**CEN 4010 Principles of Software Engineering**

**Spring 2023**

**Milestone 3: More Detailed Requirements, Architecture and a Vertical Software Prototype**

RememberAll

Team 6

March 31, 2023

Group Members Email address

Kyle Palmer - kylepalmer2021@fau.edu

Abby Denton - adenton2016@fau.edu

Juan Rivera - Jriveraa102@gmail.com

Christopher Rodriguez - rodriguezc2020@fau.edu

|  |  |  |
| --- | --- | --- |
| Revision History | | |
| **Name** | **Revision Dates** | **Revision Description in Github** |
| Kyle Palmer | 03/27/2023  03/31/2023 | Update ReadMe  Update ReadMe  Added Files via upload |
| Abby Denton | 03/26/2023  03/31/2023 | Update ReadMe  Added Files via upload |
| Juan Rivera | 03/28/2023  03/31/2023 | Update ReadMe  Added Files Via upload |
| Christopher Rodriguez | 03/25/2023  03/25/2023 | Documentation Created  Update ReadMe  Data definition created |

2. Executive Summary

If you're seeking for a fresh and easy way to remain on top of your day-to-day obligations, go no further than our smartphone reminder app. The application's intuitive layout makes it simple to schedule reminders for future activities and appointments, ensuring that users never forget a single detail. Our mobile reminder software helps people live more relaxed lives by keeping track of their daily responsibilities, which is one of its main advantages. Users will never miss a meeting or a deadline because our app is designed to reliably notify them at the time, they need it. Additionally, the software can be customized to meet the needs of the user, allowing them to prioritize tasks and set reminders. Our smartphone reminder app is novel because of how easy it is to use. We designed our reminder software with the user in mind, making it easy to use even for people who don't know much about technology.

The user-friendliness of the software makes it accessible to anyone who wants to use it. Our mobile reminder app has value because it can streamline people's daily routines, allowing them to save time and work more efficiently. Our application is ideal for busy people who need to keep track of multiple tasks and stay organized. Individuals can avoid the stress and hassle of forgetting important events and appointments by using our application. People can manage their daily tasks and appointments with the help of the mobile reminder application, which is easy to use and convenient. Location- based reminders are also essential features of the application. This will allow users to set reminders based on their location, such as reminding them to buy groceries when they are near a supermarket. This feature requires the app to have access to the devices location based services.

The application's novelty lies in its simplicity and effectiveness, and its key advantages include customization, timely notifications, and user-friendliness. Individuals can simplify their lives, reduce stress, and increase productivity by utilizing the application. The application has value because it simplifies daily routines and helps people avoid forgetting important events and appointments. In general, the mobile reminder app is a great tool for anyone looking for a quick and easy way to keep track of their daily tasks and appointments.

In conclusion, our mobile reminder app is a useful tool that can make life easier, lessen stress, and make people more productive. Timely notifications, customization, and ease of use are among its primary benefits. It is novel because it is easy to use and works well, and its value comes from helping people save time and stay organized. If you're looking for a quick and easy way to keep track of your daily tasks and appointments, our mobile reminder app is a great choice.

3. Competitive Analysis

In today's fast-paced world, the demand for reminder apps has increased significantly. With so many apps available in the market, it becomes essential to analyze the competition and identify the key players to create a unique selling proposition for a new reminder app. A competitor for this remind app would be “Todoist”.

|  |  |  |
| --- | --- | --- |
| **Key Features** | **Todoist** | **RemindAll (Our Planned App)** |
| Customization | Flexible | More Flexible |
| Task Organization | Organization | Better Organization |
| Communication/Collaboration | Teamwork Functionality | More Advanced |
| Smarter Automation | Basic | Advanced |
| User-Interface | Friendly for User | More Friendly for User |

In comparison to Todoist, our planned product will have some additional advantages:

1. More customization options: Our product will offer greater flexibility for users to customize their task lists and workflows, allowing them to tailor the product to their individual needs and preferences.

2. Enhanced task organization: Our product will provide more advanced organizational features, such as task grouping and prioritization, making it easier for users to manage and prioritize their to-do lists.

3. Better communication and collaboration: Our product will facilitate seamless communication and collaboration among teams, with real-time updates and commenting functionality.

4. Smarter automation: Our product will include more intelligent automation features, such as smart suggestions for task completion and personalized task recommendations based on usage history.

5. Intuitive user interface: Our product will have a user-friendly interface with a modern and sleek design, making it easy for users to navigate and use the product.

Overall, our planned product aims to build upon the core features of Todoist while offering more advanced functionality and customization options, enabling users to work more efficiently and effectively.

4. Data definition

App ID: Every app deployed to the App Engine is identified by the Application ID and its version.

App update: App update happens when developer offers new features in an app or change the look or design of an app.

Binary search: to get stored and search of the different reminders.

Data form receiver: a form that will hold all the data that a user enters for a specific task.

Delay notification: if the user wants another reminder for them in a 5 min spam.

End User: The person for which the app was intended for.

Front-end frameworks: are JavaScript libraries with CSS and Bootstrap components that assist you in creating a website, web app, or mobile app.

Metadata: Data that defines and markets an app to consumers, like title, keywords, description, icon, screenshots, company name, and category.

Notification Message: message sent to the user when the reminder is activated.

Offline use: since it’s a reminder app it should be able to be used with no internet.

User: the person that decides to put a reminder.

System: in this case our software is one of the entities having a direct relationship with the end user

Type of Users: reminder app has a broad range of users, basically anyone that can use a mobile phone can be a target user, since all of need reminders of tasks.

5. Overview, scenarios and use cases.

The mobile reminder app is a productivity tool designed to help users remember important events, deadlines, and appointments. It will be available on iOS platforms. A mobile reminder app is a software application designed to help users remember important events, appointments, or tasks. The app provides users with the ability to set reminders for specific dates and times and can send notifications to their mobile devices to ensure they don't forget.

Busy Professional:

As a busy professional, John has a lot of meetings and appointments to attend each week. He needs to ensure that he doesn't forget any important events, so he uses the mobile reminder app to set reminders for each appointment. The app sends notifications to his phone prior to the event, ensuring that he never forgets an important meeting.

Forgetful Student:

As a forgetful student, Alice has trouble remembering her homework assignments and upcoming exams. She uses the mobile reminder app to set reminders for each assignment. The app sends notifications to her phone, ensuring that she stays on top of her academic responsibilities.

Fitness Enthusiast:

As a fitness enthusiast, Mike needs to stay on track with his workout schedule. He uses the mobile reminder app to set reminders for each workout session. The app sends notifications to his phone, ensuring that he never misses a workout.

Reminder for Medication:

The mobile reminder app can be used by individuals who need to take medication at specific times each day. The app can be used to set reminders for medication, ensuring that the user never forgets to take their medication.

Reminder for Appointments:

The mobile reminder app can be used to set reminders for important appointments, such as doctor's visits or meetings. The app sends notifications to the user's phone, ensuring that they don't forget about their appointments.

Reminder for Important Dates:

The mobile reminder app can be used to set reminders for important dates, such as birthdays, anniversaries, or other special occasions. The app sends notifications to the user's phone, ensuring that they don't forget about these important events.

Reminder for Tasks:

The mobile reminder app can be used to set reminders for tasks that need to be completed, such as paying bills or completing household chores. The app sends notifications to the user's phone, ensuring that they stay on top of their responsibilities.

Overall, the mobile reminder app is a useful tool for anyone who needs help staying organized and managing their time effectively. It is easy to use, customizable, and helps users remember important events and tasks.

6. high-level functional requirements

1. Allow users to create multiple reminders list for different purposes: This requirement allows users to create and organize their reminders into separate lists based on different categories such as work, personal, fitness, etc. This helps users keep their reminders organized and easy to manage.
2. ﻿﻿﻿Provide a range of customization options for each reminder: Users need the ability to customize their reminders as per their needs. This includes setting a specific date and time, location, repeating frequency (daily, weekly, monthly, yearly), and priority level. This ensures that users receive reminders at the right time and with the right frequency.
3. ﻿﻿﻿Allow users to categorize their reminders using tags or labels: This requirement allows users to add tags or labels to their reminders for easy sorting and filtering. For example, users can add the "work" tag to their work-related reminders, and the  
   "personal" tag to their personal reminders.
4. ﻿﻿﻿Provide various reminder types: Users may prefer different types of reminders, such as simple notifications, sound alarms, or vibration. Providing multiple reminder types helps users choose the most appropriate type for their specific needs.
5. ﻿﻿﻿Enable users to share their reminder lists with other users or export them to other apps or devices: This requirement allows users to share their reminder lists with others, such as family or colleagues or export them to other apps or devices. This ensures that users can access their reminders from any device or share them with others easily.
6. Provide a snooze function: This requirement allows users to delay a reminder for a specific period of time, such as 5 minutes or 10 minutes. This is helpful when users are busy or not able to attend to the reminder application immediately.
7. Offer a user friendly interface: The app will have a user friendly interface with easy navigation, intuitive controls and an appealing design, This will ensure that the users can use the app without any difficulty and enjoy a good user experience.
8. Allows users to set reminders based on their location: This requirement allows users to set reminders based on their location, such as reminding them to buy grocery if they are near a supermarket.

7. List of non-functional requirements

1. Performance: The mobile app must be fast and responsive, with minimal loading times and low memory consumption. The app must respond to user input within 1 second.

2. Usability: The mobile app must be intuitive and user-friendly, with clear and concise instructions for all features. The design must be optimized for mobile devices and should support gestures like swipe and pinch.

3. Accessibility: The mobile app must meet the standards of WCAG 2.1, with support for screen readers, large text sizes, and high contrast modes. The app must be compatible with both iOS and Android accessibility features.

4. Security: The mobile app must ensure the privacy and security of user data. All user data must be encrypted and stored securely. The app must be able to detect and prevent unauthorized access, tampering, or data loss.

5. Availability: The mobile app must have a minimum uptime of 99%, with downtime not exceeding 2 hours per month for maintenance. The app must be available for download from both Apple App Store and Google Play Store.

6. Scalability: The application must be able to handle increased user load without performance degradation, and scale horizontally as needed.

7. Storage: The application must be able to store large amounts of data, with a minimum of 1GB per user.

8. Fault Tolerance: The application must be resilient to hardware and software failures and should have a disaster recovery plan in place to ensure data availability in the event of a catastrophic failure.

8.High-level system architecture

Proposal:

The mobile reminder app we propose to build will be developed for iOS using native iOS frameworks and programming languages such as Swift and Objective-C. The app will allow users to set reminders for various tasks and events, including appointments, deadlines, and recurring events. The app will also have a feature for location-based reminders that trigger when the user enters or exits a specified location.

A UML class diagram is a suitable visual representation of the code architecture. The class diagram will have classes like Reminder, User, Category and Notification. A reminder’s title, description, date, time and location will all be stored in the Reminder class. A user’s name, email address and password are stored in the user class. The Notification class will house information about alerts that were delivered to the user, while the Class class will house categories for reminders.

The relationships between the classes will be visualized using UML associations. To demonstrate how each reminder can be connected to a category, the Category class will be linked to the Reminder class, for example. The User class will be connected to the Reminder class in a similar way, indicating that one user may have numerous reminders.

DB Organization:

Four tables will make up the core database structure for the mobile reminder app: Users, Reminders, Notifications, and Categories. The Users table will keep track of user login information including username and passwords as well as any user profile data like email or name. The Name, Description, Due Date, and Category of each individual reminder will be stored in the Reminders table. Each reminder will have a special ID that will allow the database to recognize it. Other fields in the table, like repetition frequency or notes, may also be present. Information about each notification that must be sent for a reminder, such as the notification time and the device to send it to, will be stored in the Notifications table. Each notification's ID will be used to link it to a specific reminder.

The many categories of reminders that the user can select from will be stored in the Categories table. Each category will have its own ID, name, and sometimes a color code. Overall, the database structure is built to effectively store and retrieve reminder-related data, and links between the various tables are constructed to guarantee data consistency and integrity.

To inform the user of any changes to their reminders, the app can also use the Observer pattern in addition to the MVC design. A push notification system will be used to implement the Observer pattern; when a reminder is approaching, alerts will be sent to the user's smartphone.

With an emphasis on scalability, maintainability, and usability, the high-level architecture of the code will generally adhere to accepted principles of software architecture and design patterns.

Media Storage:

It is typically advised to store large media items like pictures, films, and music in a file system rather than a database. Hence, it is advised to just keep references to photos and videos in the database for the reminder mobile app and to store them in a file system.

Instead of keeping the accompanying image or video file itself in the database, the Reminders table, for instance, could have a field for storing the file path or URL. This strategy is more effective and makes managing and backing up media files simpler.

While storing GPS data in the database, the app can use standardized GPS data formats like latitude and longitude. The software can also use common audio file formats for audio data, including MP3 or WAV, and keep the file path or URL in the database. The secret is to adopt uniform formats that work with the app's functionality and are simple to read.

Search/Filter Architecture and Implementation:

The reminder mobile app's search/filter algorithm will be determined by the search criteria and

the sorts of data being searched.

For instance, the app might offer search parameters for reminders that are overdue, today's deadline, or related with a specific tag. Reminder titles, due dates, tags, and other reminder related metadata will all be looked for in the database.These terms will be kept in the database's Reminders table as fields.

High-level system architecture:

The app will consist of a client-side iOS application and a server-side API for syncing and backup purposes. The app will be developed using Xcode, Apple's integrated development environment (IDE), which includes a suite of tools for designing, building, and testing iOS applications.

The client-side application will be built using Swift and Objective-C and will utilize Apple's Core Data framework for storing and managing data. The app will also use Apple's Location Services framework for location-based reminders.

The server-side API will be built using Node.js and will utilize a RESTful API architecture for communication with the client-side application. The API will be responsible for syncing reminders across multiple devices and providing backup and restores functionality. The API will also utilize a database system for storing user data, which will be encrypted and secured using industry-standard security protocols.

Our mobile reminder app uses Node.js and Express, two very popular tools in the JavaScript ecosystem.

Node.js provides an event driven model which makes it very light weight and efficient.

Express on the other hand, is a web application framework that simplifies the process of building APIs and web applications.

The integration of both allows our application to handle requests and responses efficiently, which in turn makes it fast and responsive.

Additionally, the usage of JavaScript for both the client and server side reduces development time and simplifies code maintenance.

Programming languages:

1.Javascript

2.CSS

3.HTML

Tools:

1.Git

2.Github

3.Visual Studio Code

Supported Browsers:

1.Google chrome

2.Safari

3.Microsoft edge

Core APIs:

1.Google Maps API

2.Stripe API

3.OpenWeatherMap API

List of core APIs available at this point:

Apple's Core Data and Location Services frameworks RESTful API for syncing and backup purposes Supported browsers:

Non-Trivial Algorithm:

Reminders will be ranked in the app according to their urgency and importance using an algorithm. Users can customize each reminder's urgency and importance levels in the app, and the app will prioritize reminders depending on these settings. Additionally, the app will automatically reschedule reminders, if necessary, based on machine learning algorithms that anticipate the likelihood that a user will finish a reminder.

Automatic ranking of reminders based on their urgency or importance is one non-trivial technique that might be employed in this reminder software.

This might be accomplished by giving each reminder a priority score depending on the reminders due date, time-sensitivity level, and user-defined tags or labels.

The reminders might then be sorted by priority, with the most urgent or significant ones at the

top of the list.

A weighted scoring system, in which various elements are assigned various weights based on their relative importance, could be used by the algorithm to determine the priority score.

For instance, the due date can be given a higher priority than the tag or label.

The algorithm might also dynamically change the weightings based on user feedback and behaviour, making the prioritising more individualised and suited to the user's requirements over time.

9

10. Identify actual key risks for your project at this time

1. Skills risks: There may be a risk that the team members don't have the required skills to complete the project successfully. This risk can be mitigated by identifying the skills needed for the project and ensuring that the team has access to training or hiring new members with the required skills.

2. Schedule risks: There may be a risk that the project schedule is too ambitious or that unexpected delays occur, causing the project to be delayed. This risk can be mitigated by setting realistic timelines, identifying potential roadblocks, and having contingency plans in place.

3. Technical risks: There may be a risk that the technology used for the project is not suitable or that technical problems arise during the development process. This risk can be mitigated by thoroughly testing the technology before using it, having backup plans in place, and involving technical experts in the development process.

4. Teamwork risks: There may be a risk that communication or collaboration problems arise among team members, causing delays or misunderstandings. This risk can be mitigated by establishing clear communication channels, having regular meetings to discuss progress and issues, and promoting a positive team culture.

5. Legal/content risks: There may be a risk that the content or software used for the project is not legally obtained or licensed. This risk can be mitigated by ensuring that all content and software used for the project is legally obtained and properly licensed.