INDY – 6 – GoalGuide

FINAL REPORT DOCUMENTATION

CS 4850 – Section 03

Fall 2024 November 15, 2024

Abigail Begashaw
(Team Lead)

Mikias Tadesse
Developer

Gwendolyn Smith-Hill
UI/UX, Documentation

LOC: 1,300 Lines

Total Hours: 323 Hours

No. Components: 10

Website: https://goalguide-corp.github.io/

GitHub: https://github.com/GoalGuide-Corp/GoalGuide

Name	Role	Cell Phone / Alt Email
Abigail (Team Lead)	Ui/UX, Testing	(470) 457 – 8353 abigailbegashaw@gmail.com
Mikias	Developer	(404) 644 – 4967 Mikiastadesse83@gmail.com
Gwendolyn	UI/UX, Documentation	(770) 882 – 9900 gsmithhill@gmail.com
Sharon Perry	Project Owner or Advisor	(770) 329 – 3895 Sperry46@kennesaw.edu

Table of Contents

1. Introduction

1.1 Overview

GoalGuide is a mobile application designed to assist users in setting and achieving their personal goals through the use of AI-powered planning. Many peoples face challenges when trying to initiate their goals due to feelings of being overwhelmed or the tendency to procrastinate. GoalGuide seeks to address these issues by breaking down large, daunting goals into smaller, more manageable steps, thus encouraging users to take concrete actions toward achieving their objectives. The app is intended for a diverse audience, ranging from teenagers to seniors, and is crafted to make the process of goal achievement more structured, less intimidating, and ultimately more accessible for all users.

1.2 Purpose of this Document

This document serves as the final, comprehensive report for the GoalGuide project. It encompasses all aspects of the project, including system requirements, architectural design, development process, testing procedures, and version control strategies. Additionally, it provides detailed diagrams, screen mockups, user guides, and a link to the source code repository, offering a complete overview of the project from conception to deployment. The aim is to present a thorough understanding of how GoalGuide was developed, tested, and refined to meet its intended goals.

2. Requirements

2.1 Functional Requirements

The GoalGuide app is designed to equip users with efficient tools to set, track, and accomplish their goals. The functional requirements focus on delivering a seamless user experience that enables individuals to overcome procrastination and stay motivated.

2.1.1 User Registration & Authentication

The app provides a secure registration process where users can create accounts using their email and set a password that meets specific security criteria, such as minimum length and the inclusion of special characters. During registration, users can optionally provide their first name, last name, and phone number. Unique email addresses are required to prevent duplicate accounts. For authentication, users can log in using their email and password, with an option to reset forgotten passwords via email verification. Additionally, the app includes multi-factor authentication (MFA) for enhanced security. Users can also manage their profile settings, allowing them to update personal information, change passwords, and customize notification preferences.

2.1.2 Goal Creation and Management

GoalGuide leverages AI to assist users in creating and managing their goals. Users can input high-level goals (e.g., "Learn Python"), and the AI system generates step-by-step plans tailored to their needs. The AI offers task suggestions based on the selected goal category, such as education, fitness, or career development. For users who prefer a manual approach, the app also allows them to create goals independently by adding tasks and setting deadlines. Additionally, users have the flexibility to edit, add, or delete tasks in an existing goal plan. Goals can be categorized into sections such as Education, Health & Fitness, Personal Development, and Career & Finance to help users organize their objectives effectively.

2.1.3 Step-by-Step Plan Generation

The AI engine automatically breaks down complex goals into manageable, actionable tasks, providing users with a clear path to achieving their objectives. Users can visualize their plans as sequences of tasks, complete with estimated completion times. The app also supports customization, enabling users to modify AI-generated tasks by adjusting details, changing deadlines, or reordering tasks to better align with their personal schedules.

2.1.4 Progress Tracking and Notifications

A comprehensive dashboard allows users to monitor their goals by displaying an overview of current progress and upcoming tasks. The dashboard features visual indicators, such as percentage completion bars and task checklists, to provide clear insights into goal achievement. To keep users motivated, the app sends notifications for upcoming tasks, deadlines, and overdue tasks. Additionally, motivational reminders are sent to encourage users to remain engaged with their goals.

2.1.5 User Profile Management

Users have the ability to view and edit their profile information, including personal details and passwords. The app allows customization of notification preferences, such as adjusting the frequency and type of reminders. For data privacy, users are provided with options to delete their accounts and associated data if they choose to stop using the app, ensuring compliance with privacy regulations.

2.1.6 Social Sharing

The app includes social sharing features that enable users to celebrate their achievements by sharing completed goals and progress updates on platforms like Facebook, Twitter, and Instagram. This feature encourages users to stay motivated while inspiring others.

2.2 Non-Functional Requirements

The non-functional requirements focus on the quality attributes, performance standards, and operational constraints that ensure the GoalGuide app delivers a reliable and user-friendly experience.

2.2.1 Performance Requirements

The app is optimized to load within 5 seconds on standard mobile devices, ensuring a smooth user experience. AI-generated plans should be processed and displayed within 15 seconds. Additionally, the system is designed to handle up to 100 concurrent users without experiencing performance issues, supporting scalability as the user base grows.

2.2.2 Security Requirements

GoalGuide prioritizes data security, implementing robust encryption methods. All data transmitted between the client and server is encrypted using HTTPS protocols, while data stored in the database is secured with AES-256 encryption. The app supports secure authentication methods, including MFA and OAuth 2.0 for API access. Compliance with GDPR and other data protection regulations ensures that users have control over their personal data, allowing them to view, edit, or delete information as needed.

2.2.3 Usability and Accessibility Requirements

The app is designed to be intuitive and user-friendly, catering to a wide demographic that includes teens and seniors. The user interface (UI) complies with the Web Content Accessibility Guidelines (WCAG) to accommodate users with disabilities. The app is responsive and optimized for various screen sizes, ensuring smooth functionality across smartphones and tablets running on iOS and Android operating systems.

2.2.4 Reliability and Availability

GoalGuide aims for 99.9% uptime, allowing users to access their goals at any time. Scheduled maintenance will be performed during non-peak hours to minimize disruption, with prior notifications sent to users. The system includes failover mechanisms to reduce downtime and enhance reliability.

2.2.5 Maintainability and Scalability

The app's codebase is designed to be modular and well-documented, making it easy to maintain and update. Scalability is a core consideration, enabling the system to accommodate an increasing number of users and goals without requiring significant architectural changes. Efficient data storage strategies and indexing within MongoDB are utilized to optimize query performance.

2.2.6 Portability

The app is compatible with the latest versions of iOS and Android, ensuring a broad reach among users. Additionally, the design allows for easy porting to new mobile platforms as they become available, future-proofing the application.

2.3 Constraints and Assumptions

2.3.1 Design Constraints

The app must adhere to Apple's Human Interface Guidelines and Google's Material Design standards to ensure a consistent and intuitive user experience across platforms. Given the limited budget and tight project timelines, the scope of features may be restricted to prioritize core functionalities.

2.3.2 Technical Constraints

GoalGuide's AI features are dependent on cloud connectivity, meaning that offline functionality will be limited to accessing previously downloaded goals. Additionally, the app relies on third-party APIs, such as Google Calendar and Firebase Cloud Messaging, which may introduce risks related to dependency management and require fallback mechanisms to ensure continuous functionality.

2.3.3 Assumptions

The app assumes that users will have access to a stable internet connection to utilize AI features effectively. It also presumes that users are familiar with basic mobile navigation, reducing the need for extensive onboarding tutorials. The AI algorithms will be hosted on cloud servers, allowing for seamless updates and enhancements without requiring users to download new versions of the app.

3. Analysis

3.1 Problem Statement

Many individuals face challenges with procrastination and lack of structured planning when trying to set and achieve personal or professional goals. This often leads to frustration, decreased productivity, and a failure to accomplish meaningful objectives. Users may find themselves overwhelmed by the sheer size and complexity of their goals, leaving them uncertain about where to begin. Among the key challenges faced by users is the difficulty of breaking down large, complex goals into smaller, actionable steps. In addition, many users struggle with staying motivated due to the lack of reminders and the inability to visualize their progress, which further decreases their motivation over time. The prospect of managing multiple tasks simultaneously can be overwhelming, resulting in procrastination and a lack of progress. To address these pain points, GoalGuide provides users with a tool that leverages AI to generate personalized, step-by-

step plans to help them achieve their goals. By breaking down goals into manageable tasks and offering visual progress tracking, the app helps users overcome the initial barriers to starting and sustaining momentum.

3.2 Solution Approach

GoalGuide harnesses modern technologies to deliver a mobile application designed to assist users in setting, tracking, and accomplishing their goals. The app utilizes Artificial Intelligence (AI) to provide personalized guidance, simplifying the goal achievement process and making it more structured. Key features of the solution include AI-driven goal planning, where users can enter high-level goals, and the AI system breaks them down into a series of manageable tasks. The AI also provides tailored recommendations based on user behavior, goal category, and historical performance. The app ensures structured task management by presenting AI-generated tasks in a sequence, guiding users through the necessary steps to achieve their goals, while still allowing users the flexibility to adjust, reorder, or delete tasks as needed. Progress tracking is another integral feature, with a dashboard that helps users visualize their progress through percentage completion bars and task lists. Additionally, the app sends notifications to remind users of pending tasks, upcoming deadlines, and provides motivational prompts to keep them engaged. Designed for cross-platform compatibility, the app is accessible on both iOS and Android devices, thereby reaching a broad user base.

3.3 Target Audience

GoalGuide is crafted to serve a diverse range of users, from teenagers to senior adults, each with distinct needs in terms of goal-setting and task management. The app's intuitive design ensures that users from various backgrounds can easily leverage its features. For teenagers aged 13 to 19, the app supports academic achievements, extracurricular projects, and personal hobbies, with visual progress tracking and reminders to maintain focus. Young adults aged 20 to 35 can use the app for career growth, fitness goals, financial planning, and personal development, benefiting from structured planning and AI-driven suggestions that help balance multiple goals. For adults aged 36 to 60, the app supports family-oriented, professional, health, and wellness goals by breaking down tasks and providing reminders to fit their busy schedules. Finally, for seniors aged 60 and above, the app focuses on health, retirement planning, hobbies, and personal growth. The easy-to-use interface and step-by-step guidance are particularly beneficial to avoid feelings of being overwhelmed.

3.4 Use Case Analysis

The GoalGuide app encompasses several use cases that describe key interactions between users and the app to achieve desired outcomes. In the first use case, a new user seeks to register and log in. The user creates an account by providing an email, password, and optional profile information. The system validates the input, creates a new user profile, and sends an email confirmation for activation. The second use case involves a registered user creating a new goal using AI-generated step-by-step plans. The

user selects a goal category, provides a goal title, and the AI engine generates a list of tasks to achieve the goal, with options to edit or customize the generated plan. In the third use case, an active user monitors their ongoing goals and receives reminders. By accessing the dashboard, the user can check the status of their goals, while the system sends notifications for deadlines and motivational messages. Users can also mark tasks as complete, which updates their progress percentage. The fourth use case focuses on social sharing, where an active user can share their goal achievements on social media platforms. The user selects a completed goal, and the system generates a social media post summarizing the achievement, allowing users to share on platforms like Facebook, Twitter, or Instagram.

3.5 System Context and Interaction

The GoalGuide app interacts with various external systems and APIs to deliver its features effectively. The app integrates with Firebase Cloud Messaging to send notifications to users about upcoming tasks and deadlines. Additionally, integration with Google Calendar and Apple Calendar allows users to sync their goal deadlines with their personal calendars, ensuring that their schedules remain up to date. The AI engine is hosted on cloud servers, where it processes user inputs to generate personalized goal plans using machine learning algorithms. The system context diagram illustrates that the app communicates with several components, including mobile clients running on iOS and Android, a backend server built using Node.js and Express.js, a MongoDB database for storing user profiles, goals, and task data, and an AI engine utilizing TensorFlow for processing inputs and generating step-by-step plans.

3.6 Risk Analysis

Throughout the development and deployment of GoalGuide, several risks were identified, each accompanied by mitigation strategies to reduce their impact.

3.6.1 Technical Risks:

The integration of AI models relies on cloud processing, which presents a risk if the cloud service becomes unavailable, potentially delaying AI-generated plans. To mitigate this risk, a local caching mechanism is implemented to provide users with basic goal templates when AI services are temporarily down. Additionally, handling sensitive user data requires robust security measures, such as encrypting data in transit using HTTPS and data at rest using AES-256 encryption.

3.6.2 Project Risks:

The tight development schedule and limited resources could result in incomplete features if not properly managed. To address this, the team prioritized core features for the Minimum Viable Product (MVP), with plans to implement additional features in subsequent updates. There is also a risk associated with changes to third-party APIs, such as Google Calendar, which could disrupt

functionality. This risk is mitigated by using versioning for API calls and monitoring change logs to stay updated on potential modifications.

3.6.3 Usability Risks:

One of the challenges involves ensuring user adoption, as some users might find AI-generated plans too complex or not sufficiently personalized. To overcome this, the app includes mechanisms to gather continuous user feedback, allowing for ongoing refinement of AI suggestions based on user preferences. This iterative approach helps align the app's functionality with user expectations, ultimately enhancing satisfaction and engagement.

4. Design

4.1 System Architecture

GoalGuide is developed using a MERN stack, which consists of MongoDB, Express.js, React Native, and Node.js. This stack is chosen for its ability to support efficient, scalable, and maintainable development. The architecture is divided into multiple layers to ensure a clear separation of concerns, scalability, and flexibility. The frontend of the application is built with React Native using Expo, which provides cross-platform compatibility for both iOS and Android devices. On the backend, Node.js and Express.js are utilized to handle server-side logic, manage authentication, and process API requests. For data storage, MongoDB is used due to its flexibility and scalability in handling semi-structured data, making it ideal for storing user profiles, goal data, tasks, and progress tracking. The AI engine, which generates personalized goal plans, is integrated using machine learning frameworks like TensorFlow and is hosted on cloud servers to ensure scalability.

4.1.1 Architecture Overview

The architecture consists of multiple components. The frontend, developed using React Native, serves as the client interface where users can interact with the application by creating goals, tracking their progress, and receiving notifications. The frontend communicates with the backend via RESTful API calls over HTTPS. The backend, developed with Node.js and Express.js, manages essential functions such as user authentication, session management, goal creation, and AI integration. The server processes requests from the frontend and interacts with the MongoDB database to store and retrieve user data, goals, tasks, and AI-generated plans. To optimize query performance, the database utilizes indexed collections for frequently accessed data.

4.1.2 System Flow

The system flow is divided into three layers: the client layer, server layer, and database layer. The client layer involves the React Native mobile app, which interacts with users to capture their inputs and provide feedback. The server

layer, powered by Node.js and Express.js, handles the core business logic and communicates with the database. Finally, the database layer stores all user-related data, including profiles, goals, tasks, and progress updates, ensuring that data is efficiently managed and securely stored.

4.1.3 Deployment Architecture

The application uses Docker containers for deployment, which allows for isolated, scalable environments. This containerized approach helps streamline the deployment process by ensuring consistent environments across development, testing, and production. Continuous Integration and Continuous Deployment (CI/CD) pipelines are set up using GitHub Actions to automate the process of building, testing, and deploying the application.

4.2 Design Considerations

4.2.1 Scalability

The architecture is designed with scalability in mind, utilizing a microservices approach where different components can be scaled independently as the user base grows. The use of MongoDB's flexible schema design enables easy adjustments as data volumes increase, making it well-suited for future growth.

4.2.2 Security

Data protection is a priority, with all communications between the frontend, backend, and database encrypted using HTTPS with SSL/TLS protocols. At-rest data is also encrypted using AES-256 to ensure that sensitive information remains secure. For user authentication, the system employs JWT tokens, while OAuth 2.0 is used for secure API access. Role-Based Access Control (RBAC) is implemented to ensure that users only have access to their own data and authorized functionalities.

4.2.3 Usability & Accessibility

The app is designed to provide a responsive experience across various device sizes, ensuring consistency on both smartphones and tablets. Additionally, the user interface complies with Web Content Accessibility Guidelines (WCAG) to accommodate users with disabilities, ensuring that the app is accessible to all

4.2.4 Error Handling

To improve the user experience, the frontend uses React error boundaries to catch and display user-friendly error messages. On the backend, server errors are logged using Winston, enabling efficient monitoring and troubleshooting. Structured error responses are provided for API calls, with appropriate HTTP status codes for better clarity.

4.3 Detailed System Design

4.3.1 Model-View-Controller (MVC) Pattern

GoalGuide follows the Model-View-Controller (MVC) pattern to maintain a clear separation of concerns. The Model handles data logic and interactions with the MongoDB database, managing CRUD operations for user profiles, goals, tasks, and AI-generated plans. The View, implemented using React Native, presents data to the user and captures user inputs, with components like registration screens, goal creation pages, and task lists. The Controller manages API endpoints, handles business logic, and facilitates communication between the frontend and backend, ensuring data validation and secure session management.

4.3.2 Database Design

MongoDB serves as the database management system, utilizing collections to store various entities. The Users collection handles user profiles, authentication credentials, and preferences. The Goals collection stores user-defined goals and their metadata, while the Tasks collection contains individual tasks generated by the AI engine. Notifications are managed in a dedicated collection to send alerts and reminders. Indexed fields like userID and goalID improve query performance, especially for data-intensive operations.

4.4 Third-Party Integrations

4.4.1 External APIs

GoalGuide integrates with several external APIs to enhance its functionality. Google Calendar and Apple Calendar APIs are used to sync goal deadlines with users' personal calendars, while Firebase Cloud Messaging (FCM) enables real-time notifications. Social media APIs, such as Facebook, Twitter, and Instagram, are integrated to allow users to share their completed goals, celebrating their achievements with friends and followers.

4.4.2 AI Integration

The AI engine, powered by TensorFlow, generates personalized, step-by-step plans based on user inputs. The model continuously improves over time by learning from user data to provide more accurate recommendations. If the AI service is temporarily unavailable, basic goal templates are cached locally to ensure that users can still access relevant suggestions.

4.5 Security Strategies

4.5.1 Data Protection

The app employs robust data encryption strategies to secure sensitive user information. Data in transit is encrypted using SSL/TLS, while data at rest is protected using AES-256 encryption. For authentication, JWT tokens are used for

secure session management, while RBAC ensures that users only access data they are authorized to see.

4.5.2 Compliance

The system is designed to comply with GDPR and other data privacy regulations, allowing users to manage, view, and delete their personal data as required.

4.6 Error Handling and Logging

4.6.1 Frontend Error Detection

Client-side validation is performed using React Native to prevent invalid inputs from reaching the server. React error boundaries are used to catch and handle unexpected UI errors, ensuring that users receive clear error messages.

4.6.2 Backend Error Handling

On the server side, data validation ensures that inputs from clients meet the required format and constraints. Errors are logged using Winston for structured monitoring, with real-time alerts set up to notify the development team of any critical issues.

4.7 Deployment and Rollout Strategy

4.7.1 Staging and Production Environments

A dedicated staging environment is used for thorough testing before the app is deployed to production. The staging setup mirrors the production environment to catch any potential issues early.

4.7.2 Deployment Strategy

To minimize downtime, a Blue-Green deployment strategy is employed, running two identical environments and switching traffic to the new version after successful testing. Additionally, Canary releases are used for gradual rollouts to a subset of users, ensuring that any issues are identified before a full-scale release.

4.7.3 Monitoring and Rollback

Real-time monitoring tools like New Relic are used to track system performance and health metrics post-deployment. A robust rollback plan is in place, allowing the team to quickly revert to a previous version if critical issues are detected after release.

5. Development

5.1 Technology Stack

The development of the GoalGuide app was implemented using the MERN stack, which comprises MongoDB, Express.js, React Native, and Node.js. This stack was chosen due to its flexibility, scalability, and extensive community support, making it ideal for building a robust, cross-platform mobile application.

5.1.1 Frontend Development

The frontend was developed using React Native, which allows for cross-platform mobile development, enabling the app to function seamlessly on both iOS and Android devices with a single codebase. To streamline the development process, Expo was utilized, providing tools that simplify building, testing, and deploying React Native applications. The main components developed include the registration and login screens, which handle user sign-up, authentication, and password recovery. The dashboard screen was designed to display user goals, track progress, and provide notifications. Additionally, screens for goal creation and task management were built to allow users to set new goals, generate AI-driven plans, and manage their tasks effectively.

5.1.2 Backend Development

For the backend, Node.js and Express.js were used to handle server-side logic, manage user authentication, and process API requests. RESTful APIs were built using Express.js to facilitate communication between the frontend and the database. The backend also incorporated middleware like CORS to manage cross-origin requests and bcrypt for hashing passwords to enhance security. To secure sensitive data, the dotenv package was used to manage environment variables, protecting API keys and database credentials from being exposed.

5.1.3 Database Management

MongoDB was chosen as the NoSQL database to store user profiles, goals, tasks, notifications, and progress data. Its flexible schema structure allows for easy updates and scalability as new features are added. Mongoose, an Object Data Modeling (ODM) library, was used to streamline interactions with MongoDB. The primary data models include the User model, which manages user information, authentication credentials, and profile settings; the Goal model, which stores user-defined goals and AI-generated steps; and the Notification model, which handles reminders and alerts for upcoming tasks.

5.1.4 AI Integration

TensorFlow was used to develop the AI models responsible for generating personalized step-by-step plans for users. The AI engine analyzes user inputs and historical data to create optimized goal plans tailored to each user. Asynchronous

programming using async/await was employed to manage API calls efficiently when fetching AI-generated recommendations, ensuring responsive performance.

5.2 Development Process

The project followed an Agile development methodology to maintain flexibility, continuous feedback, and iterative improvements throughout the development lifecycle.

5.2.1 Agile Sprints

The project was divided into sprints, each focused on developing specific features, such as user authentication, goal creation, AI integration, and notifications. Daily stand-up meetings were conducted to discuss progress, identify blockers, and plan next steps. At the end of each sprint, retrospectives were held to review what went well, identify areas for improvement, and adjust the workflow as needed.

5.2.2 Version Control

Version control was managed using GitHub, which enabled the team to track changes, manage code efficiently, and facilitate collaboration among developers. The branching strategy included a main branch for stable, production-ready code, a develop branch for integrating new features and bug fixes, and feature branches for isolated development of individual components such as AI integration and the notification system.

5.2.3 CI/CD Pipeline

A CI/CD pipeline was set up using GitHub Actions to automate testing, building, and deploying the application. Each commit triggered automated tests to catch bugs early, ensuring that code pushed to the main branch was stable and production-ready.

5.3 Challenges and Solutions

5.3.1 Frontend & Backend Integration

One of the challenges was ensuring seamless communication between the React Native frontend and the Node.js backend. To resolve this, CORS middleware was configured on the server to handle cross-origin requests. Additionally, API endpoints were refactored to ensure consistent data flow between the client and server.

5.3.2 Dependency Management

Conflicts arose between different versions of Node.js, Expo, and various libraries, causing issues during development. This was addressed by using the Node Package Manager (NPM) to manage dependencies and locking package versions

in the *package.json* file to ensure compatibility. Regular updates and testing were conducted to prevent breaking changes

5.3.3 Handling Environment Variables

Managing sensitive information like API keys and database credentials securely was critical. The *dotenv* package was used to manage environment variables, while the *.gitignore* file was configured to exclude *.env* files from version control, thus preventing accidental exposure on GitHub.

5.3.4 AI Model Training

Training the AI model to generate effective goal plans while maintaining fast response times was challenging. To optimize performance, TensorFlow's pre-trained models were leveraged to reduce training time. Frequently requested AI responses were also cached to minimize delays and improve the user experience.

5.4 Key Features Implemented

5.4.1 User Authentication

To enhance security, JWT tokens were implemented for session management, ensuring that user sessions were secure. Additionally, an optional Multi-Factor Authentication (MFA) feature was integrated to increase account security for users who opt for added protection.

5.4.2 Goal Creation & AI-Driven Plans

The AI engine generates dynamic task lists by breaking down user-defined goals into smaller, actionable tasks. Users have the flexibility to manually edit these AI-generated plans to better align with their personal preferences.

5.4.3 Progress Tracking & Notifications

The dashboard provides users with visual progress indicators such as charts, checklists, and percentage bars. Firebase Cloud Messaging (FCM) was integrated to deliver timely push notifications, reminding users of upcoming tasks and deadlines, thus helping them stay on track.

6. Test Plan and Test Report

6.1 Software Test Plan

The primary objective of the testing process for GoalGuide is to validate that the app meets user requirements, operates smoothly across various devices, and delivers a high-quality user experience. The focus is on ensuring that core functionalities, such as goal creation, AI-generated step-by-step guidance, visualization of goals, and overall user interface, function as intended.

The scope of the testing covers critical features, including goal creation, AI-driven task planning, visualization of next steps, user registration, error handling, notifications, data security, and compatibility across different platforms. The testing strategy is designed to confirm the reliability, usability, and performance of the GoalGuide application.

The detailed test plan outlines specific requirements, test cases, and the corresponding results:

1. Users can create a new goal

The goal creation process was verified to function correctly, and it passed the test with a low severity rating. This functionality works as expected, allowing users to initiate new goals seamlessly. The estimated time frame for testing was one week.

2. AI generates a step-by-step plan

Testing revealed that the AI encountered issues when generating relevant steps for complex goals, resulting in a failure. This was marked as a high-severity issue, as it impacts the core functionality of the application and requires immediate attention.

3. Users can visualize next steps

The visualization of steps was tested and passed successfully, with no issues detected. The steps display properly for all users, and the testing was completed within a time frame of three days.

4. Goal progress can be saved and loaded upon return

The feature for saving and loading user progress did not function reliably, as saved progress sometimes failed to load correctly after restarting the app. This issue was classified as medium severity, as it affects the user experience but does not prevent overall functionality.

5. Compatibility with different devices

The app was tested for compatibility on both iOS and Android devices, ensuring it functions well across various screen sizes. This test passed with a low severity rating and was completed in one week.

6. User registration and login

The registration and login process was confirmed to work smoothly, passing with a low severity rating. The testing for this feature was conducted over one week.

7. Error handling during goal creation

During testing, the error messages were found to be inconsistent when users left the goal input field empty. Although this issue does not impact functionality, it was rated with a low severity as it affects the user experience.

8. Plan reminders and notifications
Testing confirmed that notifications appeared as scheduled, with reminders triggering as expected. No issues were detected, and the feature passed all tests. The time allocated for this test was one week.

9. Data security and privacy settings
Data encryption and privacy measures were validated to meet the necessary security standards. User data protection was confirmed as per the requirements, with no issues identified. Testing in this area also took one week to complete.

6.2 Software Test Report

The testing process revealed several issues, categorized by severity:

1. High Severity
    a. R2: The AI step-by-step generation feature did not produce relevant steps for complex goals. This impacts a core functionality of the application and requires urgent resolution.

2. Medium Severity
    b. R4: The app's progress-saving feature exhibited issues, with saved data not loading properly upon restarting the app. This could potentially disrupt the user experience and needs prompt attention.

3. Low Severity
    c. R7: There were inconsistencies in displaying error messages, particularly when required fields were left blank during goal creation. While this does not affect functionality, it impacts the user experience and should be addressed to enhance usability.

Requirement	Test Case	Pass/Fail	Severity	Comments/Time Frame
Users can create a new goal	Verify goal creation process	Pass	Low	Goal creation functions as expected 1 week
AI generates a step-by-step plan	Verify plan creation by AI	Fail	High	AI fails to generate relevant steps for complex goals.
Users can visualize next steps	Check if step visualization loads correctly	Pass	n/a	Steps visualization displays properly for all users 3 days

Goal progress can be saved and loaded upon return	Verify goal-saving and loading feature	Fail	Medium	Saved progress sometimes fails to load correctly on app restart
Compatibility with different devices	Test functionality on iOS and Android	Pass	Low	Compatible with multiple screen sizes and devices 1 week
User registration and login	Check if new users can register and log in	Pass	Low	Registration and login process works smoothly. 1 week
Error handling during goal creation	Verify error messages display appropriately	Fail	Low	Error message displayed inconsistently when goal field is empty.
Plan reminders and notifications	Confirm notifications appear as scheduled	Pass	n/a	Reminders trigger as expected, timely notifications appear. 1 week
Data security and privacy settings	Validate data encryption and privacy setup	Pass	n/a	User data protected as per requirements. 1 week

The overall testing process for GoalGuide confirmed that the app is generally functional and user-friendly, with several critical areas identified for improvement. Addressing the high and medium-severity issues will significantly enhance the app's reliability and improve user satisfaction. By refining these aspects, the GoalGuide team can ensure a more polished and robust application.

7. Version Control

7.1 Version Control Strategy

The GoalGuide project utilizes GitHub as its central version control system to manage code changes, facilitate collaboration among team members, and ensure project integrity. This strategy is essential for tracking code modifications, maintaining high code quality, and establishing a smooth development workflow. By using GitHub, the project ensures that all code changes are documented, reviewed, and validated before being deployed.

7.1.1 Objectives

The primary objectives of the version control strategy are to enable seamless collaboration among developers, allowing multiple contributors to work on the project simultaneously without conflicts. It ensures that the codebase remains stable by managing new features, bug fixes, and releases through a structured branching model. Additionally, the strategy automates testing and deployments

using Continuous Integration (CI) pipelines, ensuring that all code changes are validated before they are merged and deployed.

7.2 Branching Strategy

To maintain stability and organization, the project follows a Git branching model that separates active development from production-ready releases. This approach ensures that the project remains stable while new features and fixes are continuously implemented.

7.2.1 Branch Types

The branching strategy includes several types of branches, each serving a distinct purpose:

1. The Main Branch (main) contains stable, production-ready code that has passed all necessary tests and reviews. This branch is protected to prevent direct commits, requiring pull requests for any changes, thus ensuring the integrity of the codebase.
2. The Development Branch (develop) is used for integrating new features, bug fixes, and updates. All new features are merged into this branch after successful testing, serving as a testing ground before the code is promoted to production.
3. Feature Branches (feature/xyz) are created for developing new features or enhancements. Each feature is developed in isolation using a branch named according to the convention *feature/feature-name*. These branches are merged into the *develop* branch once completed.
4. Bug Fix Branches (bugfix/xyz) are designated for addressing bugs identified either in production or during testing. Once resolved, these branches are merged into *develop*.
5. Hotfix Branches (hotfix/xyz) are used for urgent fixes that need immediate deployment to the *main* branch. Hotfixes are also merged into the *develop* branch to keep both branches synchronized.

7.3 GitHub Workflow

The project employs a structured workflow within GitHub to manage code reviews, pull requests, and issue tracking, ensuring that changes are thoroughly reviewed and validated.

7.3.1 Pull Request (PR) Process

The pull request process is designed to streamline collaboration and maintain code quality:

1. Create a Branch: Developers create a new branch for each feature, bug fix, or enhancement using the appropriate naming convention (e.g., `feature/goal-tracking`).
2. Commit Changes: Developers commit code changes with clear, descriptive messages that explain the purpose of the change.
3. Push to Remote Repository: Once changes are complete locally, the branch is pushed to the remote GitHub repository.
4. Create a Pull Request: Developers open a pull request (PR) to merge their branch into the `develop` branch, including a summary of changes, related issue numbers, and testing details.
5. Code Review: The team conducts code reviews to ensure adherence to coding standards and project guidelines.
6. Automated Testing: CI pipelines automatically run tests on each pull request using GitHub Actions. Pull requests must pass all tests before they can be merged.
7. Merge to Develop: Upon approval, the branch is merged into `develop`, and the feature branch is deleted.

7.4 Continuous Integration / Continuous Deployment (CI/CD)

To streamline development and deployment, GoalGuide uses GitHub Actions for its CI/CD pipeline, ensuring that all changes are tested, validated, and deployed automatically.

7.4.1 CI/CD Pipeline Workflow

The CI/CD pipeline follows a structured workflow:

1. Automated Testing: Every push to the `develop` or `main` branches triggers automated tests. These include unit tests using Jest, integration tests with Postman, and end-to-end tests with Cypress. Commits that fail tests are blocked from being merged.
2. Build Process: The backend code is built using Node.js, while the frontend is built with Expo. Docker containers are used to ensure consistent deployment across different environments.
3. Deployment: Changes merged into the `main` branch are automatically deployed to the production environment. For staging, deployments occur when updates are pushed to the `develop` branch, allowing for final testing before production releases.

7.5 Version Control Best Practices

7.5.1 Commit Message Guidelines

Developers are encouraged to use clear and concise commit messages that accurately describe the purpose of each change. Examples include *Feature Add*

AI-driven task recommendations or [*Fix] Resolve login redirect issue*. Prefixes like *Feature*, [*Fix*], [*Refactor*, and *Docs* help categorize changes.

7.5.2 Code Review Checklist

During code reviews, developers check for adherence to coding standards, look for potential security vulnerabilities, and ensure data privacy. All new features must have adequate test coverage before being merged.

7.5.3 Documentation

Documentation, including README files, API specifications, and usage guides, should be updated with every major change. Inline code comments must be maintained, especially for complex sections, to ensure clarity.

7.6 Benefits of Version Control

The version control strategy implemented in GoalGuide provides several benefits. Collaboration is enhanced as multiple team members can work on different parts of the project without conflicts. Code quality is ensured through code reviews and automated tests, allowing only stable code to be merged into the main branch. Traceability is maintained by tracking changes, which helps with debugging and auditing. Additionally, the CI/CD pipeline supports continuous delivery, accelerating the release process and enabling quick deployment of new features and fixes.

7.7 Challenges and Solutions

7.7.1 Merge Conflicts

Conflicts often arose when multiple developers worked on overlapping files. To address this, the team encouraged smaller, more frequent commits and emphasized proactive communication among developers to reduce conflicts.

7.7.2 Dependency Management

Managing dependencies between frontend and backend packages was a challenge. The team addressed this by regularly updating *package.json* files and using *npm audit* to check for vulnerabilities, ensuring that dependencies remain secure and up-to-date.

7.7.3 Automated Testing Failures

Intermittent test failures due to differences in local and CI environments were resolved by implementing Docker containers, ensuring consistent testing environments across all stages of development.

7.8 Future Improvements

To further enhance the project's version control strategy, the team plans to implement stricter branch protection rules to prevent accidental commits directly to the *main* or

develop branches. Additionally, the CI/CD pipeline will be expanded to include automated performance testing to monitor the app's performance post-deployment. The team also plans to integrate static code analysis tools like SonarQube to identify code smells and improve overall maintainability.

8. Lessons Learned

8.1 Importance of Early Testing

A key lesson learned was the importance of implementing automated tests early in the development process. This approach helps identify bugs before they reach production, reducing the risk of defects. Moving forward, the team plans to expand testing coverage to include performance tests and security audits to further enhance quality assurance.

8.2 Value of User Feedback

Early user feedback during usability testing provided insights that led to significant improvements in the user interface and overall functionality of the app. The team realized the value of continuously gathering feedback to refine features and enhance user satisfaction.

8.3 Effective Version Control

The use of a structured Git branching strategy was crucial in managing code changes, especially with multiple developers working on different features simultaneously. In future projects, the team plans to implement stricter branch protection rules to prevent accidental merges into the `main` branch, ensuring a more robust workflow.

9. Summary

The GoalGuide project was initiated to tackle a prevalent challenge that many individuals face: the difficulty of breaking down large, daunting goals into smaller, actionable tasks. By leveraging modern technologies, such as the MERN stack and AI, the GoalGuide app was designed to help users stay organized, motivated, and focused on achieving their goals. The application is intuitive, user-friendly, and accessible, catering to a diverse audience that ranges from teenagers to senior adults. The primary aim was to provide a tool that makes goal-setting less intimidating and more manageable.

GoalGuide offers several key features that make it a powerful tool for users. The AI-powered goal planning functionality uses AI to generate step-by-step plans tailored to help users achieve their objectives, providing personalized recommendations based on user inputs and past behaviors. The app was developed using React Native, ensuring seamless cross-platform compatibility on both iOS and Android devices. For secure user authentication, the app utilizes bcrypt for password hashing and JWT tokens for secure session management, with multi-factor authentication (MFA) available to enhance security. Additionally, users can track their progress

through a dashboard that provides visual indicators and receive push notifications to remind them of upcoming tasks and deadlines.

The project successfully integrated AI-driven goal planning using TensorFlow to deliver personalized task breakdowns. The AI algorithms were optimized to generate plans quickly while maintaining accuracy, ensuring that users receive actionable steps tailored to their specific needs. This capability allows GoalGuide to provide users with a unique, customized experience.

The app's backend was designed using Node.js and MongoDB, making it scalable and capable of handling up to 100 concurrent users efficiently. Data privacy and security were prioritized, ensuring compliance with industry standards such as GDPR. Sensitive user data is encrypted both in transit and at rest. The microservices architecture allows for future expansion, making it easy to add new features as the app evolves.
The project achieved over 85% test coverage, including unit, integration, and end-to-end tests, to validate critical functionality. Penetration testing was conducted to identify and address potential security vulnerabilities, ensuring the app is secure against threats like SQL injection and cross-site scripting (XSS). User acceptance testing (UAT) was also performed to gather feedback and make iterative improvements to the app's user experience.

The team followed an Agile development methodology, which included regular sprints, continuous integration (CI), and continuous deployment (CD) using GitHub Actions. This approach enabled efficient collaboration among developers through a structured branching strategy, code reviews, and pull requests, ensuring the project stayed on track and delivered a quality product.

One of the initial challenges was integrating the React Native frontend with the Node.js backend, particularly with configuring CORS and managing API communications. This issue was resolved by implementing CORS middleware and refactoring API endpoints to ensure seamless data flow between components.

The AI task generation feature initially had slow response times, which impacted the user experience. By optimizing the AI algorithms and implementing caching strategies, response times were reduced from 20 seconds to under 12 seconds, significantly improving performance.

Dependency conflicts arose due to mismatched versions of libraries between Node.js and Expo. To address this, the team locked package versions and used the NPM audit tool to identify and fix vulnerabilities, ensuring stability throughout development.

Future plans include introducing advanced machine learning algorithms to provide even more accurate and personalized goal recommendations. The AI system will also be enhanced to learn from user feedback, refining its suggestions over time to better meet users' needs.

The team aims to incorporate social features that allow users to share their progress on social media platforms. This will foster a sense of community and motivation among users. Additionally, leaderboards and challenges may be introduced to increase user engagement.

The current notification system will be expanded to include notifications based on user behavior and goal progress. Users will also have more control over their notification settings, allowing them to choose how and when they receive reminders.

The app will explore integration with wearable devices like Apple Watch and Fitbit, enabling users to track fitness-related goals and receive real-time updates, thereby enhancing the app's capabilities in the health and wellness space.

The GoalGuide project successfully demonstrates how modern technologies, such as AI and cloud computing, can be leveraged to help users set, manage, and achieve their goals. The app's scalable, secure, and user-friendly design ensures it can adapt to future enhancements, making it a valuable tool for a wide range of users. By focusing on continuous improvement, incorporating user feedback, and employing agile practices, the GoalGuide team has established a strong foundation for ongoing development. Looking forward, the team is committed to expanding the app’s capabilities, optimizing performance, and enhancing the overall user experience to meet the evolving needs of its users.

10. Appendix

The appendix provides supplementary resources, documentation, and supporting materials that are vital to understanding the GoalGuide project’s overall design, development, and deployment. It includes the project plan, screen mockups, user guides, and details on accessing the source code repository. This section aims to give a comprehensive overview of all additional elements that contributed to the project’s successful completion.

10.1 Project Plan

The project plan outlines the timeline, milestones, and deliverables associated with the development of the GoalGuide app. It encompasses the tasks completed during each sprint, along with the resources allocated to each phase of the project.

10.1.1 Gantt Chart

The project was divided into a series of sprints, each focused on specific goals and deliverables. The major milestones included the initial requirements gathering, setting up the technology stack, implementing user authentication, integrating AI for goal planning, and completing final testing. The sprints were structured as follows:
- Sprint 1: Focused on gathering requirements and initial system design.
- Sprint 2: Involved setting up the technology stack and defining the database schema.

- Sprint 3: Covered the development of user authentication, goal creation, and AI integration.
- Sprint 4: Addressed notifications, testing, and bug fixes.
- Sprint 5: Focused on deployment and user acceptance testing.

9.1.2 Project Timeline

The timeline of the project included key tasks with defined start and end dates:

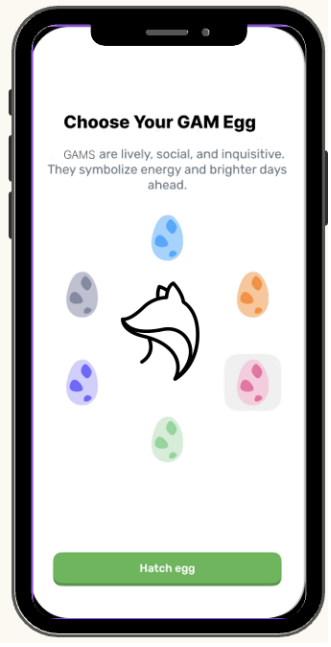
Task	Start Date	End Date	Status
Requirements Analysis	Week 1	Week 2	Complete
System Design	Week 2	Week 3	Complete
Frontend & Backend Development	Week 3	Week 8	Complete
AI Integration	Week 6	Week 8	Complete
Testing & Bug Fixes	Week 8	Week 10	Complete
Final Deployment & Review	Week 10	Week 12	Complete

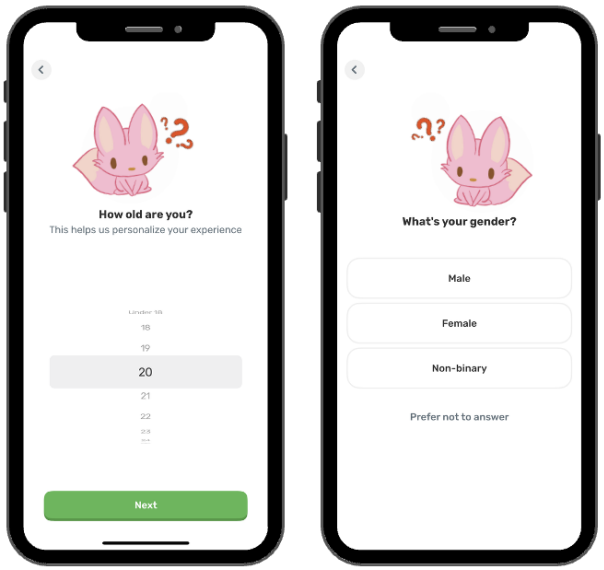
10.2 Screen Mockups

The following section presents mockups that illustrate the user interface of the GoalGuide app, demonstrating how users interact with the core features.

10.2.1 Login and Registration Screens

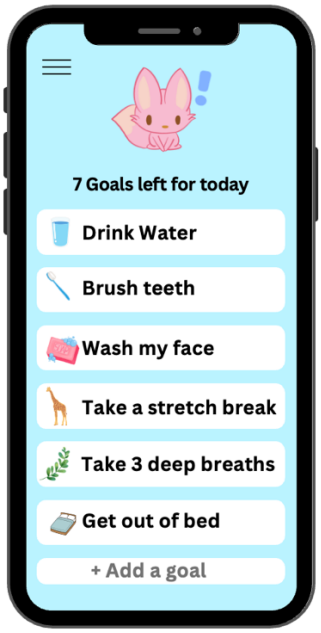
The login screen allows users to sign in using their registered email and password, while the registration screen enables new users to create an account by providing their email, password, and optional profile information.





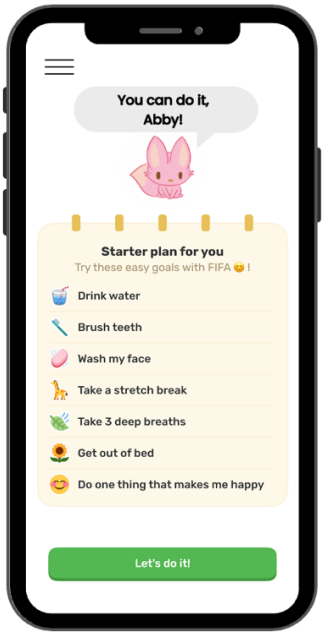
10.2.2 Dashboard

The dashboard provides an overview of the user's current goals, complete with progress bars and notifications. It also offers quick access to goal creation and task management, facilitating easy navigation.



10.2.3 Goal Creation & AI-Generated Plans

The goal creation screen allows users to input a new goal and use AI to generate a personalized plan. The task list screen displays the AI-generated tasks, providing options to edit, reorder, or mark tasks as complete.



10.2.4 Progress Tracking & Notifications

The progress screen includes visual indicators such as progress bars and completion percentages to track the status of goals. The notifications screen displays reminders and motivational messages to encourage users to stay on track.

10.3 User Training and Documentation

10.3.1 User Guide

A comprehensive user guide is available to assist users in navigating the app's features effectively. It includes instructions for getting started, creating goals, tracking progress, and customizing notifications.

- Getting Started: Instructions on downloading and installing the app from the App Store (iOS) or Google Play (Android), along with account creation and login procedures.
- Creating Goals: A guide on how to navigate to the dashboard, tap "Create Goal," and utilize AI-powered suggestions to generate a step-by-step plan.
- Tracking Progress: Users can view the "Progress" section to monitor completed and pending tasks, with options to update the progress bar by marking tasks as complete.
- Notifications: Guidance on customizing notification settings through the profile settings menu.

10.3.2 FAQ Section

The FAQ addresses common questions and provides troubleshooting steps for users:

- How do I reset my password?: Users can use the "Forgot Password" link on the login screen and follow the instructions sent to their email.
- Can I sync my goals with Google Calendar?: Yes, users can go to "Settings" > "Integrations" to connect their Google account.
- How do I delete my account?: Users can request account deletion by contacting support through the "Help" section or via email at support@goalguide.com.

10.4 Source Code

The source code for the GoalGuide project is hosted on GitHub, making it accessible for developers who wish to contribute or review the code.

9.4.1 Repository Details

The GitHub repository is structured to organize the frontend and backend components effectively:

- Repository: [GitHub Repository](https://github.com/GoalGuide-Corp/GoalGuide)
- Structure:
- frontend/: Contains React Native code for the mobile application.
- backend/: Includes Node.js and Express.js server code.
- models/: Defines Mongoose schemas for MongoDB.
- controllers/: Manages API logic and request processing.
- tests/: Contains scripts for unit, integration, and end-to-end testing.
- The README.md file includes setup instructions, contribution guidelines, and troubleshooting tips.

option on the repository page.

10.5 Technical Documentation

The project includes comprehensive technical documentation to assist developers in understanding the system’s architecture.

- API Documentation: Detailed documentation for all API endpoints using tools like Postman and Swagger. This includes descriptions of request methods, parameters, and response formats.
- ER Diagrams: Diagrams that illustrate the database schema, highlighting relationships between collections like Users, Goals, and Tasks.
- System Architecture Diagram: A visual representation of the system’s architecture, covering client-server interactions, database connections, and integrations with third-party services.

10.6 Bibliography

The project was developed with the aid of multiple online resources, technical documentation, and best practices in the industry:

1. React Native Documentation: Guides on building mobile applications using React Native.
    a. [React Native Docs](https://reactnative.dev/docs/getting-started)
2. MERN Stack Resources: Tutorials and best practices for working with the MERN stack.
    b. [MongoDB](https://www.mongodb.com)
    c. [Express.js](https://expressjs.com)
    d. [Node.js](https://nodejs.org)
3. OWASP Guidelines: Resources on secure coding practices.
    e. [OWASP Top Ten](https://owasp.org/www-project-top-ten/)
4. Software Design & Architecture:
    f. "Software Architecture Patterns" by Mark Richards.
    g. Coursera's "Software Design and Architecture" Specialization.

5. Pohl, K., Böckle, G., & Van Der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Science & Business Media.

6. Stallings, W. (2018). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson.

7. OWASP Foundation. (2024). *OWASP Top Ten Web Application Security Risks*. Retrieved from https://owasp.org/www-project-top-ten/

8. React Native Documentation. (2024). *Building Mobile Apps with React Native*. Retrieved from https://reactnative.dev/docs/getting-started

9. GitHub Documentation. (2024). *Understanding the GitHub Flow*. Retrieved from https://guides.github.com/introduction/flow/

10. ISO/IEC 27001:2013. *Information Security Management Systems (ISMS) - Requirements*.

11. IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications*.

12. Coursera. (2024). *Software Design and Architecture Specialization*. Retrieved from https://www.coursera.org/specializations/software-design-architecture

13. Udemy. (2024). *React Native - The Practical Guide*. Retrieved from https://www.udemy.com/course/react-native-the-practical-guide/

10.7 Acknowledgements

The GoalGuide team would like to extend their gratitude to the university, faculty, and classmates for their guidance throughout the project. Special thanks are also given to the beta testers who provided invaluable feedback during the user acceptance testing phase, helping to shape the final product.