Software Design Document

for

Mindful Life Companion

Version 1.0 approved

Prepared by: Andy Liang, Tommy Works, Anthony Chieng, Bryam Ochoa, Luis Sanchez, Emily Perez, Giovanni Chumpitazi, Jay Chow

CS 3337-03

5/9/2024

Table Of Contents

Revision History

Name	Date	Reason For Changes	Version
Andy	5/2/24	Removed a component on app, therefore no longer needs a database	1.0.1

<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>

(Jay Chow)

1. Introduction

1.1 Purpose

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.

The purpose of this document is to specify the software requirements for the Mindful Life Companion, a mental health app. This document represents the first release of the SRS (Software Requirements Specification) for the app.

The scope of this SRS covers the entire Mindful Life Companion app. It describes the required functionality, features, and behavior of the app as a whole, rather than focusing on a specific subsystem or only a part of the system. The SRS aims to provide a comprehensive understanding of the software requirements for the initial release of the Mindful Life Companion app.

1.2 Document Conventions

Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.

Requirement Prioritization: Each requirement statement in this document has its own priority assigned to it. The priority levels indicate the relative importance or urgency of fulfilling the specific requirement. The priority levels can help guide the development process and decision-making regarding resource allocation and implementation order.

Fonts and Formatting: Standard fonts and formatting conventions are used throughout the document. The main content is presented in a clear and legible font, typically in a regular style. Headings and subheadings may be formatted in a bold or larger font size to provide visual hierarchy and improve readability.

Highlighting or Emphasis: Specific terms, concepts, or important information may be highlighted or emphasized using techniques such as bold text, italics, or underlining. This is done to draw attention to critical aspects of the requirements or to convey essential points.

Acronyms and Abbreviations: Acronyms and abbreviations are defined upon their first use in the document. This ensures that readers can easily understand the meaning of these terms as they encounter them.

1.3 Intended Audience and Reading Suggestions

Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Developers: Developers will refer to this document to understand the functional and non-functional requirements of the app, as well as the system architecture and technical specifications.

Project Managers: Project managers will utilize this document to gain a comprehensive understanding of the app's requirements, scope, and constraints. It will assist them in planning and coordinating the development process, resource allocation, and project scheduling.

Users: Users of the Mindful Life Companion app may refer to this document to gain insights into the app's functionality, features, and overall user experience. It can serve as a reference to understand what the app offers and how it can benefit them.

Testers: Testers will utilize this document to develop test cases and scenarios based on the specified requirements. It provides a clear understanding of the expected behavior and functionality, enabling comprehensive testing.

Documentation Writers: Documentation writers will refer to this document to create user manuals, help guides, and other instructional materials. It provides essential information about the app's features and usage, assisting in creating accurate and informative documentation.

Readers should read this document in manner the sections are organized (see table of contents).

1.4 System Overview

Provide a general description of the software system including its functionality and matters related to the overall system and its design (perhaps including a discussion of the basic design approach or organization

The mental health app being developed by our group aims to provide users with a comprehensive set of features to support their well-being and promote mental health. The app incorporates functionalities such as account authentication, mood tracking, journaling, medication reminders, and a quote generator. These functionalities will be explored in other sections.

Giovanni C.

2. Design Considerations

2.1 Assumptions and Dependencies

Assumptions

- The app will require users to sign up and verify their email address to provide security.
- The app will use Firebase as its database for journaling purposes.
- The app will integrate with an external API to generate random quotes.

Dependencies

- The app will depend on Firebase for storage and data management.
- The app will rely on Firebase for user authentication and cloud-based functionalities.

2.2 General Constraints

Hardware or Software Environment

- The app is designed specifically for Android operating systems.
- Firebase will handle online data synchronization.

Standards Compliance

- The app must comply with Google Play Store guidelines for app submission.
- Firebase implementations will follow Android development best practices.

Performance Requirements

- The app should maintain responsiveness and performance under varying network conditions.
- Efficient data caching and synchronization strategies will be implemented for optimal performance.

2.3 Goals and Guidelines

Emphasis on User Experience

- The design prioritizes a seamless and intuitive user experience.
- Journaling features should be user-friendly and accessible.

Security Requirements

- User data must be securely stored and transmitted.
- Authentication mechanisms will be implemented for data security.

Mandatory Delivery Date

- End of CS 3337-03 Spring 2024 Semester

2.4 Development Methods

The development approach for this software design follows an Agile methodology. Iterative development cycles allow for flexibility and rapid iteration based on user feedback. Sprint planning, and retrospectives are key components of the Agile process. Continuous integration and deployment pipelines ensure efficient code integration and testing.

Emily

3. Architectural Strategies

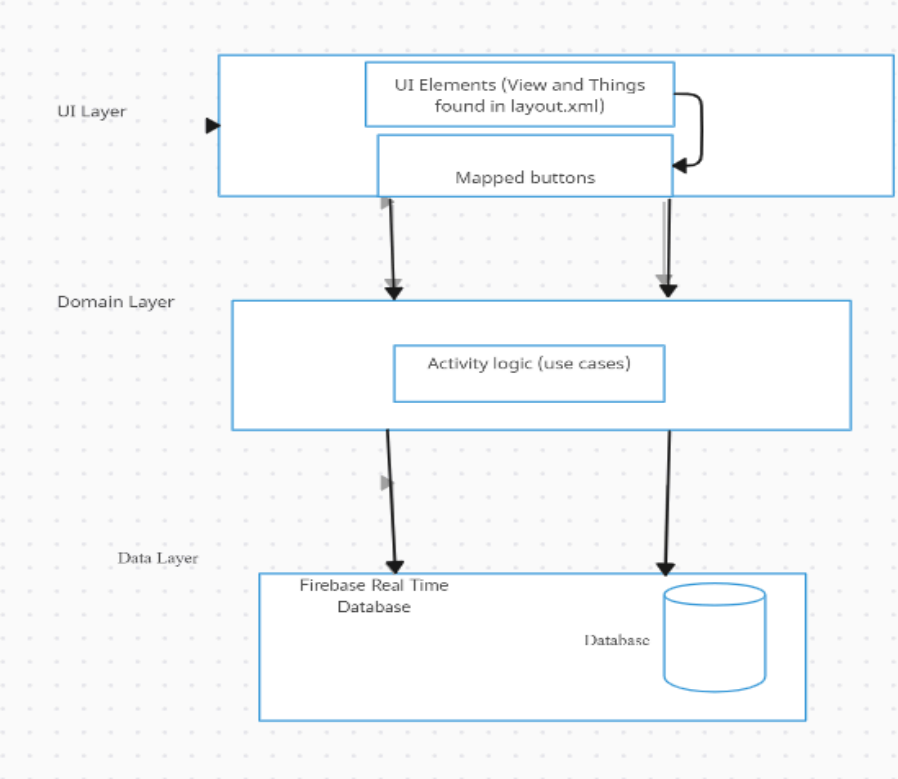
Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

- Architectural Pattern
- Use of a particular type of product (programming language, database, library, etc. ...)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery
- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms
- Management of other resources

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

- Architectural Pattern:

The chosen architectural pattern for Mindful Life Companion developed in Android Studio Iguana using the Kotlin programming language is a layered architecture.



Reasoning:

A layered architecture provides a clear separation of concerns and promotes maintainability and testability. It involves organizing the app into layers, such as the presentation layer, business logic layer, and data layer. Each layer is responsible for a specific functionality, ensuring that changes in one layer do not impact the others. This approach allows for better code organization and makes it easier to maintain and test different parts of the app independently.

- Database:

Firebase is chosen as the database solution for our app, providing a scalable and cloud-based data storage and synchronization platform.

Reasoning:

Firebase offers real-time data synchronization, offline support, and easy integration with Android apps. It provides a NoSQL database, Firestore, which is well-suited for mobile applications and eliminates the need for complex server-side setups. Firebase also offers user authentication and other useful features like cloud messaging and remote configuration.

- Reuse of Existing Software Components (API for Quote Generation):

The app leverages an external API to retrieve and display quotes within the app.

Reasoning:

By utilizing an API for quote generation, the app can access a vast collection of quotes without the need to manually manage and update them within the app's codebase.

- Future plans for enhancing software

Future iterations of the Mindful Life Companion app will primarily focus on enhancing existing features while ensuring their stability/usability.

Reasoning: Our priority is to refine the current core features of the app, ensuring a seamless user experience and addressing any usability issues. This entails optimizing existing functionalities such as mindfulness exercises, mood tracking, and goal setting to better meet the needs of our users.

- User Interface Paradigms:

The app's user interface follows Android's Material Design guidelines, incorporating modern UI paradigms and responsive design principles.

Reasoning:

Material Design provides a visually appealing and intuitive user experience, with consistent design patterns across different Android devices. By adhering to Material Design guidelines, the app can offer familiarity and usability to users. The UI design should align with the app's goal of promoting a mindful and enjoyable user experience.

- Error Detection and Recovery:

The app implements comprehensive error handling mechanisms, including structured exception handling, logging, and user-friendly error messages. As well as implementing retry mechanisms for network requests.

Reasoning:

Effective error detection and recovery ensure a stable and reliable app experience for users. By implementing structured exception handling and logging, developers can identify and troubleshoot issues more efficiently. User-friendly error messages help users understand and resolve encountered errors, enhancing the overall user experience. This approach will prevent the app from crashing due to errors.

- External databases and/or data storage management and persistence

Our external data storage options include cloud-based databases (Firebase), local storage (SQLite databases), or file systems.

- Distributed data or control over a network

Syncing user data across multiple devices or accessing data from remote servers for features like the motivational quote generator or medication reminders.

- Communication mechanisms

Communication mechanisms are essential for various functionalities such as retrieving motivational quotes from an external API, syncing user data with a cloud-based server, and sending medication reminders to users' devices.

Bryam Ochoa

4. System Architecture

4.1 Logical View

4.2 Development View

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

Authentication Subsystem: Responsible for user authentication and account management.

Journaling Subsystem: Manages the journaling feature of the app.

Mood Tracker Subsystem: Facilitates the mood tracking feature of the app.

Medication Reminder Subsystem: Helps users with medication management.

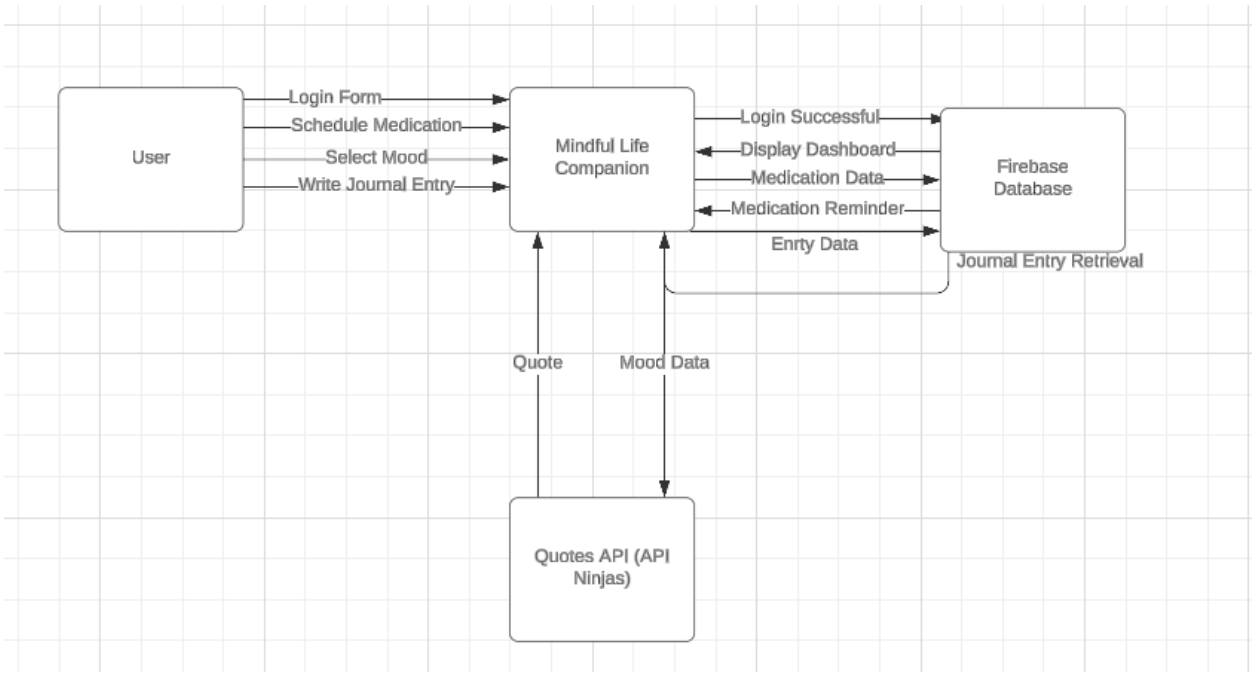
Quote Generation Subsystem: Integrates with an external API to generate quotes.

Firebase Database: Serves as the backend database for storing user data.

Integration and Communication: Ensures interaction and data sharing between subsystems/components.

User Interface: Responsible for rendering features and enabling user interactions.

This is where the level 0 DFD will probably work best.



At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.

At the top-most level, the Mindful Life Companion app can be broken down into the following major responsibilities and roles:

User Authentication and Account Management:

- Responsible for managing user authentication and account-related operations.
- Handles user registration, login, logout, and password management.
- Ensures secure access to the app's features and data.

Journaling and Calendar View:

- Responsible for managing the journaling feature of the app.
- Enables users to create, view, edit, and delete journal entries.

Mood Tracking and Calendar View:

- Facilitates the mood tracking feature of the app.
- Allows users to record and track their daily moods.
- Presents mood data on a calendar view for easy visualization and analysis.
- Collaborates with the Calendar View component to display mood records in a calendar format.

Medication Reminder:

- Manages the medication reminder feature of the app.
- Sends reminders and notifications for scheduled medications.
- Allows users to set medication schedules and track adherence.

Quote Generation and Integration with API:

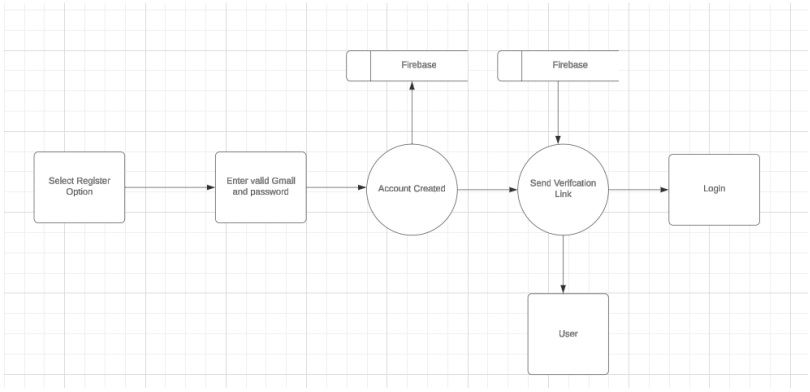
- Integrates with an external API to generate quotes.
- Retrieves quotes based on the user's selected mood.
- Presents the quotes to the user for inspiration and motivation.

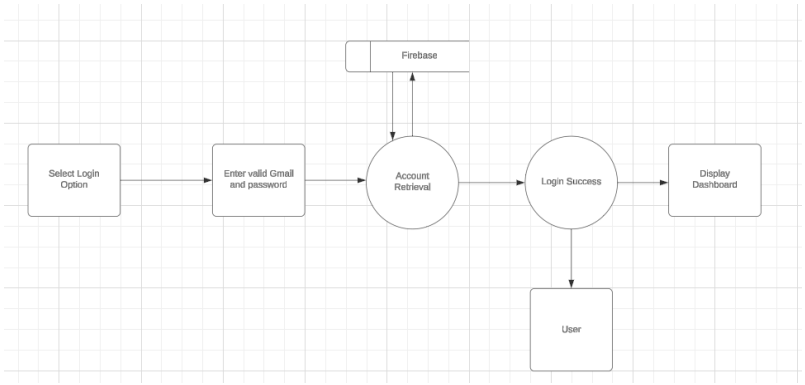
Firebase Database Integration:

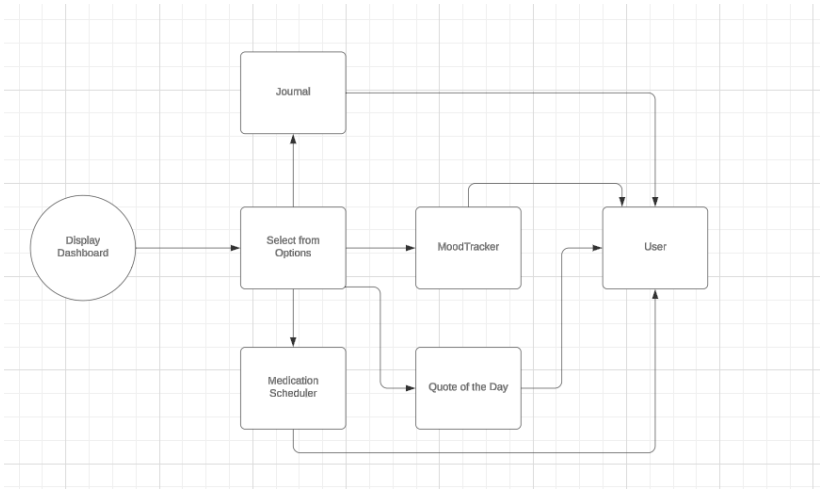
- Integrates with Firebase database for data storage and retrieval.
- Stores user authentication information, journal entries, mood records, and other relevant data.

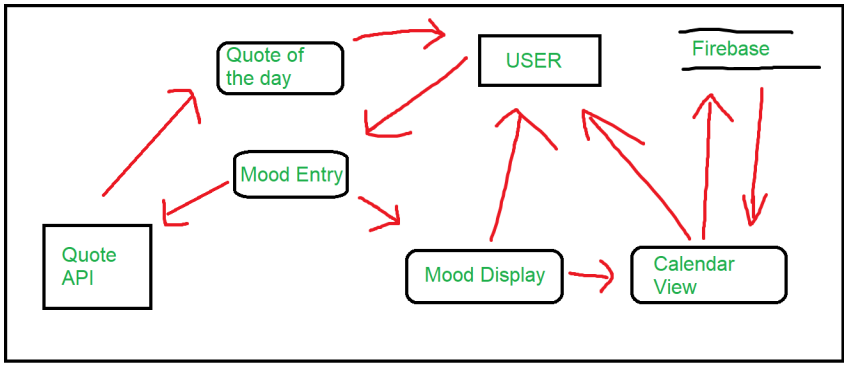
- Ensures secure and scalable data management.

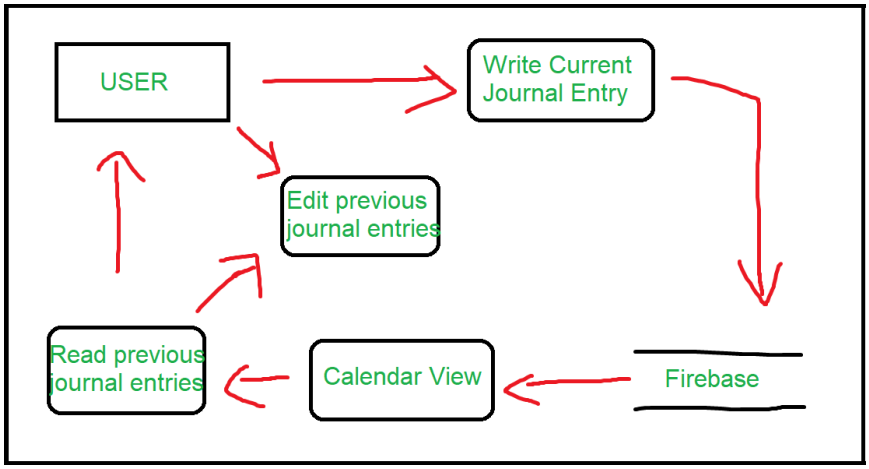
Level 1 Data Flow Diagrams (DFD)

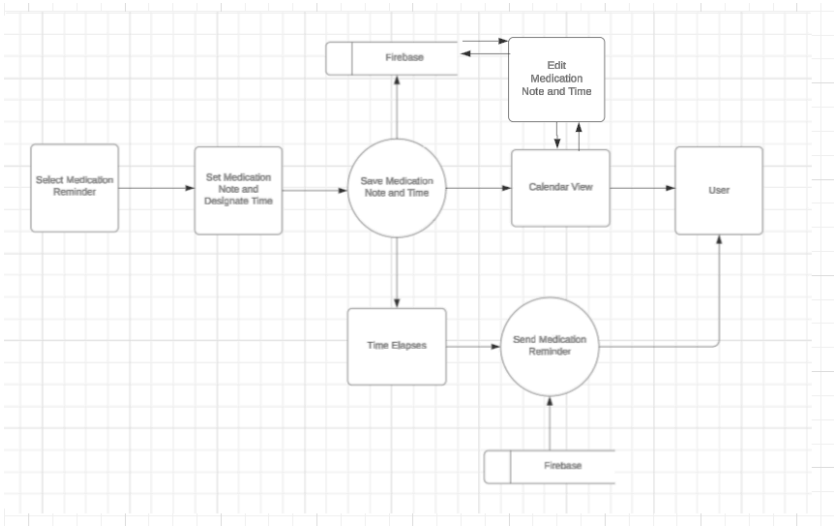


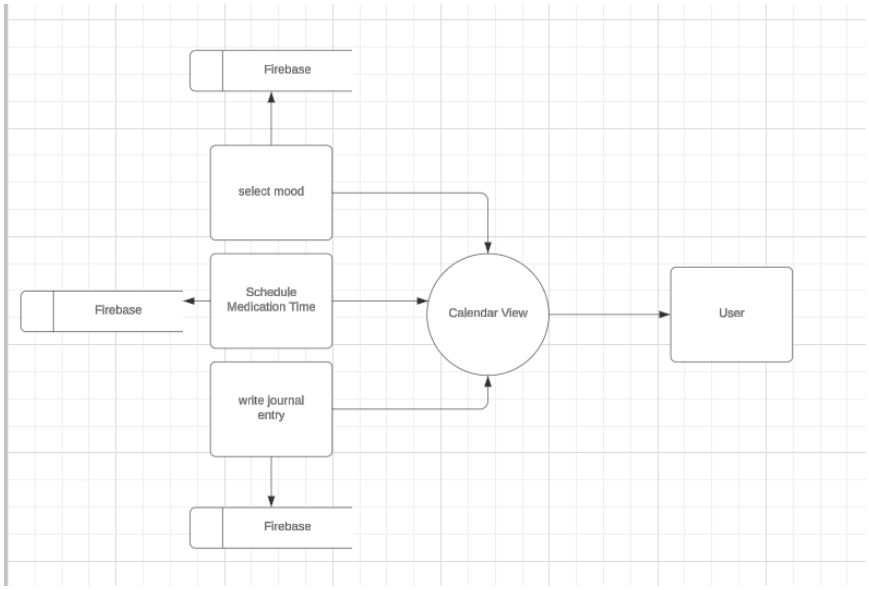


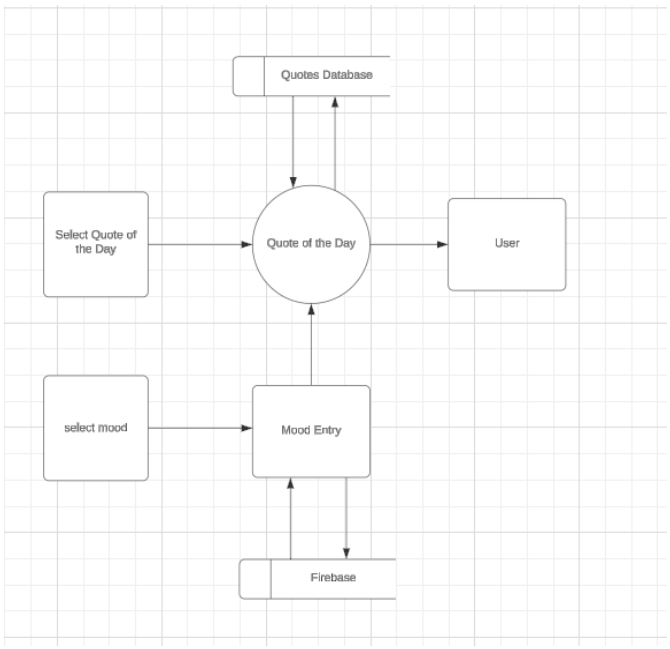



Quote of the day
USER
Firebase
Mood Entry
Quote API
Mood Display
Calendar View


USER
Write Current Journal Entry
Edit previous journal entries
Read previous journal entries
Calendar View
Firebase

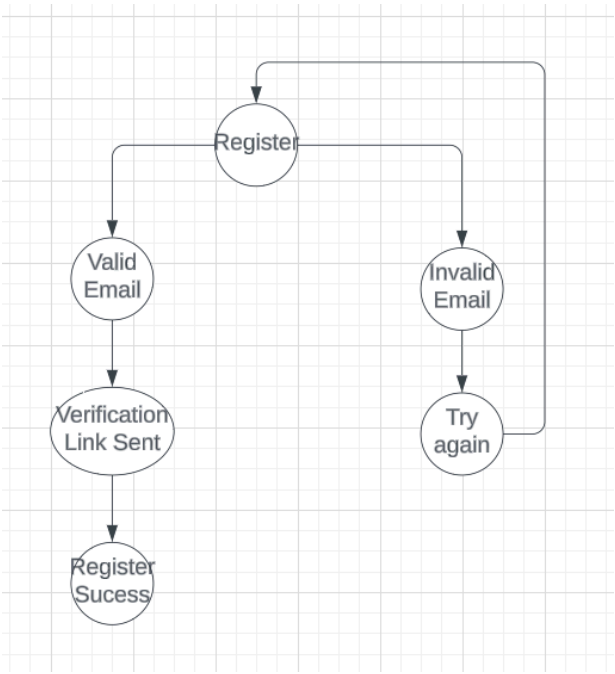

Firebase
Edit Medication Note and Time
Select Medication Reminder
Set Medication Note and Designate Time
Save Medication Note and Time
Calendar View
User
Time Elapses
Send Medication Reminder
Firebase

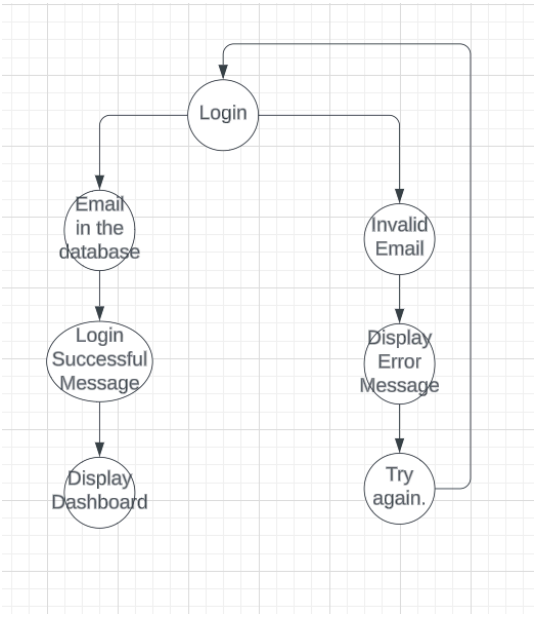


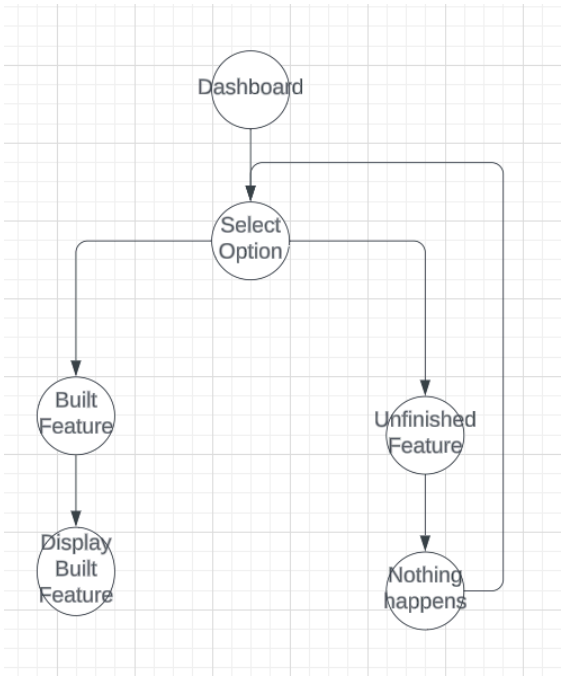


4.3 Process View

Control Flow Diagrams (CFD) go here.


Register
Valid Email
Invalid Email
Verification Link Sent
Try again
Register Sucess


Login
Email in the database
Invalid Email
Login Successful Message
Display Error Message
Display Dashboard
Try again.



Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

Collaboration between Components:

- The Authentication component collaborates with all other components to ensure secure access to the app's features and data. It provides user authentication and account management functionalities, allowing users to register, log in, and manage their account credentials.
- The Journaling and Calendar View components work together to enable users to create, view, edit, and delete journal entries. The Calendar View component displays journal entries in a calendar format, providing an intuitive way for users to navigate and interact with their journaling history.
- The Mood Tracking and Calendar View components collaborate similarly, allowing users to track their moods and view them on the calendar. Users can record daily moods, and the Calendar View component visualizes the mood records for easy analysis and reflection.

- The Medication Reminder component operates independently but may interact with other components to retrieve user information or send reminders. It helps users manage their medication schedules by sending notifications for scheduled medications.
- The Quote Generation component interacts with the Mood Tracking component to generate quotes based on the user's selected mood. It integrates with an external API to retrieve relevant quotes, providing users with motivational and inspirational content.

Rationale for System Decomposition:

- The system is decomposed to achieve modularity, separation of concerns, and maintainability. Each component focuses on specific functionalities, allowing for easier development, testing, and maintenance.
- The decomposition enables parallel development and promotes code reuse. Each component can be developed independently, enabling developers to work concurrently on different parts of the system.
- Alternatives such as merging the Journaling and Mood Tracking functionalities into a single component were rejected to keep the responsibilities separated and ensure clear separation of concerns.

4.4 Physical view

5. Policies and Tactics (Anthony Chieng)

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

5.1 Choice of which specific products used

(IDE, compiler, interpreter, database, library, etc. ...)

- Android Studio Iguana

- Firebase
- Quotes API from API Ninjas

5.2 Plans for ensuring requirements traceability

- We will be using Git and GitHub to track changes and maintain version control to keep track of requirements through the codebase.
- Additionally, we will be using GitHub projects to create roadmaps and backlogs to keep track of features and user stories that contain those requirements.

5.3 Plans for testing the software

- Unit testing will be implemented to ensure functions and methods in the Mindful Life Companion app are working correctly and do not fail.
- Compatibility testing to ensure multiple devices are able to run the app with multiple different screen sizes and versions by using both physical phones and virtual phones to test them.

5.4 Engineering trade-offs

- An online-based app that requires the internet means that while users are offline, its features will not work. However, this is with the benefit of multiple devices being able to share the same data for an account.

5.5 Coding guidelines and conventions

- The coding guidelines and conventions of the Mindful Life Companion app will mostly be using Kotlin's coding guidelines and conventions.
- The names of both classes and objects will start with an uppercase letter and use camel case.
- Functions, properties, and local variables will start with a lowercase letter and use camel case.
- Constants will be uppercase and separated with underscores.

5.6 The protocol of one or more subsystems, modules, or subroutines

- User authentication through registration with username, password, and an email or a Google/Facebook sign in.
- Mood tracking through a button that tells the system to update and display the calendar view after storing the mood onto Firebase.
- Medication reminders customized and scheduled by the user which gives a phone notification and a ring notification to the user's android device.
- Motivational quote generator that generates quotes depending on category selected for display on the dashboard from an api which can be bookmarked.
- Journaling that allows users to submit journal entries to Firebase and a journaling list that allows users to find and edit previous entries.

5.7 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

- We would be using object-oriented programming to solve problems. For example, a user could be represented as a class that contains properties of classes representing different settings, moods, reminders, and journaling.

5.8 Plans for maintaining the software

- Regularly testing the app and adding updates to features that may be causing bugs in the app.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

(SKIP)

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD's will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.x Name of Component (Module)

6.x.1 Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.x.2 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.x.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.x.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

6.x.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.x.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

(SKIP)

7. Detailed Lower level Component Design

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.x Name of Class or File

7.x.1 Classification

The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)

A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy

Class inheritance: parent or child classes.

7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation

8. Database Design (Andy)

8.1 Data Model

The Mindful Life Companion app will utilize Backend-as-a-Service that offers a NoSQL database. This section outlines the proposed database schema, including tables, columns, and their relationships.

8.2 Tables

8.2.1 Authentication

- **uid (String):** A unique identifier assigned to each user upon creation. This is a critical value used to link a user to their data in other parts of your application
- **identifier:** A string that provides the method for Authentication
- **email:** The user's email address provided during registration. Email address can be null if the user signs in with a provider such as Google or Facebook
- **displayName (String):** The user's display name, typically retrieved during registration or login from the chosen authentication method. It can be null if unavailable.

8.2.2 User Data

- **user_id (String):** Id that references the user whom the data belongs to
- **MoodEntries (Collection):** Collection of all mood entry documents
- **JournalEntries (Collection):** Collection of all journal entry documents
- **SavedQuotes (Collection):** Collection of all quote generator documents

8.2.3 Mood Entry Data

- **entry_id (Date):** Unique date used as identifier for entry
- **date (Date)**: Date as mm-dd-yyyy of the mood entry.
- **mood (String):** User's selected mood for the entry (e.g., Happy, Sad, Anxious).

8.2.4 Journal Entry Data

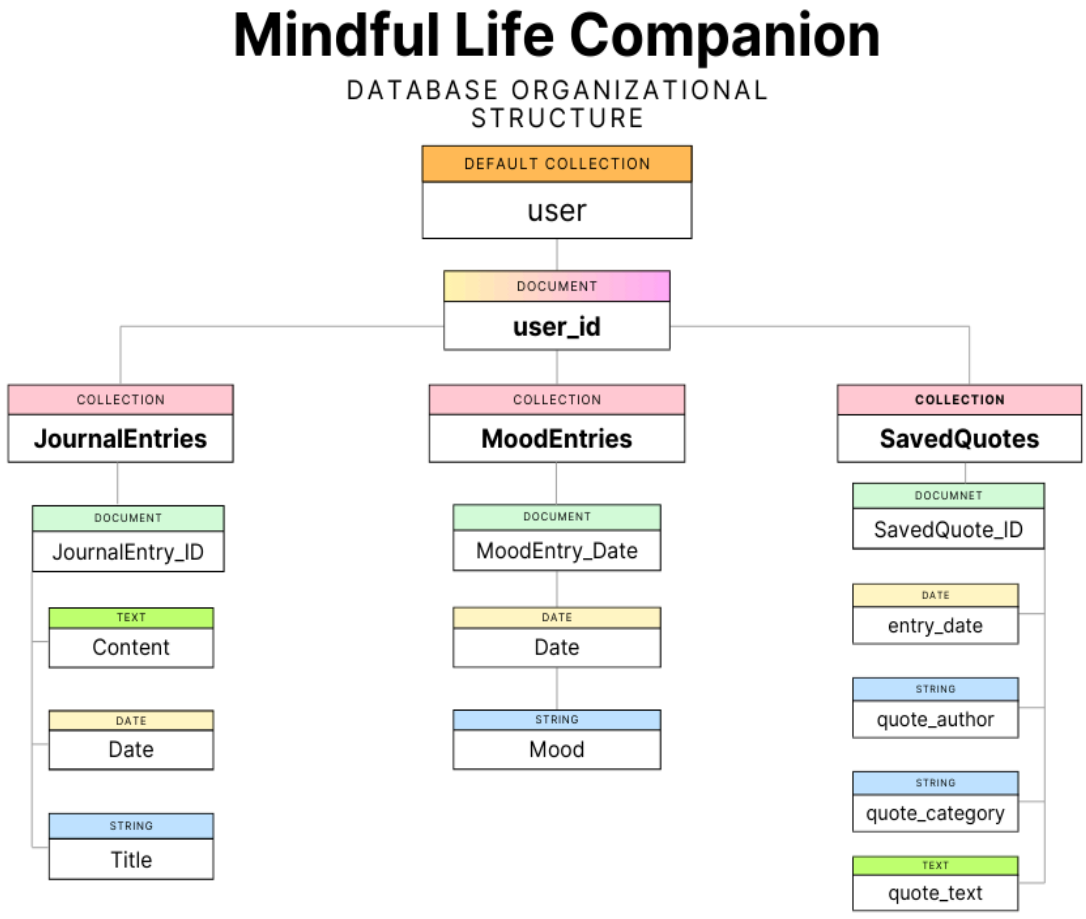
- **entry_id (String):** Unique identifier for the journal entry
- **Content(text):** The content of the body
- **Date (date):** The date of the journal entry as a full timestamp

- **Title (string):** Title of the entry

8.2.5 Saved Quotes Data

- **saved_quote_id (String):** Unique identifier for a saved quote
- **quote_category (String):** The default category of quotes which the API will use to pull quotes
- **entry_date (timestamp):** The date the quote was bookmarked
- **quote_author (string):** Author of the quote
- **quote_text (text):** The contents of the quote body

8.3 Relationships



9. User Interface (Luis Sanchez)

9.1 Overview of User Interface

The user interface of our app is designed to be intuitive and user-friendly, providing easy access to essential features aimed at improving mental health and well-being. Here's an overview of the functionality and user flow:

Welcome and Authentication: Upon downloading and opening the app, users are greeted with a welcoming login page. They will need to authenticate their email to access the app fully. This ensures the security and personalization of the experience.

Dashboard: After logging in successfully, users are directed to the dashboard, which serves as the central hub for accessing different app features. The dashboard prominently displays four main options:

- **Mood Tracker:** Clicking on this option redirects the user to a page where they can input how they are feeling. Users can select from a range of emotions or write a custom description of their mood. This information is then stored to track their emotional well-being over time.
- **Journal:** This option takes users to a journal-taking page where they can record their thoughts, experiences, or activities of the day. Writing in the journal can serve as a form of self-reflection and emotional processing.
- **Quote of the Day:** Users can generate daily quotes tailored to their current mood. Depending on their feelings, the app provides uplifting, humorous, inspirational, or reflective quotes. Users have the option to select the type of quote they prefer based on their mood.
- **Medication Reminder:** This feature helps users stay on track with their medication schedules. Users can set up reminders for taking medications at specific times throughout the day. This helps prevent accidental omissions and ensures the accuracy of prescribed treatment plans.

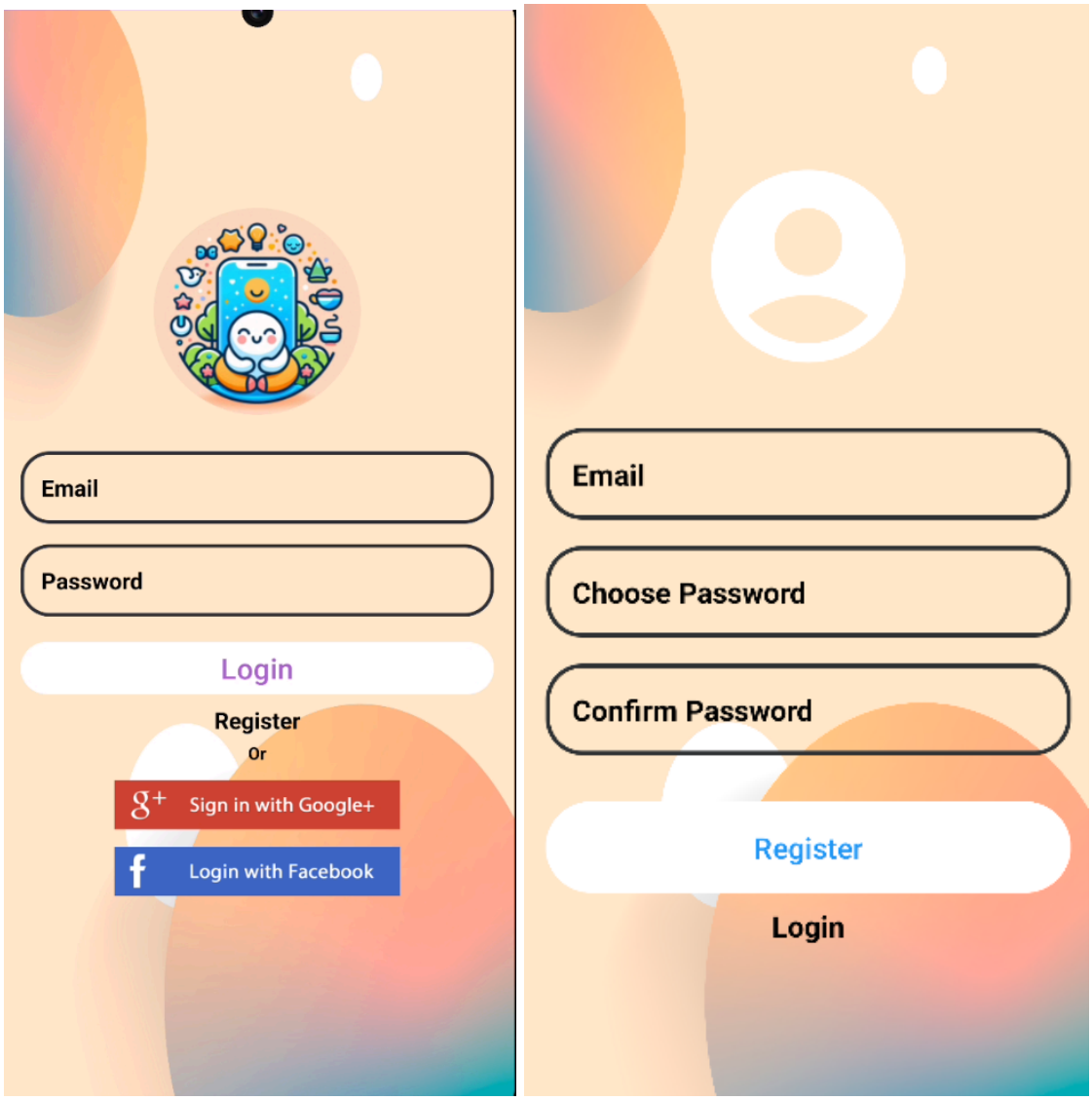
Feedback Information: Throughout the user journey, the app provides feedback to enhance the user experience. This includes confirmation messages upon successful actions (e.g., logging in, saving journal entries), and alerts for upcoming medication reminders.

Overall, the goal for the UI is designed to be visually appealing, easy to navigate, and supportive of the user's mental well-being journey. Combining mood tracking, journaling, personalized

quotes, and medication reminders offers a comprehensive approach to promoting self-awareness, reflection, and self-care.

9.2 UX Standards

9.2 Screen Frameworks or Images



Mindful Life Companion

Happiness

I thank my God for graciously granting me the opportunity of learning that death is the key which unlocks the door to our true happiness.

- Wolfgang Amadeus Mozart

Feeling the Week

Mon	Tues	Wed	Thurs	Fri	Sat	Sun
						

Menu


Mood


Journal


Quote


Reminder

BACK TO DASHBOARD

What is your mood today?

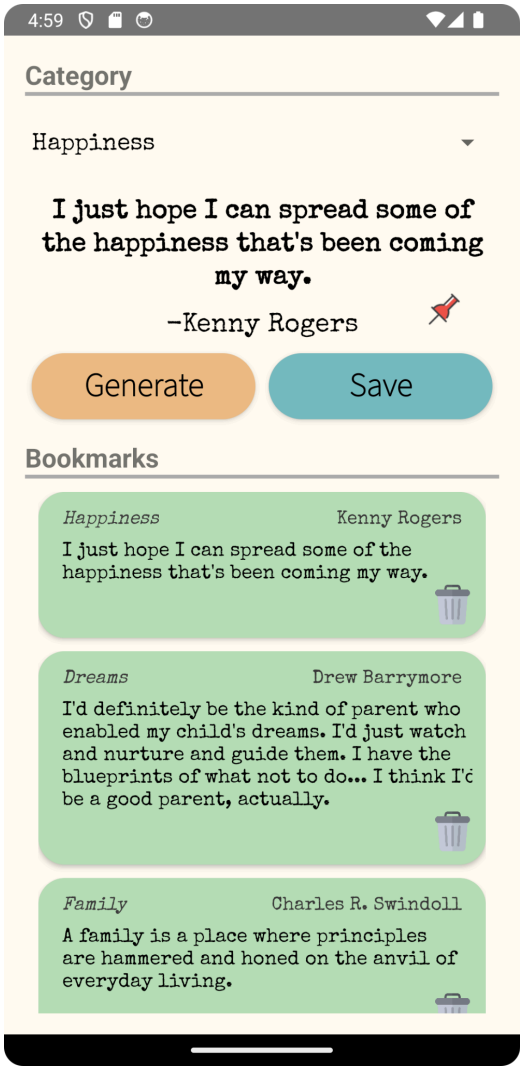
HAPPY 

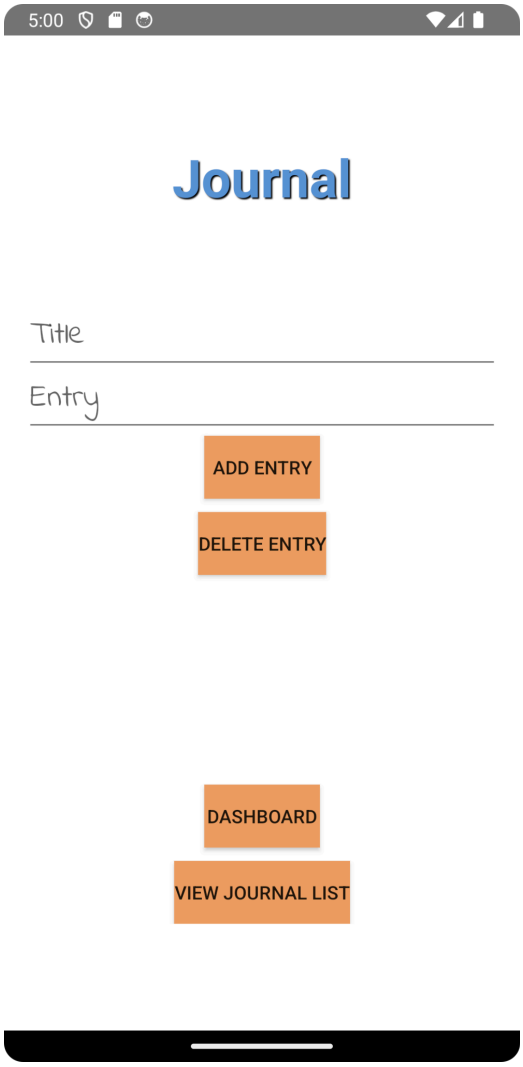
PRODUCTIVE 

SAD 

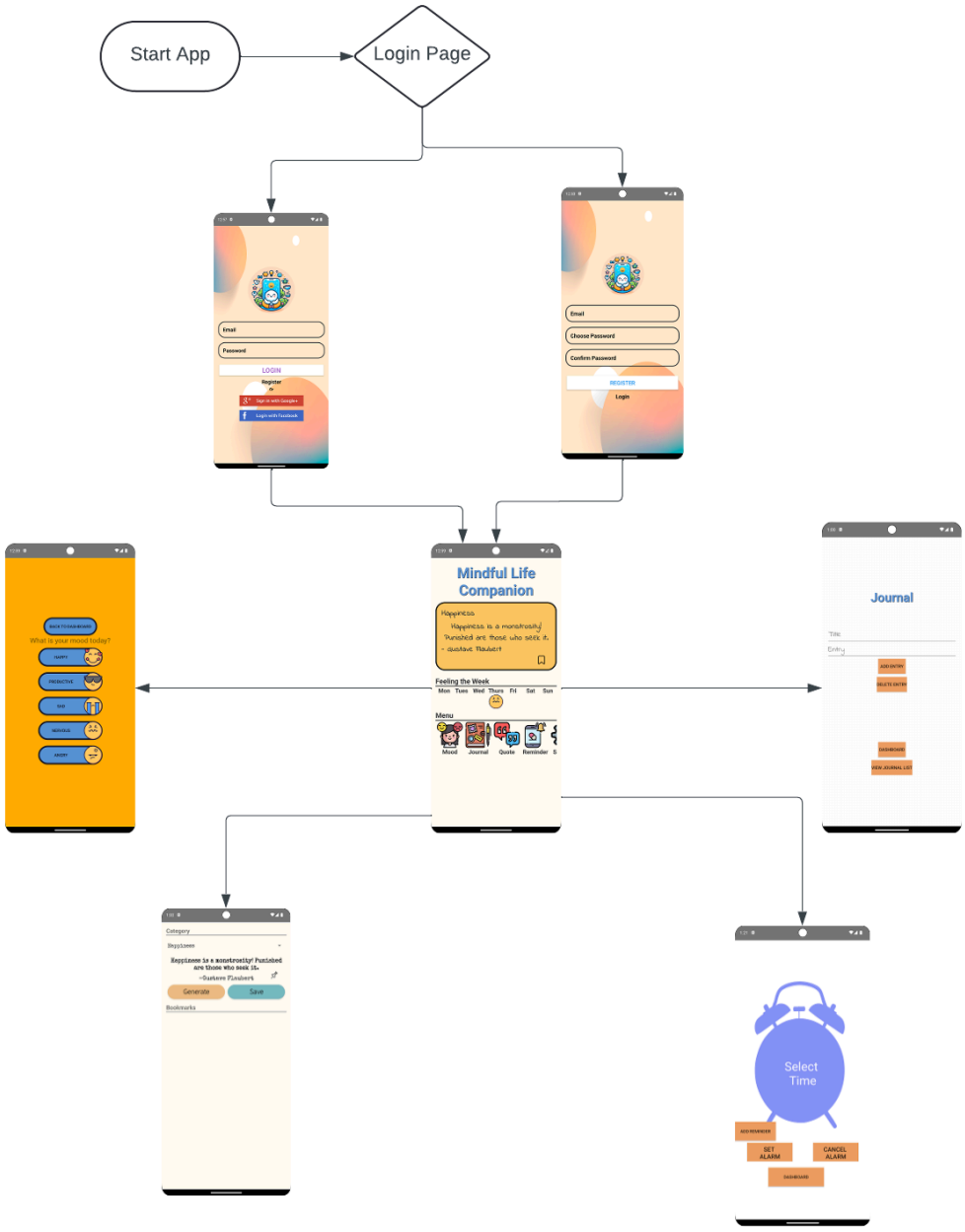
NERVOUS 

ANGRY 


4:59
Category
Happiness
I just hope I can spread some of the happiness that's been coming my way.
-Kenny Rogers
Generate
Save
Bookmarks
Happiness
Kenny Rogers
I just hope I can spread some of the happiness that's been coming my way.
Dreams
Drew Barrymore
I'd definitely be the kind of parent who enabled my child's dreams. I'd just watch and nurture and guide them. I have the blueprints of what not to do... I think I'd be a good parent, actually.
Family
Charles R. Swindoll
A family is a place where principles are hammered and honed on the anvil of everyday living.


5:00
Journal
Title
Entry
ADD ENTRY
DELETE ENTRY
DASHBOARD
VIEW JOURNAL LIST

9.3 User Interface Flow Model



Tommy Works

10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the requirement has been met.

Requirement ID	Requirement Description	Component Modules	UI Elements	Low-Level Component	Testing Method
REQ001	Motivational Quote Generator	Motivational Quote Module	Quote Display Component	Quote API, Category Filters	Automated and manual testing to verify quotes are displayed correctly and filter functionality works
REQ002	Calendar View for Progress Tracking	Calendar View Module	Calendar Display Component	Mood Data Storage	Automated testing to ensure correct display of moods and manual testing for data update functionality

REQ003	Alarm Notification for User's Device	Alarm Module	Notification Component	System Alarm Service	Automated testing to verify notification triggers and manual testing for user experience
REQ004	Gmail Verification	Authentication Module	Verification UI	Gmail API	Automated testing to ensure successful verification process and manual testing for user experience
REQ005	Mood Tracking and Notes	Mood Tracking Module	Mood Tracking UI	Local Storage, Reminder	Automated and manual testing to verify UI functionality, data entry, visualization and reminders
REQ006	Dashboard for Overview	Dashboard Module	Dashboard UI	Data Visualization	Automated and manual testing to ensure correct display of overview information and user interaction

REQ007	Mood of the Day	Mood of the Day Module	Mood Display Component	Local Storage	Automated testing to ensure correct display of mood of the day and manual testing for data accuracy
REQ008	Medication Reminder	Reminder Module	Reminder UI	Alarm Service	Automated and manual testing to verify reminder triggers and user interaction
REQ009	Journaling	Journaling Module	Journaling UI	Local Storage	Automated and manual testing to verify journal entry functionality, data storage, and user interaction

(Everyone) Tommy

11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

1.) Software Requirements Specification (SRS): A document that outlines the functional and non-functional requirements of a software system to be developed.

2.) Firebase: A platform developed by Google for creating mobile and web applications. It provides various services including real-time databases, authentication, cloud messaging, and analytics.

3.) API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other. In this context, an API is used to integrate external services, such as the Quotes API, into the Mindful Life Companion app.

4.) Agile Methodology: An iterative approach to software development that emphasizes flexibility, collaboration, and customer feedback. It involves breaking down the development process into smaller increments called sprints, allowing for continuous improvement and adaptation.

5.) Material Design: A design language developed by Google that emphasizes principles of good design such as minimalism, typography, and grid-based layouts. It provides guidelines for creating visually appealing and intuitive user interfaces for Android apps.

(Everyone)

12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information

so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Brad Appleton <brad@bradapp.net> http://www.bradapp.net

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc

Group SRD