Bible Journaling Application

Easten Elefson

CST-452 Capstone Final Architectural Plan

Grand Canyon University

Instructor: Professor Mark Reha

Revision: 1

Date: April 27, 2025

**ABSTRACT**

The proposed project is a bible journaling app intended to aid and digitalize biblical notetaking and devotional journaling, while providing additional features to the user that facilitate personalized Bible study and journaling. The primary object of the application is to provide users with an intuitive interface for users to facilitate consistent engagement with scripture and provides additional features to encourage a personal and meaningful relationship with God. The app shouldn’t replace traditional journaling, but instead incorporate physical journaling and notetaking while providing the advantages of a digital landscape.

The app is envisioned to include a range of features to meet these goals. Key functionalities will encompass daily scripture readings, comprehensive annotation tools for personal note-taking and reflections, and the ability to create journal entries in an organized manner. Search capabilities will allow users to quickly find specific verses or topics. Additionally, AI technologies will be implemented to enhance user study sessions, and user authentication and authorization modules will ensure secure and personalized experiences.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 11/16/2024 | Easten | Initial Project Design |
| 4/27/2025 | Easten | End of Capstone 2 |
|  |  |  |

|  |
| --- |
|  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 4

Detailed High-Level Solution Design 5

Detailed Technical Design 6

Appendix A – Technical Issue and Risk Log 7

Appendix B – References 8

Appendix C – External Resources 9

**Design Introduction**

The Bible journaling app is designed to provide an intuitive and enriching experience for users who wish to document their spiritual journey. Developed using the Flutter framework, the app will be capable of running on Android 4.1 and higher. The application architecture follows a client-server model, leveraging Firebase for backend services, and includes authentication and real-time database management.

The application consists of an Android mobile device running the Flutter app, which comprises the presentation layer, the business logic layer, and the data access layer. The application leverages a cloud based Firebase backend, using technologies like Firebase Authentication for user verification and a Firebase Realtime Database for storing journal entries, reflections, and user data. This tier ensures data is securely accessed and managed, leveraging Google's cloud infrastructure for scalability and reliability.

The application also includes external APIs such as OpenAI for natural language processing, bible-api.com for accessing scripture content. All communications between the app and these services are conducted over HTTPS (Port 443) to maintain high security standards.This all contributes to create a intuitive to use mobile bible journaling application for users.

The Bible journaling app begins with user authentication, where users register or log in. Users can reset their password if they forget by requesting a password reset email and entering a new password. Once logged in, they will then be directed to the home page (see wireframe document for a detailed design), where they can create or edit journal and reflection entries using the app's interface. Additionally, users can request portions of scriptures or query the in app open ai tool and copy their responses into journal or reflection entries.

This design ensures a scalable, secure, and responsive application that meets user needs effectively. By leveraging Firebase's managed services, the app benefits from automated scalability, security features, and real-time data synchronization. The integration with external APIs enhance the user experience, offering additional functionalities such as in-app scripture integration and access to ai conversations. Flutter was chosen as the development framework guarantees to guarantee a high-performance, cross-platform user interface that provides a consistent experience across all supported Android devices.

**External Documents References**

The following table lists the external documents and references that are relevant to the design and development of the Bible journaling app. These documents provide essential guidelines, technical specifications, and API documentation that are critical for the integration and implementation of various components within the application.

|  |  |  |
| --- | --- | --- |
| File Name | Purpose | Description |
| ComponentDesign.csv | Contains detailed information on the component design of the Bible journaling app, including each component's purpose and the associated API calls. | This file includes a comprehensive breakdown of all components used in the Bible journaling app, their purpose, and the API calls made to integrate the app with external services like OpenAI, api-bible.com, and Firebase. |
| BibleJournalingWireframes-Sitemap-Logical-Physical.pdf | Contains the sitemap, wireframes, and logical/physical diagrams for the Bible journaling app. | This document provides the visual and structural representation of the app’s components, including detailed wireframes, sitemap, and diagrams outlining the logical and physical architecture of the app. |

**Detailed High-Level Solution Design**

The high-level solution design outlines the architectural framework and key components required to develop and implement the Bible journaling app. This section provides an overview of the software and hardware technologies, frameworks, and third-party libraries that will be utilized, as well as the proof of concepts (POCs) necessary to validate the chosen solutions. The aim is to present a cohesive and detailed plan that ensures the application meets its functional and non-functional requirements, offering a robust, secure, and user-friendly experience. Readers can expect to gain insight into the technical direction of the project and the foundational elements that support the development and deployment of the app.

**High-Level Logical Solution Design**

The Bible journaling app aims to create a seamless, secure, and user-friendly experience for individuals documenting their spiritual journey. Leveraging the Flutter framework for cross-platform mobile development, while ensuring a responsive and intuitive interface. The design incorporates backend services using Firebase for authentication and data management, ensuring real-time synchronization and data security. Additionally, the integration of external APIs over HTTPS (port 443), such OpenAI, and api-bible, enriches the app's functionality, offering features like email notifications, AI chatbot integration, and scripture content access. This approach also includes support for the security non-functional requirements of the application.

The primary non-functional requirement for the Bible journaling app is the encryption of user notes and login credentials to ensure data privacy and security. All data saved in the Firebase Realtime Database will be encrypted before being stored. Flutter's encryption libraries (such as the encrypt package) will be utilized to implement this encryption, while Firebase Security Rules will ensure that only authorized users can access their own encrypted data.

For login credentials encryption, we will rely on Firebase Authentication, which securely handles user credentials using OAuth 2.0 and hashed passwords for login. This service provides built-in encryption to protect users' sensitive information. This approach ensures that both user notes and login credentials are securely encrypted, meeting the app’s privacy and security requirements.

In order to develop this application, a development environment for the Bible journaling app will be set up using VS Code with the necessary extensions to support Flutter development, including Dart and Flutter plugins for efficient coding, debugging, and testing. An Android Emulator will be used for testing the app on virtual devices. Although a GitHub repository is planned for version control and collaboration, it has not yet been implemented. This setup will provide a streamlined environment for building, testing, and refining the app during development.

Refer to the Logical Solution Design section for a detailed overview of the system's architecture and configuration, including the presentation, business logic, and data access layers, along with their interactions and integrations with external APIs.

For a comprehensive view of the system's deployment and physical infrastructure and configuration, please see the Physical Solution Design section, which outlines the hardware components, network configurations, and the environment in which the system operates.

**High-Level Solution Design for Hardware and Software Technologies**

The Bible journaling app will leverage a combination of modern software frameworks, third-party libraries, and reliable hardware components to ensure a secure, efficient, and user-friendly solution. Below is a detailed inventory of the technologies and frameworks that will be used, as well as a table of proof of concepts (POCs) that will be completed to validate these technologies, as not all of them have been completed.

Software Frameworks and Libraries:

|  |  |  |
| --- | --- | --- |
| Technology | Description | Rationale |
| Flutter | Framework for building cross-platform mobile applications. | Provides a rich set of pre-designed widgets and tools, supports rapid development, and ensures compatibility across a wide range of Android devices. |
| Firebase Authentication | Service for user authentication and management. | Simplifies the implementation of secure user authentication. |
| Firebase Realtime Database | Cloud-hosted NoSQL database for storing user notes and other data. | Provides real-time synchronization and ease of use with Firebase SDK. |
| Firebase Cloud Firestore | Flexible, scalable database for app data. | Offers real-time data synchronization and offline support. |
| OpenAI API | API for natural language processing tasks. | Enhances user experience by providing reflective prompts and sentiment analysis. |
| Bible-api | API for accessing Bible verses and scripture content. | Provides easy integration of scripture content into the app. |
| Encrypt Package | Flutter package for data encryption. | Ensures secure storage and transmission of user notes. |

Hardware Components:

|  |  |  |
| --- | --- | --- |
| Hardware | Description | Rationale |
| Android Mobile Devices | Devices running Android OS used for testing and deployment. | Ensure compatibility and performance across different device models and Android versions. |
| Development Machines | Laptops or desktops used by developers. | Equipped with VS Code, Android Studio, and necessary SDKs for development. |
| Cloud Infrastructure | Google Cloud Platform services supporting Firebase. | Provides scalable, reliable, and managed services for backend operations. |

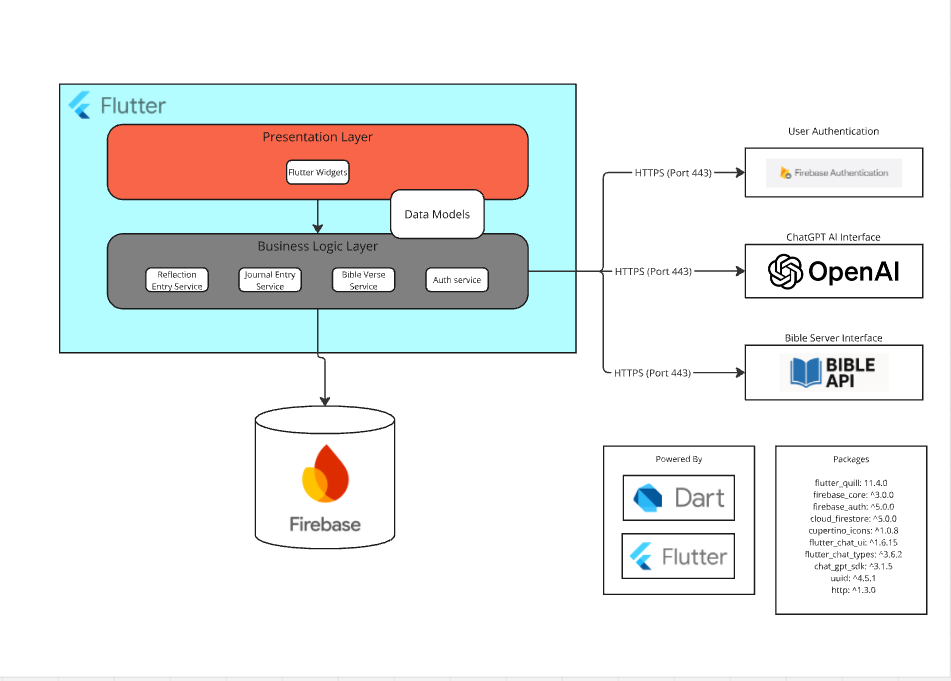
Proof of Concepts:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Description | Rationale | Results | | 1. Flutter Integration | Ensure Flutter framework supports mobile design for Android devices while allowing support for iOS devices in the future. | The flutter framework meets the requirements for mobile development neatly. | | 2. Firebase Authentication | Validate secure user authentication, including registration, login, and session management. | N/A | | 3. Firebase Realtime Database | Test real-time data synchronization and offline capabilities for user notes and data. | N/A | | 4. OpenAI API Integration | Confirm the API’s ability to provide relevant reflective prompts. | N/A | | 5. API.bible Integration | Verify the ability to retrieve specific sections of scripture in different biblical versions. | N/A | | 6. Mailgun API Integration | Test the ability to send email notifications to reset a password. | N/A | | 7. Data Encryption with Encrypt Package | Ensure secure encryption and decryption of user notes and passwords stored in the database. | N/A | |
|  |
|  |
|  |
|  |
|  |
|  |

This detailed inventory of hardware and software technologies, along with the planned POCs, provides a comprehensive roadmap for developing and validating the Bible journaling app. Each POC will ensure that the selected technologies and frameworks are fit for purpose, cost-effective, and capable of solving the specific problems addressed by the app.

**Logical Solution Design**

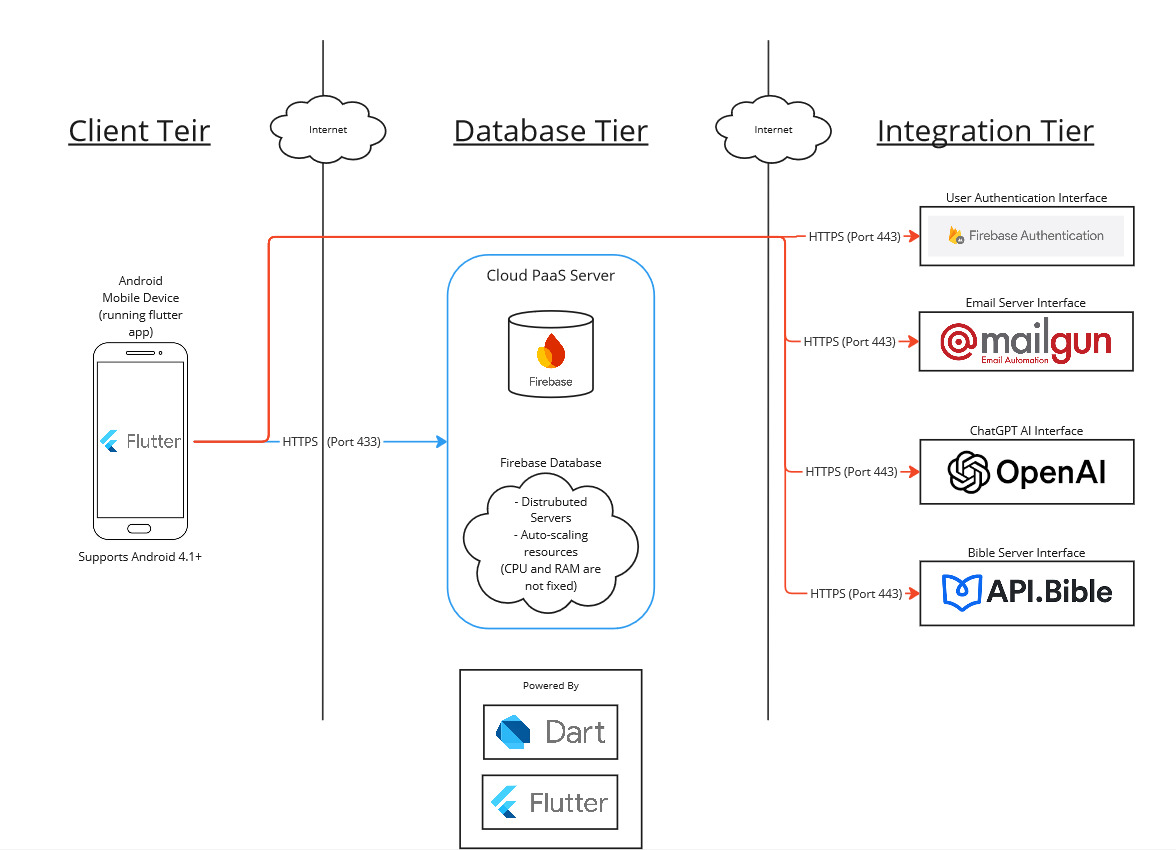
The Logical Solution Design diagram provides a high-level overview of the architecture and key components of the Bible journaling app. It illustrates the interactions between different layers of the application, including the presentation layer, business logic layer, data access layer, and external services. The diagram aims to demonstrate how these components collaborate to deliver the app's functionality, ensuring a clear understanding of the system's structure and flow for developers and stakeholders.



The logical architecture diagram showcases a layered design using Flutter. It highlights the separation of concerns into two layers: Presentation and Business Logic. The Presentation Layer consists of Flutter widgets for user interaction. The Business Logic Layer contains services such as Reflection Entry, Journal Entry, Bible Verse, and Auth Service. The Data Access Layer manages the database interactions using Firebase and DAO objects. The diagram also demonstrates external integrations, including Firebase for authentication, Mailgun for emails, OpenAI for AI interactions, and API.Bible for scripture data, all secured through HTTPS over port 443.

**Physical Solution Design**

The Physical Solution Design diagram presents a detailed view of the hardware and software infrastructure required to support the Bible journaling app. It includes the client devices, servers, databases, and external services, highlighting the physical connections and interactions between them. This diagram aims to provide a comprehensive understanding of the deployment environment and the physical architecture, ensuring that all components are correctly configured and integrated to deliver a reliable and efficient solution.



The physical architecture diagram illustrates the deployment of the app’s components across different tiers. The Client Tier represents Android devices running the Flutter application. The Database Tier is powered by Firebase, providing a distributed, auto-scaling backend for storage and computation. The Integration Tier connects the app to external services, including Firebase for authentication, OpenAI for AI interactions, and bible-api for scripture data. All communication is secured through HTTPS over port 443.

Detailed Technical Design

**General Technical Approach:**

The Bible journaling app's technical design prioritizes modularity, scalability, and secure integrations to ensure an optimal user experience and reliable performance. The design follows a layered architecture, dividing the system into Presentation, Business Logic, and Data Access layers to promote separation of concerns and easier maintainability. The development process involved brainstorming sessions, in which multiple database types and scripture api’s were considered. In the end, Firebase was chosen as the backend platform for its scalability, reliability, and seamless integration with Flutter. Additionally, external APIs, such as OpenAI for AI-powered features and API.Bible for scripture data, were integrated to extend the app’s functionality. All components communicate securely via HTTPS, and security measures such as encryption and Firebase Authentication were incorporated to protect user data.

**Key Technical Design Decisions:**

The Bible Journaling app has several key technical design decisions. These include, but are not limited to, Flutter for the front end app, Firebase for the backend, and external APIs, such as OpenAI for AI-powered features and bible-api. Additionally, developmental technologies like VS Code and Android emulators are used to provide a development environment for the team. For a comprehensive list of technologies and frameworks, as well as their purpose in the design and why they were chosen, see the Detailed High Level Solution section of this document. This section of the document also provides the needed proof of concepts and their rationale and purpose.

**Schema Design:**

**Firestore Collection Design:**

**1. Users Collection (users)**

Users are handled by Firebase authentication and the backend is hidden from the developer and automatically handled.)

**2. Reflections Collection**

This subcollection stores reflection entries related to each user. These are personal thoughts and insights related to a specific days worth of studying

* + journalEntryID: reflectionEntryID (auto-generated or custom)
  + reflectionText: String (user’s written reflection on the verse)
  + tldr: String (quick summary of reflection purpose)
  + createdAt: Timestamp (when the reflection was added)
  + updatedAt: Timestamp (When the reflection as last updated)
  + userId: string (Id of the associated user)
  + pinned: Boolean (whether or not a current reflection is pinned)

**3. JournalEntries Collection (journal\_entries)**

This collection stores the journal entries where users can write about their thoughts or experiences, including references to scripture.

* + Document ID: journalEntryID (auto-generated or custom)
  + userID: String (reference to the user who created the journal entry)
  + title: String (title of the journal entry)
  + content: String (content of the journal entry)
  + createdAt: Timestamp (when the journal entry was created)
  + updatedAt: Timestamp (when the journal was last updated)
  + tags: [] (An array of tags)

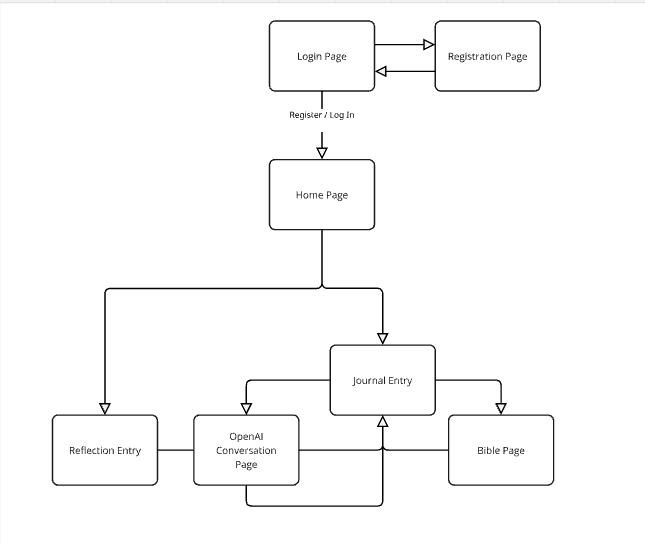
**4. Verses Collection (pinned\_verses)**

This collection stores Bible verses that can be referenced and pinned by users.

* Document ID: verseID (auto-generated or custom)
  + userID: String (reference to the user who pinned the verse)
  + book: String (book of the Bible, e.g., "John")
  + chapter: Number (chapter number)
  + verse: Number (verse number)
  + text: String (the content of the verse)
  + pinned: Boolean (true if the user has pinned this verse to their dashboard)

**Sitemap Diagram:**

A sitemap is an essential tool for organizing the structure and navigation of the Bible journaling app, ensuring a seamless user experience. It outlines the hierarchy and relationship between different screens and features, helping developers visualize the app's flow while guiding users through its functionalities. The following sitemap provides a clear blueprint for the app's layout, enabling efficient development and intuitive user interaction.



The sitemap begins with a Login Page that links to a Registration Page for new users. After successful login or registration, users access the Home Page, which serves as the central hub. From the Home Page, users can navigate to two key pages: Reflection Entry and then Journal Entry, which navigates to Bible Page, and OpenAI Conversation Page, allowing seamless access to all app features.

**User Interface Diagrams:**

Wireframes are a crucial step in the design process, providing a visual representation of the app’s layout and functionality. They help outline the structure, placement of elements, and overall user flow, ensuring a clear and consistent design. The following wireframes illustrate the primary screens of the Bible journaling app, offering a foundational guide for developers and stakeholders to visualize the app before implementation. This ensures alignment with user needs and project goals while streamlining the development process. To view the wireframe, see BibleJournalingWireframes-Sitemap-Logical-Physical.pdf, listed in the External Documents References section of this document.

**Service API Design:**

Firebase Authentication API:

The Firebase Authentication API is used for secure user authentication, providing functionality for user registration, login, and logout. The API is accessed through the Firebase SDK, integrated within the Flutter application. Key methods include createUserWithEmailAndPassword, signInWithEmailAndPassword, and signOut. The API operates over HTTPS and requires parameters such as the user’s email and password for authentication requests. Responses include unique user identifiers (UID) and authentication tokens for session management. These tokens are essential for maintaining secure access to user-specific data and ensuring compliance with security best practices.

OpenAI API:

The OpenAI API provides AI-driven features that enhance user interaction, such as generating personalized reflections and insights. This service is accessed through the /v1/completions endpoint, secured with an API key. Parameters include model to specify the AI engine, prompt to define the input query, max\_tokens to control the length of the response, and temperature to adjust the creativity of the output. The API returns structured text data, which can be rendered directly within the app. This integration allows for dynamic content generation, enriching the journaling experience with personalized, AI-powered insights.

API.Bible:

The Bible-API service integrates scripture data into the application, enabling features like Bible study and journaling. The primary endpoint for accessing scripture is https://bible-api.com/{Book}+{Chapter}:{Verse}, requiring no authentication for public access. Parameters include the book name, chapter number, and verse number, formatted into the URL to specify the desired scripture reference. The API responds with structured JSON data containing the verse text, reference, translation information, and additional metadata as needed. This ensures the app delivers accurate and easily accessible scripture content, supporting its spiritual and educational objectives.

**NFR’s (Security Design):**

The Bible journaling app incorporates the Non-Functional Requirement: Data Encryption and Password Security through the integration of the Flutter encrypt package to ensure robust data protection. User notes and reflections are encrypted before storage in the database using AES encryption provided by the encrypt package, ensuring sensitive notes remains secure and unreadable without the appropriate decryption key. This encryption process is implemented in the Data Access Layer to standardize secure data handling.

Firebase Authentication is used to manage token-based authentication and encrypted data transmission, further enhancing user security.

**Operational Support Design:**

The Bible journaling app’s design allows for the potential integration of robust operational support features to enable efficient monitoring, logging, and maintenance in a production environment. These capabilities can help ensure system reliability, quick identification of issues, and minimal disruption to users.

**Monitoring:**

Firebase offers tools like Performance Monitoring and Crashlytics, which can be utilized to track metrics such as app start time, screen load time, and network request latency, as well as capture runtime errors and crashes. While not required, these tools are available for seamless integration into the app if needed to provide continuous performance insights in production.

**Logging:**

The app can incorporate structured logging using Firebase’s logging capabilities or third-party tools such as **Sentry** for managing log data. This would allow for the recording of key application events, such as user authentication and API interactions, while ensuring compliance with data protection standards. Logs could be stored securely and queried for analysis if operational logging is prioritized during production.

**Maintenance and Updates:**

The app’s modular architecture supports maintainability by isolating functionality into distinct layers (Presentation, Business Logic, and Data Access). This design allows for the possibility of implementing features like Firebase Remote Config to enable remote updates to app behavior or feature toggles, reducing the need for app re-deployment when adjustments are required.

**Alerting and Issue Resolution**

Tools like Crashlytics or similar alternatives could be configured to provide real-time alerts for critical issues. Detailed error reporting, including device information and activity logs, could assist in prompt resolution. These features are optional and can be added based on the specific operational needs of the app.

**Mobile Device And App Store Support:**

The Bible journaling app is designed to support Android devices running version 4.1 (Jelly Bean) and higher, ensuring compatibility with a wide range of devices. While the app may be deployed directly to users, there is the potential for distribution through the Google Play Store, providing an accessible platform for app discovery and downloads.

**Other Documentation:**

N/A

Component Design

In the development of the Bible Journaling application, a modular approach to component design was employed to ensure flexibility, scalability, and ease of maintenance. Each component serves a specific function, enabling users to interact with the app in an intuitive and seamless manner. The design incorporates a variety of components, ranging from journal entry management tools to AI-powered features, all interacting with APIs such as OpenAI, API.Bible, Mailgun, and Firebase Authentication. The attached ComponentDesign csv file located in the External Documents Reference section of this document outlines each unique component, its purpose, and the associated API calls that drive the app's functionality. The attached wireframe found in the External Documents Reference section of the document maps these unique components to certain pages using the wireframe.

Appendix A – Technical Issue and Risk Log

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Appendix B – References

API.bible. (n.d.). API.bible - *The easiest way to integrate the Bible into your app or website*. Retrieved November 17, 2024, from <https://api.bible>

Firebase. (n.d.). Firebase - *App success made simple*. Retrieved November 17, 2024, from <https://firebase.google.com>

Flutter. (n.d.). Flutter - *Build apps for any screen*. Retrieved November 17, 2024, from <https://flutter.dev>

Mailgun. (n.d.). Mailgun - *Transactional email API service for developers*. Retrieved November 17, 2024, from <https://www.mailgun.com>

OpenAI. (n.d.). OpenAI - *Powerful AI models and tools for developers*. Retrieved November 17, 2024, from <https://openai.com>

pub.dev. (n.d.). Encrypt - *Simple encryption and decryption for Dart*. Retrieved November 17, 2024, from https://pub.dev/packages/encrypt

Appendix C – External Resources

|  |  |
| --- | --- |
| **GIT URL:** | *Not currently applicable (may be in the future).* |