XBCAD7319

GROUP NAME: WORKFORCE

GROUP MEMBERS:

KARABO DLAMINI ST10139672

SIBONELO LUKHELE ST10092074

BOITSHEPO MAZABANE ST10216500

SCRUM BOARD: [XBCAD Scrum board - Backlog - Jira](https://karabongobeni12.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog)

GIT HUB REPO: <https://github.com/st10216500/XBcadWorkforce-.git>

Table of Contents

[1. INTRODUCTION TO THE PROJECT 4](#_Toc183118964)

[1.1. Purpose of the application: 4](#_Toc183118965)

[1.2. Client overview: 4](#_Toc183118966)

[1.3. Ethical and privacy concerns: 4](#_Toc183118967)

[1.4. Work agreement: 5](#_Toc183118968)

[1.5. Roles: 5](#_Toc183118969)

[1.6. Definition of Ready (DoR) 5](#_Toc183118970)

[1.7. Definition of Done(DoD): 6](#_Toc183118971)

[1.8. Sprint and Project Planning: 6](#_Toc183118972)

[1.8.1. High Level Plan: 6](#_Toc183118973)

[1.9. RISK IDENTIFICATION AND MITIGATION 9](#_Toc183118974)

[2. REQUIREMENTS 10](#_Toc183118975)

[2.1. Church Requirements: 10](#_Toc183118976)

[2.2. User Roles: 10](#_Toc183118977)

[2.3. User Stories: 11](#_Toc183118978)

[2.4. Research: 11](#_Toc183118979)

[2.4.1. Overview of Church and Bible Apps: 11](#_Toc183118980)

[2.4.2. You Version Bible App: 11](#_Toc183118981)

[2.4.3. Tithe.ly Church App: 12](#_Toc183118982)

[2.4.4. ChurchSuite 13](#_Toc183118983)

[2.5. User Experience Journey Map: 13](#_Toc183118984)

[3. NON-FUNCTIONAL REQUIREMENTS 14](#_Toc183118985)

[3.1. Non-Functional requirements: 14](#_Toc183118986)

[2.3.2. Functional Requirements: 17](#_Toc183118987)

[4. ANALYSIS ARTIFACTS 18](#_Toc183118988)

[4.1. Domain Modelling: 18](#_Toc183118989)

[4.2. Design Artifacts: 18](#_Toc183118990)

[5. IMPLEMENTATION DOCUMENTATION 19](#_Toc183118991)

[5.1. UML Sequence Diagram: 19](#_Toc183118992)

[5.2. Class Diagram: 20](#_Toc183118993)

[6. DATA SCHEMA DOCUMENTATION: 21](#_Toc183118994)

[7. ARCHITECTURE ARTIFACTS: 22](#_Toc183118995)

[7.1. Design patterns: 23](#_Toc183118996)

[7.2. Repository pattern: 23](#_Toc183118997)

[7.3. Cloud Architecture: 24](#_Toc183118998)

[7.4. Key architechtural decisions: 25](#_Toc183118999)

[7.5. Advantages of chosen architecture: 25](#_Toc183119000)

[8. SECURITY 26](#_Toc183119001)

[8.1. Authentication and authorization: 26](#_Toc183119002)

[8.2. Firebase security rules: 27](#_Toc183119003)

[8.3. Data encryption: 27](#_Toc183119004)

[8.4. Data and privacy compliance: 27](#_Toc183119005)

[8.5. Threat analysis and mitigation: 28](#_Toc183119006)

[8.6. Data back up and recovery: 28](#_Toc183119007)

[8.7. Error and security logging: 28](#_Toc183119008)

[9. DEVOPS 29](#_Toc183119009)

[9.1. Pipeline overview: 29](#_Toc183119010)

[9.2. Pipeline steps: 29](#_Toc183119011)

[9.3. Flow diagram: 30](#_Toc183119012)

[9.4. Automation strategy: 30](#_Toc183119013)

[9.5. Security testing in pipeline: 31](#_Toc183119014)

[10. RUNNING COSTS 31](#_Toc183119015)

[10.1. Predicted monthly costs: 32](#_Toc183119016)

[10.2. Technology scaling and replacement plan: 33](#_Toc183119017)

[11. CHANGE MANAGEMENT 33](#_Toc183119018)

[11.1. Organisational Adoption strategy: 33](#_Toc183119019)

[11.2. User adoption strategy: 34](#_Toc183119020)

[11.3. Maintainance and support strategy: 35](#_Toc183119021)

[11.4. Risk management: 35](#_Toc183119022)

[12. APPENDICES 36](#_Toc183119023)

[12.1. Sprint Minutes: 36](#_Toc183119024)

[References 40](#_Toc183119025)

## INTRODUCTION TO THE PROJECT

This document outlines the Software Requirements Specifications (SRS) for the development of a mobile application for Independent Church Of Christ(ICC), a small but growing church in the Zion Christian denomination. The purpose of this document is to define the functional and non-functional requirements of the app, provide insight into the research conducted for similar church apps, and outline the Agile process our team followed during the planning, development, and sprints.

### Purpose of the application:

The ICC Church app will digitize essential church operations such as financial contributions, event management, and sermon notes. Additionally, it will provide features such as a Bible with verse highlighting and a section for sermon notes, which will allow users to have access to them again in future.

### Client overview:

Our client, ICC Church, is based in Marshalltown, JHB CBD, and is led by Presiding Bishop T. Dlamini. The leadership structure consists of 29 individuals, including deacons, youth leaders, and the church secretary. The church has been experiencing difficulties in tracking finances and managing memberships due to manual processes. This app aims to alleviate these issues by offering a streamlined and efficient digital solution for both church members and leadership.

### Ethical and privacy concerns:

As we collect and process sensitive information, including financial data and personal details, the development team is aware of the ethical and privacy concerns involved. The app will require robust encryption and data privacy measures to ensure secure storage and transmission of information. We will also implement strong user authentication protocols (Valis, 2024) to safeguard user accounts.

### Work agreement:

The following principles outline the guidelines that our team (Karabo Dlamini, Sibonelo Lukhele, and Boitshepo Mazabane) has agreed to follow to ensure smooth collaboration throughout the development of the ICC Church App:

* We will meet every Monday, Wednesday, and Friday at 8:00 AM for a 15-minute stand-up meeting to discuss progress, blockers, and next steps.
* We will proactively identify problems and dependencies early on and communicate them with the team in a timely manner.
* We will review the output of every sprint in detail and treat every sprint review as if the client is participating, even if they are not present.
* We will give constructive feedback that is honest yet respectful and will not be aggressive when reviewing each other's work.
* We will assist each other if anyone is facing difficulty with their assigned tasks, ensuring a supportive team environment.
* Each team member is responsible for updating the team on their progess status in the 3 weekly meetings we will have.
* We will conduct retrospectives at the end of each sprint to assess our performance and adherence to this work agreement, and to identify opportunities for improvement.

### Roles:

**Karabo Dlamini (Scrum Master/Backend Developer)**: Oversees the Agile-Scrum process, manages the backend development, and ensures team collaboration.

**Sibonelo (Frontend Developer)**: Leads UI/UX development, designs user journeys, and ensures smooth frontend operation.

**Boitshepo (DevOps/Research Lead)**: Responsible for DevOps pipeline and research for critical components (e.g., SDK versions, libraries).

While these are the assigned roles all members will be flexible meaning all roles will be interchangeable during the course of this project.

### Definition of Ready (DoR)

Our team has agreed that the following criteria must be met before a user story or task is considered “ready” to begin:

* User story is fully written with detailed acceptance criteria.
* The task is small enough to be completed within a sprint.
* The story has been estimated for complexity (story points).
* All necessary resources (e.g., wireframes, data models) are available.
* Dependencies on external teams or services have been identified and mitigated.
* Any technical risks have been analyzed and documented.

### Definition of Done(DoD):

A user story or task will be considered “done” when:

* All acceptance criteria have been fulfilled.
* Code has been reviewed by at least one other developer and passes all tests.
* Any new features are documented in the user manual and technical documentation.
* No known bugs are left unresolved.
* The feature is integrated into the main branch and passes CI/CD pipeline checks.
* All work has been successfully demonstrated to the client, or a representative.

### Sprint and Project Planning:

### 1.8.1. High Level Plan:

Below is the the team’s high-level Agile plan, outlining the sprints and lifecycle phases of the ICC Church App:

**Sprint 1: Concept and Team Formation**

* **Agile Lifecycle Phase:** Concept
* **Sprint Goal:** Establish the project team and outline collaborative agreements, defining the Work Agreement, Definition of Ready (DoR), and Definition of Done (DoD).

**Backlog Items:**

1. Collaboratively define our Work Agreement principles to promote effective communication and support among team members.
2. Create clear DoR and DoD criteria to streamline task management and completion.
3. Identify individual roles and responsibilities to ensure accountability within the team.

**Sprint 2: Requirement Gathering**

* **Agile Lifecycle Phase:** Inception
* **Sprint Goal:** Engage with the client to collect detailed requirements, formulating user stories and sketching out the initial user journey.

**Backlog Items:**

1. Collaborate with the client to compile a comprehensive list of application requirements.
2. Draft user stories that outline expected functionalities and user interactions.
3. Initiate discussions on potential technology choices to support feature development.

**Sprint 3: Analysis and Design**

* **Agile Lifecycle Phase:** Iteration
* **Sprint Goal:** Conduct a thorough analysis of the domain,research on available SDKs and APIs , finalize the user journey diagram, and establish the technical architecture and design mock-ups.

**Backlog Items:**

1. Perform a domain analysis to identify essential features and workflows.
2. Research Available SDKs and APIs for the bible intergration
3. Complete the user journey diagram, capturing all user interactions and flow.
4. Design initial mock-ups to visualize the app’s layout and functionality for client feedback.

**Sprint 4: Core Feature Development**

* **Agile Lifecycle Phase:** Iteration
* **Sprint Goal:** Develop and implement core functionalities such as user accounts, Bible access, and sermon notes.

**Backlog Items:**

1. Build the user account creation and authentication system.
2. Implement the Bible feature, allowing users to search and highlight verses.
3. Develop the sermon notes functionality, including options for public and private visibility.

**Sprint 5: Feature Completion and Integration**

* **Agile Lifecycle Phase:** Iteration
* **Sprint Goal:** Finalize the remaining features, including financial contributions and notifications, ensuring all components work cohesively.

**Backlog Items:**

1. Complete the financial contributions tracking system with appropriate verification processes.
2. Integrate notification systems for announcements and contributions.
3. Conduct integration testing to confirm all components function correctly together.

**Sprint 6: Final Review and Presentation**

* **Agile Lifecycle Phase:** Maintain
* **Sprint Goal:** Prepare for the final presentation, ensuring the application is polished and documentation is complete.

**Backlog Items:**

1. Create a comprehensive final presentation highlighting the app’s functionalities and value to the client.
2. Perform final testing to ensure a stable and functional application.
3. Finalize project documentation, including technical specs and user guides, for client handover.

### RISK IDENTIFICATION AND MITIGATION

**Potential Risks Identified:**

* **Data Privacy**: Given that sensitive information like membership details and contributions will be tracked, there is a risk of data breaches.
  + **Mitigation**: We will implement robust security measures, such as encryption, token-based authentication, and regular security audits.
* **Client Availability**: If the client is unavailable for feedback, there might be delays in refining the product.
  + **Mitigation**: We will maintain regular communication channels and set strict deadlines for feedback loops.
* **Technological Challenges**: Integration of Bible and sermon notes features could face challenges due to compatibility issues.
  + **Mitigation**: Thorough research on available SDKs and APIs was conducted in Sprint 3 to select the best-suited solutions.

## REQUIREMENTS

## 2.1. Church Requirements:

The Independent Church of Christ (ICC) requires a mobile application that digitizes key operational tasks while maintaining a simple and user-friendly interface. The app must provide the following functionalities:

1. **Membership Management**
   * Allow members to accept church constitution before registration.
   * Enable leadership to post announcements.
2. **Financial Contributions Tracking**
   * Allow members to view their financial contributions, including donations for specific events such as Good Friday.
   * Enable leadership to verify, approve, and manage contributions securely.
3. **Event Management and Notifications**
   * Provide real-time notifications for event reminders and announcements.
4. **Bible Access and Sermon Notes**
   * Provide members with access to an online Bible via a WebView, including highlighting verses.
   * Allow members to write sermon notes, which can be saved privately or shared publicly with other users.
5. **Leadership-Only Features**
   * Restricted access for leaders to manage church funds and track contributions.
   * Ability to create and send announcements to members.
6. **Offline Functionality**
   * Provide members with the ability to access core features like contribution history and sermon notes without an internet connection, syncing data when reconnected.

## User Roles:

**Member**:

* Registers and views personal financial contributions.
* Accesses the Bible and takes sermon notes.

**Leadership**:

* Manages events and announcements.
* Tracks and verifies contributions.

## User Stories:

* + As a member, I want to view my contribution history so that I can ensure I’ve met my commitments.
  + As leadership, I want to send event reminders so that members stay informed.

## Research:

### Overview of Church and Bible Apps:

We researched several church and Bible apps to gain insights into their features, strengths, and weaknesses. This analysis has helped us identify best practices for developing the ICC Church app.

### You Version Bible App:

**Overview:**

YouVersion is one of the most popular Bible apps globally, offering an extensive range of Bible translations, reading plans, and study tools.

**Strengths:**

Bible Reading Tools:The app allows users to highlight, bookmark, and make notes on Bible verses.

Community Engagement: Users can share highlighted verses and notes with friends and study groups.

Offline Access: Users can download Bible translations and reading plans for offline access.

**Weaknesses:**

Overwhelming Features: The large number of features can be intimidating for users seeking a simple Bible app.

Sermon Support: There is no dedicated space for taking sermon notes or associating them with specific dates.

**Best Features:**

- Excellent Bible highlighting and bookmarking features.

- Community sharing options for verses and notes.

### Tithe.ly Church App:

**Overview:**

Tithe.ly is focused on providing an all-in-one solution for church giving, event management, and sermon viewing.

**Strengths:**

Donation Tools: Tithe.ly simplifies the process of donating, supporting both one-time and recurring donations.

Event Management: It allows members to RSVP to church events and see event details.

Mobile Notifications: Sends real-time notifications for contributions and announcements.

**Weaknesses:**

Bible Feature:The app lacks a Bible integration feature, which would benefit users interested in reading Scripture.

Limited Interaction: While good for financial transactions, it lacks interactive community engagement tools like sermon notes.

**Best Features:**

- Simple and effective donation process.

- Strong event management and notification tools.

### ChurchSuite

**Overview:**

ChurchSuite is a comprehensive church management software that caters to various administrative needs, including event management and membership tracking.

**Strengths:**

Membership Management: Offers detailed membership renewal and tracking.

Event Management: Users can RSVP to events, and admins can track attendance and financial contributions.

Financial Reporting: Supports event-specific contributions and automatic tallying.

**Weaknesses:**

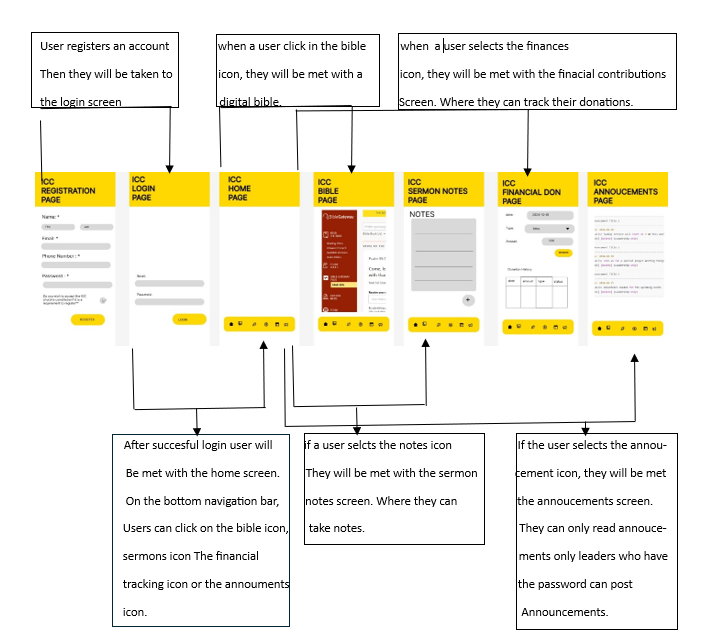
Complexity: Its wide range of features can be overwhelming for smaller congregations.

Restricted Access: Only leadership can add announcements, limiting member involvement.

**Best Features:**

* + Advanced event contribution and financial tracking.
  + Strong administrative tools for church leadership.

## User Experience Journey Map:



## NON-FUNCTIONAL REQUIREMENTS

### Non-Functional requirements:

**1. Performance Requirements**

The ICC App ensures efficient performance for all users by addressing the following requirements:

* Response Time: The app must respond to user interactions, such as financial contribution tracking, Bible access, and event registration, within 3 seconds under normal operating conditions (altexsoft, 2023).
* Load Handling: The app must handle up to 500 concurrent users, ensuring smooth performance even during high-traffic church events like Good Friday.

**2. Scalability Requirements**

Scalability ensures the app can grow with the church's needs over time:

* Membership Growth: The app must accommodate an increasing number of user registrations, financial contributions, and event RSVPs without requiring major architectural changes.
* Data Growth: Firebase Firestore is used to store member data, financial records, sermon notes, and Bible highlights efficiently, ensuring seamless scaling as user-generated data increases.
* Cloud Infrastructure: Firebase's scalable backend dynamically handles varying traffic loads, particularly during high-usage periods.

**3. Security Requirements**

Security is a critical aspect of the ICC App due to the sensitive nature of the data being managed. The following measures ensure the protection of user data:

* Authentication and Authorization: Firebase Authentication ensures only registered and verified users can access the app. Leadership roles have restricted access to financial and event management features (altexsoft, 2023).
* Encryption:
  + At Rest: All data stored in Firebase is encrypted using AES-256 (altexsoft, 2023).
  + In Transit: Data transmitted between the app and Firebase is encrypted using TLS 1.2 to prevent interception.
* Data Privacy: The app complies with South Africa’s Protection of Personal Information Act (POPIA), ensuring secure handling of user data and providing users with control over their shared information (POPIA Act Compliance, 2024).

**4. Usability Requirements**

The app is designed for ease of use, ensuring accessibility for all members:

* Ease of Use: The interface prioritizes simplicity, with intuitive navigation, clear labels, and minimal steps to complete tasks like contributions or event registrations.
* Android-Only Compatibility: As the app is designed exclusively for Android, it is optimized to function seamlessly across a variety of Android devices, including low-end smartphones and high-end tablets.
* Consistency: The user experience (UX) remains uniform across all supported Android devices to reduce confusion and improve user satisfaction.

**5. Reliability Requirements**

Reliability ensures the ICC App is dependable and consistently available for all users:

* Uptime: The app must maintain a 99.9% availability rate, minimizing downtime and ensuring members can always access key features like the Bible and sermon notes.
* Data Backup: Firebase’s built-in data replication and backup mechanisms ensure the safety and availability of user data, including contributions and notes.
* Error Handling: The app provides clear and user-friendly error messages and logs errors for developer review to prevent recurring issues.

**6. Maintainability Requirements**

Maintainability ensures the app can be updated and enhanced over time without significant disruptions:

* Modular Design: The app follows a modular structure, allowing independent updates to features like event management, Bible access, or authentication without affecting other functionalities.
* Documentation: Comprehensive technical documentation, including code comments and API references, ensures smooth handovers and efficient updates by future developers.
* CI/CD Pipeline: Integration with GitHub Actions automates testing, builds, and deployments, ensuring reliable and efficient updates (GitLab, 2024).

**7. Portability Requirements**

* Android-Specific Design: The app is designed exclusively for Android devices and will not be ported to other operating systems, allowing the development team to focus on optimizing the app for Android’s ecosystem (altexsoft, 2023).
* Device Compatibility: The app supports a range of Android devices, from low-end smartphones to high-performance tablets, ensuring accessibility for all members.

**8. Legal and Ethical Requirements**

The ICC App adheres to all legal and ethical considerations relevant to its context:

* Ethical Data Use: All data collected, including financial contributions and personal details, is used transparently and ethically.
* Consent Management: Users must explicitly consent to data collection and can withdraw consent at any time, in compliance with POPIA regulations (POPIA Act Compliance, 2024).
* Terms of Use: The app includes clear terms of service, explaining user rights and responsibilities regarding the use of the app.

## 2.3.2. Functional Requirements:

1. **User Registration and Constitution Acceptance:**

- All users must accept the church constitution upon registration.

- If users decline the constitution, they cannot register.

1. **Membership Management:**

-Members can register, renew their membership every six months, and receive reminders for renewals.

3.**Financial Contributions Tracking:**

- Users can upload financial contributions, which will be verified by the church secretary before being reflected in the app.

4. **Bible Integration:**

- Users can access a digital Bible, highlight verses, and bookmark them for future reference.

5. **Sermon Notes:**

- Users can write sermon notes by date.

- Notes can be marked as private or public.

- Public notes will be visible to all users, while private notes will only be visible to the individual.

6. **Leadership-Only Section:**

- Certain features will be restricted to church leadership, such as event management and financial reporting.

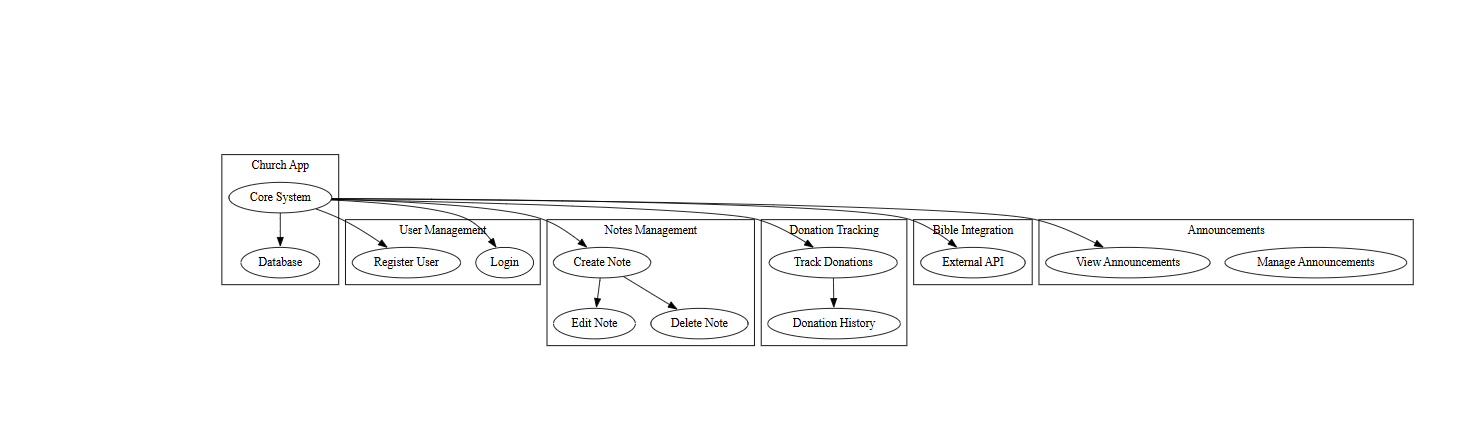
7. **Announcements and Notifications:**

- Announcements will be made by leadership and notifications sent to all app users.

## ANALYSIS ARTIFACTS

## Domain Modelling:

The domain model for the ICC App identifies the bounded contexts within the application and their respective roles. Each module is clearly defined with distinct responsibilities to ensure a modular and scalable design. The key contexts include User Management for registration and login, Notes Management for sermon notes handling, Donation Tracking for financial contributions, Bible Integration for external Bible access, and Announcements for managing church-wide communications. These bounded contexts interact seamlessly with the core system and database to maintain the app's functionality, as illustrated in the domain model diagram above.



## Design Artifacts:

The design of the ICC App translates the domain model into practical implementation models that guide development. Below are the steps and decisions taken to analyze and design the system effectively:

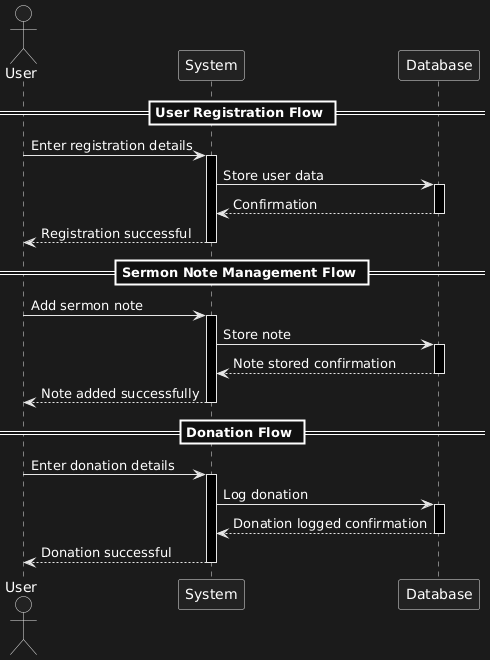
1. **User Management Design**
   * The User Management module was designed with Firebase Authentication for secure login and registration. The system ensures only authenticated users can access features based on their roles (e.g., members or leadership).
2. **Notes Management Design**
   * This module enables users to create, edit, and delete sermon notes. Firebase Firestore was chosen for efficient storage, with offline capabilities using the Room database for local caching.
3. **Donation Tracking Design**
   * The donation tracking system records contributions, providing leadership with a clear financial overview. Firebase Firestore handles data storage, and security rules enforce role-based access control for sensitive financial data.
4. **Bible Integration Design**
   * A WebView is used to integrate an external Bible API, allowing users to highlight verses and access scripture seamlessly without requiring extensive local storage.
5. **Announcements Design**
   * The Announcements module allows leadership to create and manage updates, while members can view them in real time. Firebase’s real-time capabilities ensure announcements are synchronized instantly.

## IMPLEMENTATION DOCUMENTATION

## UML Sequence Diagram:

The Sequence Diagrams depict key workflows in the app:

1. User Registration Flow: Validates and securely stores user data, confirming successful registration.
2. Sermon Note Management Flow: Processes note additions and stores them in the database with user confirmation.
3. Donation Flow: Logs financial contributions, providing confirmation to the user.



## Class Diagram:

The Class Diagram highlights the structural components of the ICC App. Key classes include:

1. App Class: Manages app initialization and navigation between pages.
2. Registration Page: Handles user registration with input validation and secure storage.
3. Login Page: Authenticates users based on provided credentials.
4. Home Page: Acts as the main dashboard linking to other features.
5. Sermon Notes Page: Enables users to create, edit, and delete notes.
6. Donations Page: Tracks and logs user financial contributions.
7. Bible Page: Integrates external Bible API for scripture access.
8. Announcements Page: Manages church-wide announcements for leadership.

A diagram of a website

Description automatically generated

## DATA SCHEMA DOCUMENTATION:

The Entity Relationship Diagram (ERD) (Hanna & Biscobing, 2024)below represents the data schema for the ICC App, detailing the relationships between key entities in the database. The primary entity, Users, is connected to Sermon Notes, Donations, and Announcements, capturing essential interactions within the app.

* Users: Stores user details such as name, email, phone number, and password, serving as the central entity.
* Sermon Notes: Links notes to specific users via user\_id, enabling personalized note management.
* Donations: Tracks financial contributions with attributes for type, amount, and status, linked to users via user\_id.
* Announcements: Manages church-wide communications, connecting announcements to users who create them.

A diagram of a computer

Description automatically generated

## ARCHITECTURE ARTIFACTS:

The architecture of the ICC App was designed to ensure scalability, security, maintainability, and usability, leveraging Firebase (Google, 2024) and Android-native technologies. The following sections provide a detailed overview of the architectural decisions and components used.

## Design patterns:

Design patterns are proven solutions to recurring problems in software design. The ICC App employs the following patterns to ensure modularity, scalability, and ease of maintenance:

1. **Model-View-ViewModel (MVVM):**
   * Purpose: Separates the user interface (UI) from the business logic, making the codebase easier to test, maintain, and scale (Wikipedia, 2024).
   * **Implementation in ICC App**:
     + Model Layer: Handles data operations. Firebase Firestore acts as the primary data source, storing user details, contributions, event records, and sermon notes.
     + View Layer: Represents the user interface, built with Android XML layouts. Each activity or fragment corresponds to a screen, such as the dashboard or Bible viewer.
     + ViewModel Layer: Acts as a mediator between the Model and View layers. For example, when a user accesses their financial contribution history, the ViewModel fetches the data from Firebase and provides it to the UI in the required format.
   * **Advantages**:
     + Simplifies testing by isolating business logic.
     + Enables easier updates to UI or backend logic without affecting the other layer.

## Repository pattern:

* + Purpose: Centralizes data handling by acting as a single source of truth for data access, whether from Firebase or local offline storage (GeekForGeek, 2024).
  + **Implementation in ICC App**:
    - A Repository class abstracts data operations, making it easier to switch between data sources (e.g., Firebase or Room database for offline mode).
    - For example, the repository handles fetching event data from Firebase while syncing local offline changes when connectivity is restored.
  + **Advantages**:
    - Decouples data sources from the rest of the app.
    - Enhances testability and maintainability.

## Cloud Architecture:

The ICC App leverages Firebase, a cloud-based backend-as-a-service (BaaS) platform, for its backend architecture. Firebase was chosen due to its seamless integration with Android and its suite of features tailored for mobile applications.

1. **Core Firebase Services Used:**
   * **Firebase Authentication:**
     + Provides secure user authentication.
     + Supports email/password login, with potential for future biometric authentication via Android's API.
     + Manages role-based access control, ensuring leadership members have access to restricted features like financial data and event management.
   * **Firebase Firestore (NoSQL Database):**
     + Acts as the primary data store for user information, contributions, events, and sermon notes.
     + Utilizes a NoSQL schema to store data in collections and documents for fast retrieval.

* **Firebase Storage:**
  + Used for storing any media files, such as event banners or sermon-related images.

**Application Data Flow:**

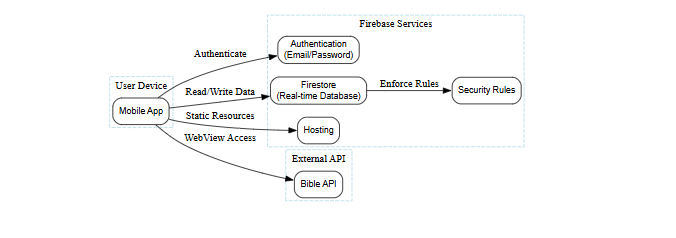
* + When a user interacts with the app (e.g., viewing their contribution history):
  1. The request is sent to the ViewModel.
  2. The ViewModel interacts with the Repository to fetch data.
  3. The Repository retrieves the data from Firestore or local storage, depending on network availability.
  4. The ViewModel processes and delivers the data to the UI for display.

**Real-Time Updates:**

* + Firebase Firestore’s real-time database functionality ensures that any updates to church events, announcements, or contributions are instantly reflected in the app without requiring manual refreshes.

**Security Rules in Firebase:**

* + Firebase Security Rules are implemented to enforce access control.
    - Regular members can only read their data.
    - Leadership members can write and manage contributions.



## Key architechtural decisions:

1. **Platform-Specific Development:**
   * The app is built exclusively for Android using native tools like Android Studio, XML for UI, and Kotlin for backend logic. This decision ensures optimal performance and a tailored experience for Android users.
2. **Single-Page Navigation:**
   * The app uses a navigation component to enable smooth transitions between pages (e.g., dashboard to Bible view).
3. **Scalability Through Firebase:**
   * Firebase automatically scales with user growth, ensuring the app can handle increased traffic without requiring server-side management.

## Advantages of chosen architecture:

1. **Scalability:**
   * Firebase's serverless infrastructure supports growth without additional management overhead.
   * NoSQL Firestore (Google, 2024) handles increasing data volumes efficiently.
2. **Security:**
   * Firebase Authentication and Security Rules ensure role-based access.
   * Data encryption safeguards user information.
3. **Efficiency:**
   * MVVM and Repository patterns ensure a clean and maintainable codebase.
   * Firebase’s real-time database reduces latency for critical updates.
4. **Cross-Integration:**
   * Seamless integration with Firebase Storage, Firestore, and Authentication reduces development complexity and provides a unified backend solution.

## SECURITY

The security of the ICC App is critical, as it handles sensitive user data such as financial contributions, personal information, and access permissions (Valis, 2024). The following section outlines the security measures and practices implemented in the app, based strictly on the information provided.

## Authentication and authorization:

1. **Firebase Authentication**
   * Firebase Authentication is used to securely verify users’ identities before granting access to the app (Google, 2024).
   * Users register and log in using their email and password, which are securely stored and managed by Firebase.
   * Firebase Authentication ensures that only verified users can access the app’s functionalities.
2. **Role-Based Access Control (RBAC)**
   * Different levels of access are defined for regular members and leadership:
     + **Members**: Can access general features such as viewing their financial contributions, using the Bible WebView, and taking sermon notes.
     + **Leadership**: Have additional permissions, such as managing contributions, creating event announcements, and verifying financial transactions.
   * Role-based permissions are enforced using Firebase Authentication and Firebase Security Rules, ensuring users only access data they are authorized to view or modify (Google, 2024).

## Firebase security rules:

* Firebase Security Rules govern how data is accessed and modified in the Firestore database.
  + Regular members can only access their data, such as their own contributions or personal information (Google, 2024).
  + Leadership members have broader access, such as adding or managing event announcements.

## Data encryption:

1. **Encryption at Rest**
   * All data stored in Firebase Firestore is encrypted using AES-256 encryption, ensuring data is protected even if the database is compromised.
2. **Encryption in Transit**
   * Data exchanged between the app and Firebase is encrypted using TLS 1.2, preventing interception or unauthorized access during communication.

## Data and privacy compliance:

1. **Protection of Personal Information Act (POPIA)**
   * The app complies with South Africa’s POPIA regulations, ensuring that user data is handled transparently and securely.
   * Key practices include:
     + Users are informed about what data is collected and how it will be used.
     + Users have the ability to withdraw consent at any time and request deletion of their data.
2. **User Consent**
   * Users provide explicit consent during registration for data collection and usage.
   * Terms of use and privacy policies are clearly displayed, ensuring users understand their rights and responsibilities.

## Threat analysis and mitigation:

The following threats were identified and mitigated during development:

1. **Unauthorized Access**
   * **Threat**: Unauthenticated users gaining access to sensitive data.
   * **Mitigation**: Firebase Authentication ensures only verified users can log in. Role-based permissions prevent unauthorized access to restricted features.
2. **Data Interception**
   * **Threat**: Interception of data during transmission.
   * **Mitigation**: TLS 1.2 encrypts all data in transit, ensuring it cannot be read or altered by unauthorized parties (Valis, 2024).
3. **Data Breaches**
   * **Threat**: Exposure of sensitive data if the database is compromised.
   * **Mitigation**: Firebase Firestore encrypts data at rest using AES-256, making it unreadable without the encryption keys.
4. **Impersonation Attacks**
   * **Threat**: Malicious actors impersonating legitimate users.
   * **Mitigation**: Firebase Authentication requires users to verify their email address during registration.

## Data back up and recovery:

1. **Automated Backups**
   * Firebase Firestore provides automated backups to prevent data loss in the event of system failures.
   * Backup data is stored securely in Google Cloud Storage.
2. **Data Recovery Plan**
   * In the rare event of a data breach or loss, Firebase’s backup and restore functionality ensures data can be quickly recovered without significant downtime.

## Error and security logging:

1. **Error Logging**
   * The app logs critical errors, such as failed API requests or database operations, to Firebase Crashlytics.
   * Logs help developers diagnose and resolve issues efficiently.
2. **Security Event Logging**
   * Firebase tracks security events, such as unauthorized access attempts, allowing developers to monitor and respond to potential threats proactively.

## DEVOPS

The ICC App employs GitHub Actions as the core tool for Continuous Integration and Continuous Deployment (CI/CD). This automation ensures rapid, consistent, and error-free software development processes.

## Pipeline overview:

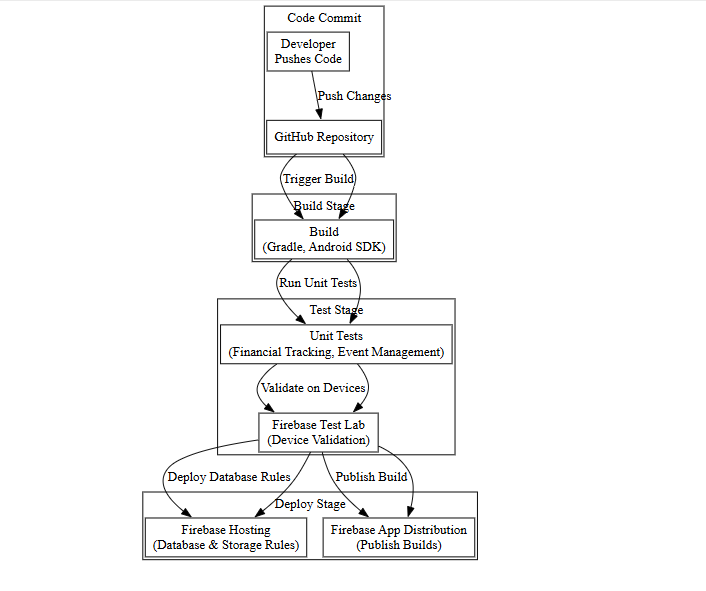
The CI/CD pipeline integrates key stages:

* + 1. Code Commit: Developers push changes to the GitHub repository.
    2. Build: The pipeline compiles the code and ensures it is free of syntax errors.
    3. Automated Testing: Unit and integration tests are run to validate the code.
    4. Deployment: Successfully tested code is deployed to Firebase Hosting.

## Pipeline steps:

* + **Build Stage**:
    1. Uses Gradle to build the Android application.
    2. Ensures compatibility with Android SDK and target devices.
  + **Test Stage**:
    1. Executes unit tests for all major features (e.g., financial contribution tracking, event management).
    2. Firebase Test Lab is used to validate app behavior on multiple Android devices.
  + **Deploy Stage**:
    1. Deploys updates to Firebase Hosting for database and storage rules.
    2. Publishes the app build to Firebase App Distribution for user testing.

## Flow diagram:



## Automation strategy:

To ensure quality and efficiency, the following automation strategies were implemented:

1. **Static Code Analysis**
   * Tools such as SonarQube are integrated into the pipeline to analyse code for vulnerabilities, bugs, and adherence to coding standards.
   * Reports generated highlight areas of improvement for developers.
2. **Testing Automation**
   * Unit Testing: Ensures each component functions as expected. For example, tests validate that financial contributions are correctly saved to Firebase.
   * Integration Testing: Verifies that components interact correctly. For example, testing the interaction between Firebase Authentication and Firestore to enforce role-based access.
   * Device Testing: Firebase Test Lab simulates app behaviour on a variety of Android devices, ensuring consistent performance across low-end and high-end phones.
3. **Deployment Automation**
   * Firebase CLI is used to automate the deployment of database rules, hosting configurations, and app builds.

## Security testing in pipeline:

1. **Firebase Security Rules Testing**
   * Before deployment, Firebase Security Rules are tested to ensure they enforce role-based access control effectively.
   * Example: Leadership-only features are validated to prevent access by regular members.
2. **Encryption Validation**
   * Automated tests validate that all data in transit is encrypted using TLS 1.2.
3. **Penetration Testing**
   * Simulated attacks are run on the app to identify vulnerabilities, ensuring robust protection against unauthorized access or data breaches.

## RUNNING COSTS

The ICC App was designed to minimize operational costs while ensuring scalability and high performance. Below is a detailed breakdown of the running costs, projected in South African Rands (ZAR), based on Firebase’s pricing and the church’s growth expectations.

## Predicted monthly costs:

The ICC App uses Firebase, which offers a free tier for small-scale applications. Costs will scale as user activity increases, and the app transitions to Firebase’s pay-as-you-go plan.

1. **Firebase Free Tier (Initial Stage)**
   * During the initial phase with 200 active users, the ICC App will operate within Firebase's free-tier limits:
     + **Firestore**: 50,000 reads, 50,000 writes, and 1 GB of storage per month.
     + **Authentication**: Email/password authentication is free for up to 10,000 monthly users.
     + **Hosting**: 1 GB of storage and 10 GB/month bandwidth for static content is free.

**Predicted Monthly Cost (Year 1)**:

* + Firebase's free tier covers the expected usage for 200 users, resulting in R0/month during Year 1.

1. **Projected Growth (Year 2)**
   * By Year 2, as the app scales to 500–1,000 active users, usage will exceed free-tier limits, triggering the pay-as-you-go (Blaze) plan:
   * **Firestore**:
     + - Reads: R1.14 per 100,000 reads.
       - Writes: R3.42 per 100,000 writes.
       - Storage: R0.50/GB/month.
   * Hosting: R0.50/GB for bandwidth beyond 10 GB/month.

**Example Costs for 1,000 Users (Year 2):**

* + Reads: 100,000 reads/month → R1.14.
  + Writes: 50,000 writes/month → R1.71.
  + Storage: 5 GB → R2.50.
  + Bandwidth: 5 GB beyond free tier → R2.50.

**Total Estimated Monthly Cost (Year 2)**:

* + Approximately R10–R50/month, depending on user activity and data usage.

## Technology scaling and replacement plan:

1. **Scalability Through Firebase**
   * Firebase’s serverless architecture automatically adjusts to higher traffic and increased user activity without requiring manual intervention.
   * During high-usage events like Good Friday, Firebase dynamically allocates resources to handle increased demand.
2. **Cost Optimization Strategies**
   * **Offline Caching**: Firebase offline persistence minimizes Firestore reads and writes, reducing costs.
   * **Efficient Queries**: The NoSQL database structure ensures that only necessary data is retrieved, avoiding unnecessary charges.
   * **Monitoring Usage**: Firebase’s usage dashboards enable proactive management of resources and costs.
3. **Long-Term Cost Projections (Years 2+)**
   * By Year 3, as the church grows to support 2,000+ active users, monthly costs may rise to approximately R200–R400/month.
   * Depending on future needs, the church may explore alternative solutions or advanced Firebase subscriptions to manage higher usage.

## CHANGE MANAGEMENT

Change management ensures the successful adoption and maintenance of the ICC App within the Independent Church of Christ (ICC). This section details strategies to facilitate organizational and user-level transitions, address challenges, and ensure long-term sustainability.

## Organisational Adoption strategy:

Effective change management starts with preparing the church’s leadership and administrative teams to embrace and promote the ICC App.

1. **Leadership Engagement**
   * The app’s development was guided by input from church leadership to align with organizational goals and address operational challenges.
   * Leadership training sessions were conducted to demonstrate app features, including membership management, financial tracking, and event planning.
2. **Stakeholder Communication**
   * **Purpose**: Keep stakeholders informed about the app’s development and deployment phases.
   * **Approach**: Regular progress updates via meetings and reports ensured transparency and encouraged feedback.
3. **Training and Capacity Building**
   * Comprehensive training workshops were conducted for leadership and key administrative staff to ensure they can:
     + Manage the app’s backend via Firebase tools.
     + Handle role-based permissions and user requests effectively.
   * Training covered scenarios such as creating announcements, verifying contributions, and generating membership reports.
4. **Pilot Program**
   * A pilot version of the app was deployed to a small group of leadership members and active congregation members.
   * Feedback from the pilot was used to refine features and address usability issues before the full rollout.

## User adoption strategy:

Ensuring that church members adopt the ICC App is critical for its success. Strategies were designed to encourage widespread use and address user concerns.

1. **User Education**
   * **Workshops and Demonstrations**: Regular on-site sessions were held after church services to educate members on using the app.
   * **Printed Guides**: Step-by-step user manuals with visuals were distributed to guide members through account creation, Bible access, and financial tracking.
   * **Video Tutorials**: Short, engaging tutorials were shared via church WhatsApp groups and social media platforms.
2. **Ease of Use**
   * The app’s intuitive design ensures accessibility for users with varying levels of technical expertise.
   * Key features like financial tracking, Bible WebView, and sermon notes are presented in a straightforward layout, minimizing the learning curve.
3. **Incentives for Early Adoption**
   * Early adopters were offered special recognition, such as acknowledgment during church services or discounted fees for specific events, to encourage engagement.
4. **Feedback Mechanisms**
   * A dedicated feedback feature within the app allows users to report issues, suggest improvements, or ask for help.
   * Leadership members are tasked with reviewing feedback and implementing necessary changes to improve the user experience.

## Maintainance and support strategy:

A robust maintenance and support framework ensures the ICC App continues to meet user needs post-deployment.

1. **Bug Fixes and Feature Updates**
   * Weekly maintenance cycles are scheduled to address any reported bugs or crashes.
   * Feature updates are planned quarterly to introduce new functionalities or refine existing ones based on user feedback and organizational requirements.
2. **Technical Support**
   * A dedicated support team, accessible via email and WhatsApp, addresses user queries and technical issues promptly.
   * Leadership members trained during the organizational adoption phase also serve as first-line support for app-related concerns.
3. **Monitoring and Analytics**
   * Firebase’s analytics tools monitor app usage, crash reports, and user engagement trends.
   * Insights gained from these metrics help the development team prioritize updates and improvements.
4. **Long-Term Sustainability**
   * Training materials and documentation are maintained and updated regularly to onboard new leadership members or technical staff.
   * A financial reserve is allocated to cover hosting and scaling costs as the app grows.

## Risk management:

Anticipating and addressing risks is a vital part of change management.

1. **Resistance to Change**
   * **Risk**: Some members may prefer manual processes over digital solutions.
   * **Mitigation**: Extensive user education, demonstrations, and incentives are implemented to show the app’s benefits and ease of use.
2. **Technical Challenges**
   * **Risk**: Users with older Android devices or limited internet access may face issues.
   * **Mitigation**: The app is optimized for low-end devices, and offline mode ensures core features remain accessible without an internet connection.
3. **Data Security Concerns**
   * **Risk**: Users may worry about their personal and financial data being mishandled.
   * **Mitigation**: Communication about the app’s compliance with POPIA and robust security measures (e.g., Firebase Authentication, encryption) reassures users.

## APPENDICES

## Sprint Minutes:

**Sprint 1: Concept and Team Formation**

**Date:** 9 September 2024  
**Duration:** 1 hour  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Define collaborative agreements (Work Agreement, Definition of Ready, and Definition of Done).
2. Assign roles and responsibilities within the team.

**Key Discussions:**

* Finalized the Work Agreement to ensure clear communication and support among team members.
* Established the DoR and DoD to streamline task management and completion.
* Assigned roles: Karabo (Scrum Master), Sibonelo (Backend Developer), Boitshepo (Frontend Designer).

**Action Items:**

* Karabo: Document the Work Agreement.
* Sibonelo: Outline DoR criteria.
* Boitshepo: Finalize DoD.

**Sprint 2: Requirement Gathering**

**Date:** 23 September 2024  
**Duration:** 1 hour  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Collaborate with the client to define detailed requirements.
2. Draft user stories and sketch the initial user journey.

**Key Discussions:**

* Met with the client to identify requirements for features like sermon notes, Bible integration, and financial contributions.
* Created preliminary user stories for core features.
* Discussed potential technologies such as Firebase and external APIs for Bible integration.

**Action Items:**

* Karabo: Compile finalized user stories.
* Sibonelo: Research Firebase capabilities for authentication.
* Boitshepo: Draft the user journey diagram.

**Sprint 3: Analysis and Design**

**Date:** 7 October 2024  
**Duration:** 1.5 hours  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Conduct domain analysis and finalize technical architecture.
2. Research SDKs and APIs for Bible integration.
3. Design mock-ups and gather client feedback.

**Key Discussions:**

* Identified essential workflows for the app through domain analysis.
* Researched and shortlisted SDKs and APIs for Bible integration.
* Created initial app mock-ups for client approval.

**Action Items:**

* Karabo: Develop a high-level technical architecture.
* Sibonelo: Prepare a comparison of SDKs for Bible API integration.
* Boitshepo: Update mock-ups based on client feedback.

**Sprint 4: Core Feature Development**

**Date:** 21 October 2024  
**Duration:** 2 hours  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Develop and implement user account and authentication system.
2. Implement Bible access and sermon notes functionality.

**Key Discussions:**

* Successfully developed user account creation and Firebase authentication.
* Bible feature implemented using WebView for seamless integration.
* Completed sermon notes functionality with public and private visibility.

**Action Items:**

* Karabo: Test all implemented features.
* Sibonelo: Optimize authentication and database interactions.
* Boitshepo: Polish the UI for sermon notes and Bible features.

**Sprint 5: Feature Completion and Integration**

**Date:** 4 November 2024  
**Duration:** 2 hours  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Complete financial contributions tracking.
2. Integrate notification systems for announcements.

**Key Discussions:**

* Successfully implemented financial contributions system with verification.
* Notifications integrated for announcements and contributions.
* Initial integration testing identified minor bugs, which were resolved.

**Action Items:**

* Karabo: Oversee integration testing.
* Sibonelo: Fix any additional integration issues.
* Boitshepo: Prepare UI for notifications.

**Sprint 6: Final Review and Presentation**

**Date:** 11 November 2024   
**Duration:** 1.5 hours  
**Attendees:** Karabo Dlamini, Sibonelo, Boitshepo  
**Agenda:**

1. Conduct final testing and prepare documentation.
2. Create a comprehensive presentation for the client.

**Key Discussions:**

* Final testing ensured the app’s stability and functionality.
* Prepared user guides and technical documentation for client handover.
* Developed a presentation to highlight app features and client benefits.

**Action Items:**

* Karabo: Finalize documentation and user guide.
* Sibonelo: Ensure all technical components are production-ready.
* Boitshepo: Finalize the presentation deck.

ALL AUTHENTECITY DECLARATIONS ARE INCLUDED IN SELF EVALUATIONS

SPRINT ATTENDANCE RECORD ATTACHED IN SEPARATE DOCUMENT

# References

altexsoft, 2023. *Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices.* [Online]   
Available at: https://www.altexsoft.com/blog/non-functional-requirements/  
[Accessed 15 november 2024].

GeekForGeek, 2024. *Repository Design Pattern.* [Online]   
Available at: https://www.geeksforgeeks.org/repository-design-pattern/  
[Accessed 14 November 2024].

GitLab, 2024. *CI/CD explained.* [Online]   
Available at: https://about.gitlab.com/topics/ci-cd/  
[Accessed 16 November 2024].

Google, 2024. *Firebase Realtime Database.* [Online]   
Available at: https://firebase.google.com/docs/database  
[Accessed 15 November 2024].

Hanna, K. & Biscobing, J., 2024. *What is an entity relationship diagram (ERD).* [Online]   
Available at: https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD#:~:text=An%20entity%20relationship%20diagram%20(ERD)%2C%20also%20known%20as%20an,information%20technology%20(IT)%20system.  
[Accessed 15 November 2024].

POPIA Act Compliance, 2024. *POPIA Implementation actions.* [Online]   
Available at: https://www.popiact-compliance.co.za/popia-information/4-popia-implementation-actions  
[Accessed 15 November 2024].

Valis, R., 2024. *How to Secure Your Mobile App: Best Practices for Developers and Businesses.* [Online]   
Available at: https://dashdevs.com/blog/mobile-app-security-tips/  
[Accessed 20 November 2024].

Wikipedia, 2024. *Model–view–viewmodel.* [Online]   
Available at: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel#:~:text=Model%E2%80%93view%E2%80%93viewmodel%20(MVVM,that%20the%20view%20is%20not  
[Accessed 15 November 2024].