

- Intuitive task creation form with input fields for title, due date, priority, category, tags, and notes.
- Clear icons and buttons for adding, editing, or deleting tasks, with easy drag-and-drop functionality for reordering tasks.
- Task lists should be grouped by categories (e.g., today, upcoming, overdue) and display color-coded priorities for quick recognition.

Screen Time Management Interface

- A screen displaying overall screen time stats, with sections for each app or category of apps.
- Progress bars and graphs to visualize screen time usage trends and highlight time spent in focus mode vs. other apps.
- Easy-to-set screen time limits with toggle options for reminders and customizable alerts when nearing limits.

Productivity Insights and Reporting Interface

- A reporting dashboard showing weekly and monthly productivity trends with charts for completed tasks, screen time, and focus mode usage.
- Visual breakdowns of time spent across different task categories and productivity scores to help users track progress.
- An option to export reports or share insights directly from the app.

Focus Mode and Task Timer Interface

- A dedicated focus mode screen with a large, central timer displaying time remaining in the session.
- Minimalistic design with limited distractions, including controls to pause or end the session early.
- Customizable options for setting focus session lengths and intervals, along with a visual progress bar.

Cross-Platform Sync Interface

- Account settings page with options to view sync status, manually trigger data synchronization, and manage backup preferences.
- User-friendly error messages for sync issues and options to troubleshoot common issues, such as reconnecting to the internet.

4.2 Hardware Interfaces

The app should be compatible with iOS and Android devices, supporting a range of hardware configurations, including different screen sizes and resolutions.
Compatibility with mobile processors and sufficient RAM to ensure smooth performance, even with AI-based task suggestions and data syncing.

4.3 Software Interfaces

Operating System Compatibility

iOS and Android OS: The app must be compatible with the latest versions of iOS and Android, as well as a minimum of two previous OS versions to support a broader user base.
Flutter Framework: The app will be developed using Flutter, allowing it to run consistently across both iOS and Android platforms with minimal differences in functionality.

Screen Time and Usage Tracking APIs

iOS Screen Time API: For iOS devices, the app will integrate with Apple's Screen Time API to track app usage and provide insights.
Android UsageStatsManager API: For Android devices, the app will use the UsageStatsManager API to collect screen time data, enabling screen time management features.

4.4 Communications Interfaces

Wi-Fi and Cellular Data:

The app requires an active internet connection (via Wi-Fi or cellular data) for syncing data, accessing cloud-based features, and receiving real-time notifications. The app should notify users if internet access is unavailable when performing actions that require connectivity.

Push Notification Services:

Firebase Cloud Messaging (FCM): For Android devices, FCM will handle push notifications for reminders, updates, and alerts. It requires device registration with Firebase, enabling real-time message delivery.
Apple Push Notification Service (APNs): For iOS devices, APNs will manage push notifications. Notifications are sent securely via Apple's servers to deliver task reminders, productivity alerts, and location-based notifications.

Google Drive / iCloud Integration:

For optional data backup and recovery, the app will communicate with Google Drive or iCloud APIs using secure HTTPS protocols to store and retrieve user data backups.

4.5 Performance Requirements

Response Time

- App Launch: The app should load within 3 seconds on devices meeting minimum hardware requirements.
- Task Creation/Editing: Users should be able to create, edit, or delete a task within 1 second of initiating the action.
- Navigation and UI Response: All in-app navigation, including switching between screens (e.g., from dashboard to calendar), should respond within 0.5 seconds.
- Notification Delivery: Push notifications (e.g., task reminders, focus mode alerts) should be delivered within 2 seconds after a specified trigger time or event.

4.6 Safety Requirements

The app will prioritize user safety through data encryption, privacy compliance (e.g., GDPR and CCPA), and secure authentication methods like multi-factor authentication (MFA) and password protection. Location-based features will be opt-in and minimally used, ensuring user control and resource efficiency to prevent battery drain. Screen time management will include customizable, non-intrusive notifications, as well as periodic break reminders for health. Compliance with platform safety guidelines and secure, vetted third-party APIs will ensure data protection. Regular backups and data recovery options will be available to prevent data loss, promoting a secure and reliable user experience.

4.7 Security Requirements

The app's security will focus on robust data protection, utilizing encryption for data both in transit and at rest, along with strict access controls to ensure only authorized personnel can access sensitive information. Secure user authentication will be implemented, with multi-factor authentication (MFA) as an optional feature and enforced strong password requirements. Regularly updated security protocols will guard against unauthorized access and data breaches. Additionally, session management, including automatic logout after inactivity, and notification of any unusual login attempts will help protect user accounts.

4.8 Software Quality Attributes

- **Reliability**: The app will maintain high reliability through rigorous testing to minimize crashes, bugs, and data loss, ensuring a stable experience for users across various devices and operating systems. Automated backups and recovery options will further support data integrity.

- **Usability**: Designed with a user-friendly, intuitive interface, the app will prioritize ease of use for all users, featuring straightforward navigation, customizable settings, and clear prompts. Accessibility features will enhance usability for individuals with disabilities.

- **Performance**: The app will be optimized for quick response times, efficient data syncing, and minimal battery usage, especially in resource-intensive functions like geofencing and AI-based task recommendations. Performance will be monitored to ensure fast, seamless operation even under heavy usage.

- **Scalability**: Built on a scalable architecture, the app will support increased user numbers, feature expansion, and data growth without degradation in performance. Cloud-based services will ensure smooth functionality as demand scales.

- **Maintainability**: Using modular code and industry-standard development practices, the app will be maintainable, allowing for efficient bug fixes, updates, and future feature additions. Documentation will support easy handoff and collaborative maintenance.

- **Security**: Security measures, including data encryption, secure authentication, and compliance with privacy regulations, will protect user information from unauthorized access and breaches, enhancing trust and user confidence.

- **Portability**: Developed using Flutter, the app will be portable across multiple platforms (iOS and Android) with consistent functionality, allowing users to switch between devices without disruption. Cross-platform compatibility will be verified with each update.

5. Appendices

5.1 Glossary

ZAI Task Suggestions: An intelligent system that recommends tasks based on user behavior, location, and time.
Geofencing: A virtual boundary that triggers specific actions or notifications when a user enters or exits a defined area.
Focus Mode: A feature that helps users minimize distractions by temporarily disabling notifications.

5.2 Data Models

Entity-Relationship diagrams or class diagrams, etc. (in progress)

5.3 Process Flowcharts

Describing any workflows or processes in the system. (in progress)

5.4 Other Supporting Information

Anything else that provides clarity or depth to the SRS. (in progress)