Kennesaw State University

College of Computing and Software Engineering

Department of Computer Science

CS 4850 - Senior Project 02

Professor Sharon Perry

INDY6-Church Locator: Final Report

Project Website - https://joeknee.github.io/

Isai Valencia - isaivalencia07@gmail.com

Johny Sam - johny21sam@gmail.com

Gregorio Mendoza - mendozav.10@outlook.com

Abstract

This project is aimed to create a church locator app that enables users to locate nearby churches and access their live streams, videos, and donation features within the app. Utilizing the Waterfall model approach in the SDLC (Software Development Life Cycle), the team successfully developed the app, providing users with an easy-to-use interface and robust functionality. The experience accumulated from developing in React Native allowed the team to gain knowledge in software development practices. The significance of this app is its potential to benefit small churches that lack the resources to create their own app, but still need to connect their members. Overall, this project highlights the importance of creating technology solutions for communities with limited resources.

Table of Contents

1. Background Information

The Church Locator app was conceived to address the need for a comprehensive application that would enable churches to better engage with their members. The project owner thought of this idea when the church he attended had a lack of an application that could provide its members with live streams, videos, and donation features. To address this gap in the market, the Church Locator project was launched with the goal of creating a mobile app that would enable users to easily access nearby churches' live streams, videos, calendar events, service times, and donation features within the app. Using prior experience with mobile app development, the team used that knowledge as a basis for developing an app that was oriented to the needs of the church members. The target audience for the Church Locator app is broad, but it is especially aimed at the elderly who may have difficulty accessing church services in person.

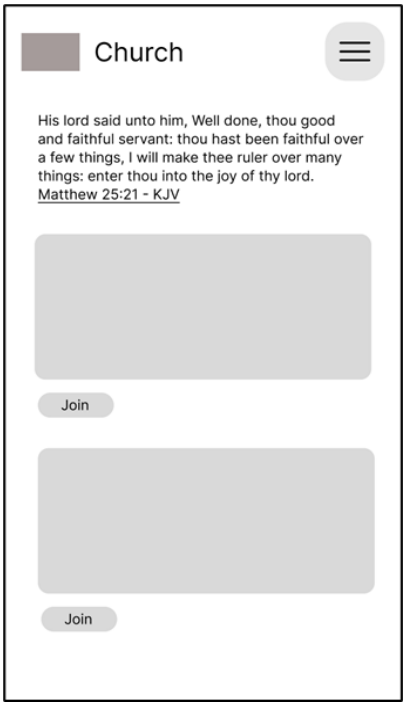
2. Requirements

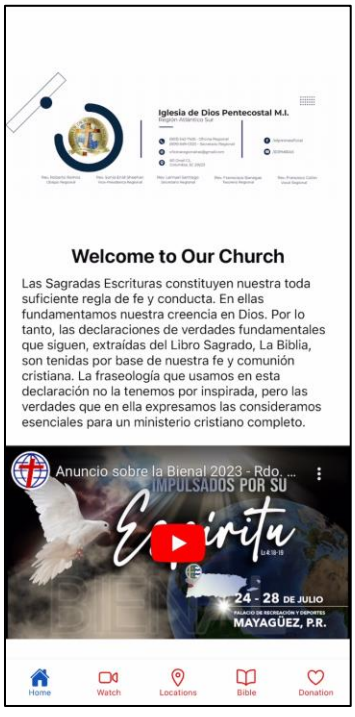
2.1 Functional Requirements

In the requirements section, it is vital to acknowledge that the initial mockup screens created in Figma may not reflect the final product entirely. As the development advanced, there can be alterations or additions to both UI and functionality that were absent from the original designs. Such changes might stem from user feedback, technical constraints, or various other factors arising during the development phase.

These were mockups designed through Figma as compared to the current version of the app:

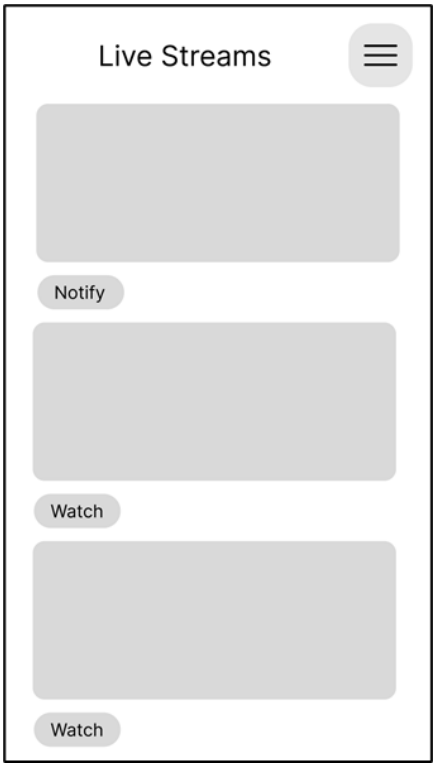
Home Screen

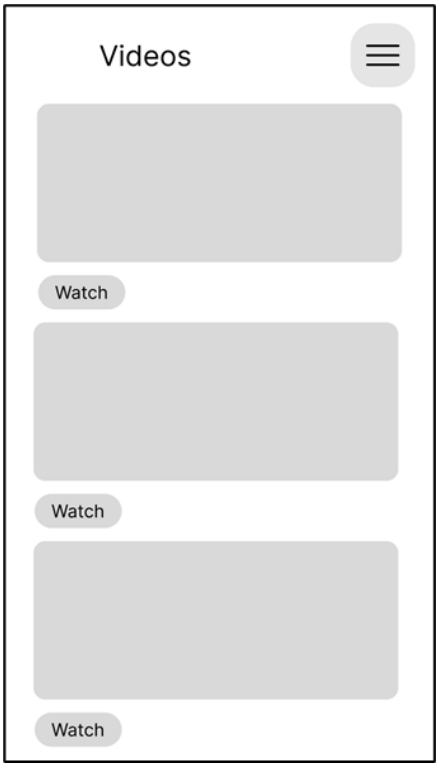




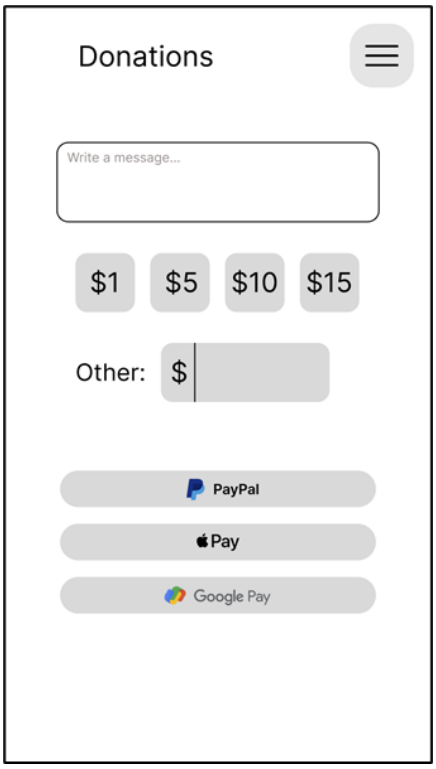
Maps Screen

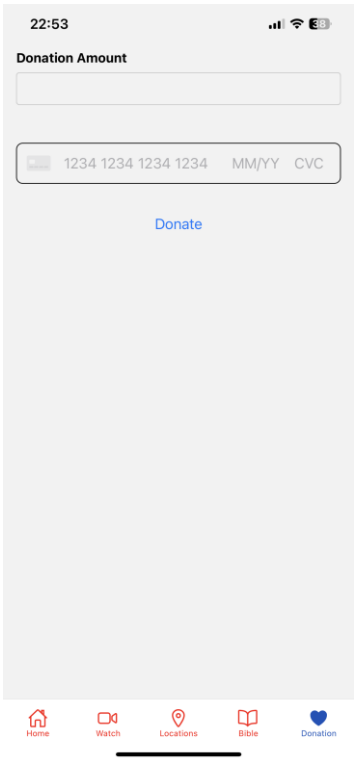
Live Streams and Video Screen - These screens were combined to lessen the complexity





Donations Screen





2.2 Non-Functional Requirements

It is imperative to acknowledge the functional requirements, that aside, there are the non-functional requirements that be adhered to ensure user-friendliness and accessibility. These requirements including usability, user interface, hardware interfaces, and software interface were planned at the start of the project. Some changes had to be made due to technical limitations and prioritization.

Usability

The app should be easy to use and navigate for individuals of all ages. This will require clear and concise instructions, intuitive navigation, and an overall visually appealing UI. Testing and evaluation of the app's usability should be conducted to determine if it provides a positive user experience.

—

To meet the usability, the team decided to change the app's user interface to fit modern user interface/experience standards. Everything is clear and concise to guide users through its various features. The team asked a variety of age groups for their input on the app and developed it accordingly to make it easy to understand.

Hardware Interface Requirements

The hardware requirements for the church locator app are minimal and should satisfy most smartphones, but the app should meet the following specifications:

Processor: ARM or x86 architecture with at least 1.5 GHz processing speed.

RAM: 2 GB or more

Storage: 100 MB of free space

Display: Minimum resolution of 720x1280 pixels.

—

To meet the hardware interface requirements, the team conducted research to determine the appropriate specifications that would provide optimal performance for the app. The team selected a minimum specification to have a smooth user experience. Additional testing was done on various devices to understand how they would run.

Software Interface Requirements

The end user would be required to have a smartphone that is equipped with Android or IOS. Running either Android 7.0 (Nougat) or later, iOS 12.0 or later.

—

To meet the software requirements, the team developed the app to run on Android and iOS operating systems. A minimum operating system (OS) is required across both systems so the team abided by it.

Communication Interface Requirements

The application will require internet connectivity to access the church's information and to process donations.

—

The application requires the device to be connected to the internet to display content or it would not function.

3. Analysis of Tech Platforms

The tech platforms chosen helped the team create a robust and scalable mobile application that provides a seamless user experience

Firstly it was to decide between two widely known mobile development frameworks: React Native and Flutter. Although Flutter had some attractive features such as its ability to produce high performant apps, ultimately the team decided to use React Native. With some research and considerations, it was opted to use React Native as the team had some familiarity with developing side projects with it. Additionally, since the framework primarily uses JavaScript as its programming language, it made it easier to build dynamic interfaces, which was crucial for the functionality of the app.

Next, was selecting an IDE (Integrated Development Environment) that could provide an efficient and effective development environment for React Native. After minimal research, it was decided to use Visual Studio Code because of its lightweight and cross-platform code editor with a range of features and extensions, with Android Studio, made it an excellent choice for developing React Native applications.

When installing React Native and Visual Studio Code, the developer needed a testing environment for development purposes. The team utilized ExpoGo, a free open-source client for testing apps on Android and iOS while providing a comprehensive suite of tools and services that simplified the development process. It only requires the developer to have Node.js and set up a self-hosted server and connect their devices to the installed ExpoGo app. This form of testing provided efficient development while writing the code. Included, the development needed to primarily focus the app to be Android-centric since developing for iOS would not only require a laptop equipped with MacOS but also required a yearly subscription to release and maintain the app to Apple's App Store. So for now developer leveraged the available Android development tools to create the Church Locator app.

For the donation system, the team had to choose between three different payment platforms: Stripe, Paypal, and Square. PayPal is a widely-used payment platform that has been around for many years and is known for its ease of use and reliability. Square is another popular payment platform that is widely used in the retail industry and has a reputation for its ease of use and user-friendly interface. However, the team ultimately chose to use Stripe for the app due to its strong reputation in the payment processing industry and its ability to handle a variety of payment types.

4. Tech Platforms

For the development of the Church Locator app, the team utilized several tech platforms to create an optimized and efficient user experience. One of the key platforms used was React Native, a framework that allows developers to develop an app for mobile devices. Although there was some difficulty concerning configuration issues, they were resolved with watching developer documentation and related tutorial videos.

During the testing phase, the team utilized Node.js, a JavaScript runtime that facilitated the running of JavaScript on the server side, which made it easier to communicate with client-side code written in React Native. With Node.js and the use of ExpoGo, the team was able to efficiently and effectively test the app to make necessary adjustments. ExpoGo allows the app to best tested in real time on mobile devices. To connect the Church Locator app to ExpoGo, a QR code scanner is needed, which enabled the team to run and test the app on both Android and iOS devices.

Visual Studio Code was the go-to IDE, as it provided a lightweight and customizable IDE. It also provided powerful debugging capabilities, Git integration, IntelliSense features, and other extensive plugin libraries. Android Studio was another IDE used for developing the app's Android version. With the help of the Android emulator, it was a good measure to test the app's performance on different devices and identify any issues that arose.

For payment processing, the team integrated Stripe using the developer documentation that was provided on their site. The reliable and secure payment processing platform that allowed the team to integrate multiple payment options. These payment options include credit cards, Apple Pay, and Google Pay.

5. Development of Software Project

During the development of the project, the team loosely followed the waterfall model with combinations of aspects of agile development. It started by defining the requirements, analyzing the system's needs, and designing the software. Then moved on to the implementation stage, where the team worked on coding the software, starting with the user interface and moving on to the back-end functionality. The project owner, Isai, was primarily responsible for the development of coding of the app, while the other two (Johny and Gregorio) focused on documentation and testing but did help out the developer when needed.

The depicted use case diagram illustrates the interactions between the application's end-users and the system's administrative users.

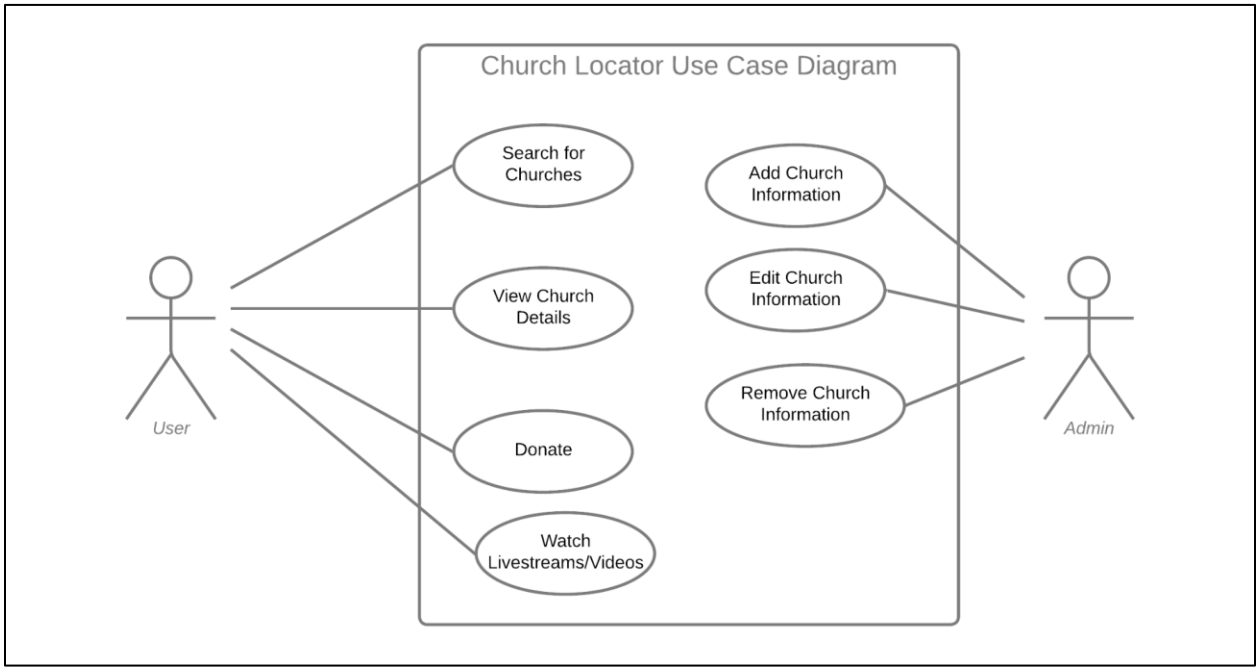


Figure 1 - Use Case Diagram

5.1 Challenges

The team faced many challenges during the development process, such as software issues with the SDK not working with the version of Expo Go that was used. The workaround was by downgrading to an earlier version of ExpoGo that was compatible with the installed SDK, which resolved the issue.

One of the biggest challenges the team encountered was when implementing the donations screen, particularly when trying to add payment functionality. Stripe's API for some reason

would not work with the developed app. To address this, the team ultimately had to follow the documentation again many times to make it functional to integrated into the system.

The Figure 1 diagram is a sequence diagram that provides a visual representation of the flow of actions and communication between the Church Locator app's client, API, and database components.

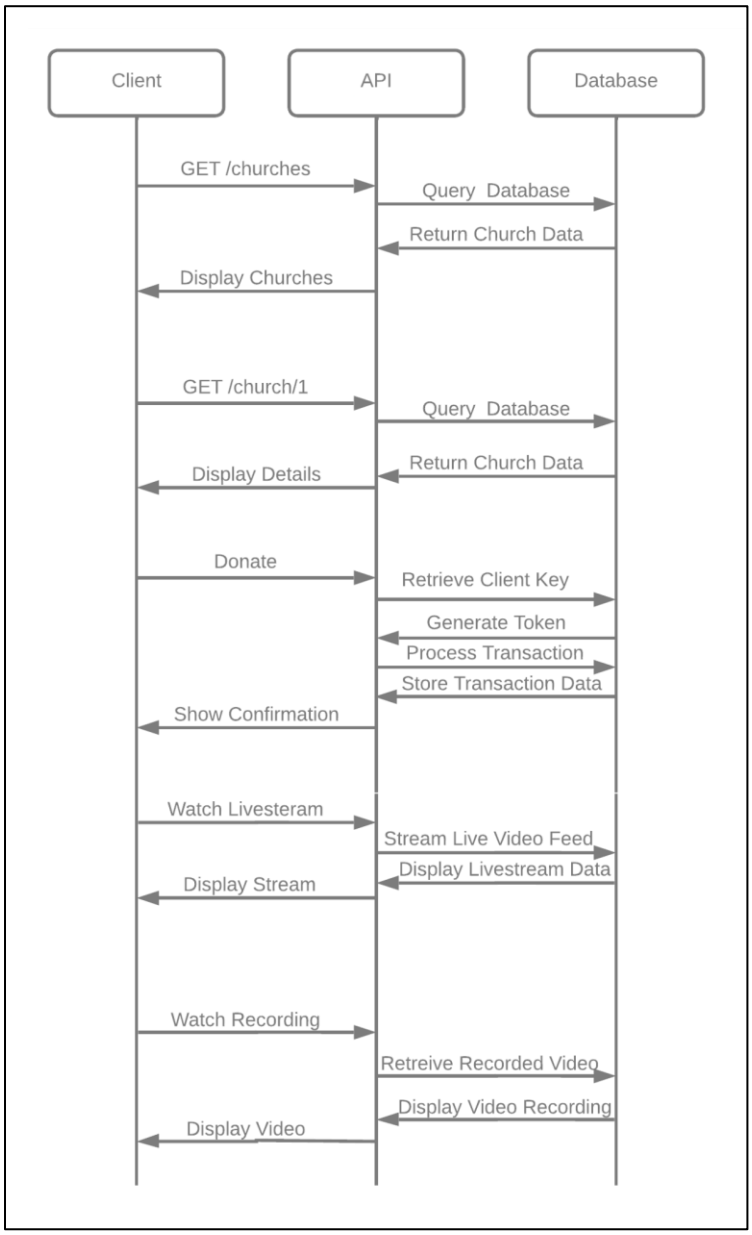


Figure 2 - Sequence Diagram

Another challenge was designing the user interface to accommodate the different screen sizes and resolutions of various devices. The team found that the app looked different on an Android compared to the iPhone due to the varying resolutions, impacting the viewability of the content.

A fix to this is to test on different resolutions provided through Android Studio and optimized for each device type.

Looking back on the development process, the team found that the approach was generally effective. However, it was able to also identify areas where the team could improve in the future. For example, the team could have conducted more extensive user testing to gather more feedback and ensure that the software met the needs of a broader range of users. Additionally, as a team could have put more emphasis on conducting regression testing earlier in the development process to identify and fix issues sooner.

5.2 Future Implementations

As the church locator application develops, one possible upgrade is the integration of Firebase for its database. This enhancement would offer a more dependable and scalable system for storing and retrieving data, thereby increasing efficiency and dependability. In addition, allowing authentication for personal feeds could prove advantageous. Specifically, users can create profiles tailored to their needs with targeted information about churches in which they're interested. To complete this set of enhancements, integrating shared functions into social media platforms will permit individuals who use the app to distribute information amongst their loved ones.

6. Project Planning and Management

To effectively plan and manage a project, it is important to establish clear communication channels to schedule recurring project meetings. For the project, the team communicated through in-person meetings, Discord, and text messages to stay connected for timely communication. As per the Gantt chart in the project plan, the team tried to at least work on the project at least twice a week to spread out all the milestones. The team also scheduled meetings to discuss progress, share ideas, and make any necessary adjustments. Using these communication platforms maintained a consistent meeting schedule.

7. Version Control

Version control management was used with github

8. Summary

To enhance member engagement and convenience, the Church Locator app was developed. React Native served as the framework for creation while Visual Studio Code enabled development with ExpoGo for testing purposes. To ensure streamlined payment processing within the app, Stripe was utilized. Within its interface, users can readily access live streams, videos, calendar events, and service times, and donate as they see fit. It wasn't all smooth sailing during development with issues such as software incompatibilities cropping up. However, both waterfall and agile methodologies were employed leading to instances of improvement being identified such as regression and user testing. The Church Locator app, with its modern approach to reaching and engaging followers through mobile devices, provides an effective solution for churches.