Calvary Chapel Corvallis

Final Report

CS 463 Spring 2017

Kevin Stine, Courtney Bonn, Maxwell Dimm

Group #62

Abstract

The purpose of this project is to produce an iOS/Android application for Calvary Chapel of Corvallis that will allow members to access a plethora of information all in one localized space. The Church's current website does not provide an interface where current members of the church can very quickly access important information such as events, bulletins, and messages from the service. The desired application will be simple enough for anyone to use while providing back end access for staff to easily upload new information to the app. The priorities lie in maximizing the usability of the app and providing bulletin, schedule, video, and giving functionality. We will work with the existing Calvary Chapel web development team to create a product that is seamlessly integrated with their already existing network.

CONTENTS

I Introduction 4

II Original Requirements 4

III Requirement Changes 12

IV Original Design Document 12

IV-A Design Changes . . . 24

V Original Technology Review 24

V-A Technology Changes . . . 47

VI Weekly Blog Posts 47

VI-A Fall 2016 . . . 47

VI-A1 Week 3 . . . 47

VI-A2 Week 4 . . . 48

VI-A3 Week 5 . . . 48

VI-A4 Week 6 . . . 48

VI-A5 Week 7 . . . 49

VI-A6 Week 8 . . . 49

VI-A7 Week 9 . . . 50

VI-A8 Week 10 . . . 50

VI-B Winter 2017 . . . 51

VI-B1 Week 1 . . . 51

VI-B2 Week 2 . . . 51

VI-B3 Week 3 . . . 52

VI-B4 Week 4 . . . 52

VI-B5 Week 5 . . . 53

VI-B6 Week 6 . . . 53

VI-B7 Week 7 . . . 54

VI-B8 Week 8 . . . 54

VI-B9 Week 9 . . . 55

VI-B10 Week 10 . . . 56

VI-C Spring 2017 . . . 56

VI-C1 Week 1 . . . 56

VI-C2 Week 2 . . . 57

VI-C3 Week 3 . . . 58

VI-C4 Week 4 . . . 58

VI-C5 Week 5 . . . 60

VI-C6 Week 6 60
VI-C7 Week 7 61

VII Final Poster 63

VIII Project Documentation 65

IX Learning New Technology 65

X Team Reflection 66
X-A Courtney Bonn 66
X-B Max Dimm 66
X-C Kevin Stine 66

References 68

Appendix A: Essential Code Listings 68

Appendix B: Photos 73

I. Introduction

Our project centered around a local church in Corvallis, Oregon, Calvary Chapel of Corvallis. The church requested our help in creating a mobile application that would work together with their existing website and be used by the members of the congregation. Our main goal was to create an application that was compatible with both iOS and Android smart phones, had a simple design and functionality so all people could use it easily, and incorporated the most important pieces of their current website. It was decided that the main sections the church wanted to see on the applications were the bulletins, events, the ability to donate, and the most recent message video.

The client team was led by project manager, Desiree Gorham. She acted as a channel of communication between us and the senior staff at the church to make sure we were designing the app in the way they had intended. The church was not involved in development, but did offer advice and ideas as to how they wanted the final product to look and function.

The members of our team were Courtney Bonn, Maxwell Dimm, and Kevin Stine. The project requirements were spread among us, though we did change those assignments throughout the process as well as helped each other when needed. The roles were as follows:

- Courtney Bonn: iOS - Bulletin Page; Android - Bulletin, Donation, Events Page
- Max Dimm: iOS - Donation, Messages Page; Android - Messages Page
- Kevin Stine: iOS - Events Page

II. Original Requirements

CONTENTS

1 Introduction 3

1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Definitions, acronyms, and abbreviations . . . 3
1.4 References . . . 3
1.5 Overview . . . 3

2 Overall description 3

2.1 Product perspective . . . 3
2.1.1 System interfaces . . . 4
2.1.2 User interfaces . . . 4
2.1.3 Hardware interfaces . . . 4
2.1.4 Software interfaces . . . 4
2.1.5 Communications interfaces . . . 4
2.1.6 Memory . . . 5
2.1.7 Operations . . . 5
2.2 Product functions . . . 5
2.3 User characteristics . . . 5
2.4 Constraints . . . 6
2.5 Assumptions and dependencies . . . 6
2.6 Apportioning of requirements . . . 6

3 Specific requirements 6

3.1 External Interfaces . . . 6
3.2 Functions . . . 6
3.3 Performance requirements . . . 7
3.4 Design constraints . . . 7
3.5 Software system attributes . . . 7
3.5.1 Reliability . . . 7
3.5.2 Availability . . . 7
3.5.3 Security . . . 7
3.5.4 Maintainability . . . 7
3.5.5 Portability . . . 7

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to list, in detail, all of the requirements intended for our project. Additionally, there will be an overall description of the project, explaining the different aspects of the application in which we are building. This document is intended for the client team at Calvary Chapel of Corvallis, as well as the professors and teaching assistants for our class.

1.2 Scope

At the completion of this project, the software product to be produced will be called Calvary Chapel Corvallis. This application will provide a portion of the information already available on the church website in a mobile-friendly application. The application will not be an exact replica of the website, but rather a combination of certain sections that are used heavily by current members of the church. The main goal of the application is to offer a space for the active members of the church to stay current with events, messages, and the e-bulletin. It will benefit the church members in a different way than their current website, because the application will allow them to have all of the necessary information saved on their smartphone, rather than opening a new browser each time they need to access the website. In order to ensure this application is available to all members, the application will be offered for both iOS and Android users.

1.3 Definitions, acronyms, and abbreviations

1) **Android:** A mobile operating system developed by Google, based on the Linux Kernel and designed primarily for touchscreen mobile devices.
2) **iOS:** A mobile operating system created and developed by Apple Inc. exclusively for Apple's hardware.
3) **App:** A software application designed to run on mobile devices such as smartphones or tablet computers.
4) **App Store:** Apple's digital distribution platform for mobile software applications.
5) **Google Play Store:** Google's digital distribution platform for mobile software applications.

1.4 References

1) **iOS Design Guidelines:** https://developers.apple.com/design/
2) **iOS Human Interface Guidelines** https://developers.apple.com/ios/human-interface-guidelines/
3) **Android Design Guidelines:** https://developer.android.com/design/index.html

1.5 Overview

The rest of this document contains information on what features are expected to be in the app and descriptions of those features.

2 OVERALL DESCRIPTION

2.1 Product perspective

The Calvary Chapel of Corvallis App is a mobile application that will be optimal for both iOS and Android users. This application will incorporate information that is present on the desktop website that is currently available. The information that will be shared between the mobile application and the desktop website includes calendar events, messages, and the e-bulletin.

2.1.1 System interfaces

Each user will be able to use this application on iOS and Android smartphones. To use the application, the user will have to download the application directly to their smartphone from the App Store and/or the Google Play Store. Once downloaded to the user's phone, there will be no additional software needed to use the application.

2.1.2 User interfaces

The interface should be very simple and easy to understand. There should not be more than five to six clickable options per page. Also the users should be able to find any available information on the app within three clicks or taps. The menu will either be a standard navigation bar at the top, or a clickable icon menu whichever proves more usable after testing.

The Calvary Chapel of Corvallis App will adhere to the design guidelines specified by both Apple and Google. The iOS and Android version of the app will be built with adaptability in mind. As iPhones and Android phones have various screen sizes, our app will scale appropriately on all screen sizes.

The app will have the various categories which the user can interact with:

1) **Announcements:** The user should be able to view all the recent announcements.
2) **Events:** The user should be able to view a calendar or list of events that are upcoming as well as past events.
3) **Donations:** The user should be able to donate to the church quickly and securely.
4) **Sermons:** The user should be able to view and/or listen to the most recent sermons.

The iOS version of the app will follow these three primary themes:

1) **Clarity:** Throughout the system, text is legible at every size, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design. Negative space, color, fonts, graphics, and interface elements subtly highlight important content and convey interactivity.
2) **Deference:** Fluid motion and a crisp, beautiful interface help people understand and interact with content while never competing with it. Content typically fills the entire screen, while translucency and blurring often hint at more. Minimal use of bezels, gradients, and drop shadows keep the interface light and airy, while ensuring that content is paramount.
3) **Depth:** Distinct visual layers and realistic motion convey hierarchy, impart vitality, and facilitate understanding. Touch and discoverability heighten delight and enable access to functionality and additional content without losing context. Transitions provide a sense of depth as you navigate through content.

2.1.3 Hardware interfaces

Apple iPhones and Android smartphones will be the devices that are supported by the application.

2.1.4 Software interfaces

We will be incorporating the church's current system for keeping track of their calendar into the new application. The software used by the church is called Church Community Builder. We will use the software's API to connect the existing calendar to the mobile application, which will prevent the church staff from having to update two different calendars.

2.1.5 Communications interfaces

Within the app there will be a page that allows users to connect to the church's existing social media. There should also be the ability to send notices or messages to the church's administration.

2.1.6 Memory

The app should not take up more than 500 Mb of storage. If time allows as well we aim to give users the ability to take down notes within the app.

2.1.7 Operations

The app should be able to display pdf or text messages of their bulletin and schedule. It should also have audio or possibly video support for their past sermons. It should also take input from the user when taking notes or in the giving portions of the app.

2.2 Product functions

The product or app we are making is intended to be a next step for the members to connect with the church after visiting their website. The website is meant as an introduction to the church, the app is the connection that will be made after members have connected. The app itself will have a couple functions but the general idea is that when people who call Calvary Corvallis their home want to access information about the church, they should go to this app to find it. The apps four primary functions will be an announcements page, the church schedule or calendar, a place to watch/listen to the sermons, and the ability to give to the church.

2.3 User characteristics

The intended user is the average, every day person who attends Calvary Corvallis Church. The application can be used by some with little to extensive education. It will be simple enough that people who are not as experienced with smartphones will still be able to use and understand the application. Technical expertise will only be required for the team of people who will be monitoring and updating the application.

We will have two phases of testing for this project. The first phase will take place during the initial design of the app. This phase will be used to test the design before we begin coding. We will then use the feedback from test users to edit the design.

The second phase of testing will take place after the alpha level release of the app. During this phase, we will pick a minimum of ten different people who have varying backgrounds in technical experiences. These people will be asked to test the application, by performing a number of tasks that are designed to test the simplicity and functionality of the app. These tasks will include:

1) Download the application
2) Find and view the calendar
3) Find and view the announcements
4) Listen to a recent sermon
5) Find the donation page and successfully donate to the church
6) Find and read the recent bulletin
7) Find and visit the church's social networking sites

If 90% of test users are able to complete all the assigned tasks, then we will consider the application successful. We will ask for feedback from the users on the simplicity of finding each section of the app and how many clicks it took them to reach a destination. Additionally, we will ask for feedback on the basic functionality of the app and whether or not they felt it was easy to navigate and worked as intended.

If more than 10% of the test users fail the assigned tasks, then we will need to review the design of the application. We will use the feedback from the users and incorporate changes accordingly. After we handle all the necessary changes, we will perform this phase of testing again.

2.4 Constraints

The client has requested that the app is simple and does not have an overly complicated color scheme. The app needs to also be secure when handling peoples donations, so their information will need to be encrypted. The app should also run smoothly on all modern smartphones made in the last 3 to 4 years. Also the app should run quickly enough that any information that can be found on the app should be accessible quicker than navigating to their website.

2.5 Assumptions and dependencies

This project assumes that the application will be accepted by Apple into the official app store. If it is not accepted, then there will be additional work needed to ensure the app follows the official Apple guidelines.

2.6 Apportioning of requirements

Our first task will be to complete a technology review that goes into depth about the already available technology that can assist us with our project. Next, we will create a design document that will detail the design of our project.

Some of the features of the app that can be further developed over time would be the video support for the sermon. At first we plan to develop to have the audio sermons on the app, but if the church has the video available we can work towards making that available in future versions of the app. There is the possibility to have the ability for members to RSVP or confirm attendance to specific events in future versions of the app as well.

3 Specific requirements

3.1 External Interfaces

The only external interface needed for this app to work properly is a iOS or Android smartphone that has the most up-to-date software. No other external interfaces are required for the functionality of the app.

3.2 Functions

This app should be a relatively simple interface that allows the user to connect with the church's most important functions. The functions that will be included are:

1) Announcements:
   - Add new announcement
   - Edit or delete an announcement
   - View an announcement
   - View previous announcements
2) Calendar:
   - View details for an event
   - View previous months

3) Sermons:
   - Upload new sermon
   - Delete/archive a sermon
   - Listen to a sermon
4) Giving:
   - Give

3.3 Performance requirements

Any number of users should be able to download the application and have it stored on their phone. The app should support all users simultaneously.

3.4 Design constraints

There are design constraints imposed by Apple guidelines for iOS applications. We will also adhere to the Material Design constraints imposed by Google. Additionally, the client would like a simple design that makes it easy for non-technical users to navigate the app with little to no help.

3.5 Software system attributes

3.5.1 Reliability

The app should operate well under all conditions and be considered reliable enough to be used for any amount of time without failing.

3.5.2 Availability

The app should be available at all times, unless there is scheduled maintenance that requires it to be down for a minimal amount of time.

3.5.3 Security

Because users will be using personal information for the donations, the app should be considered secure and not save their personal information.

3.5.4 Maintainability

The app will be simple enough that is is able to be maintained by the existing website development team at Calvary Chapel Corvallis.

3.5.5 Portability

The app should be portable as it should run on two different platforms: iOS and Android.

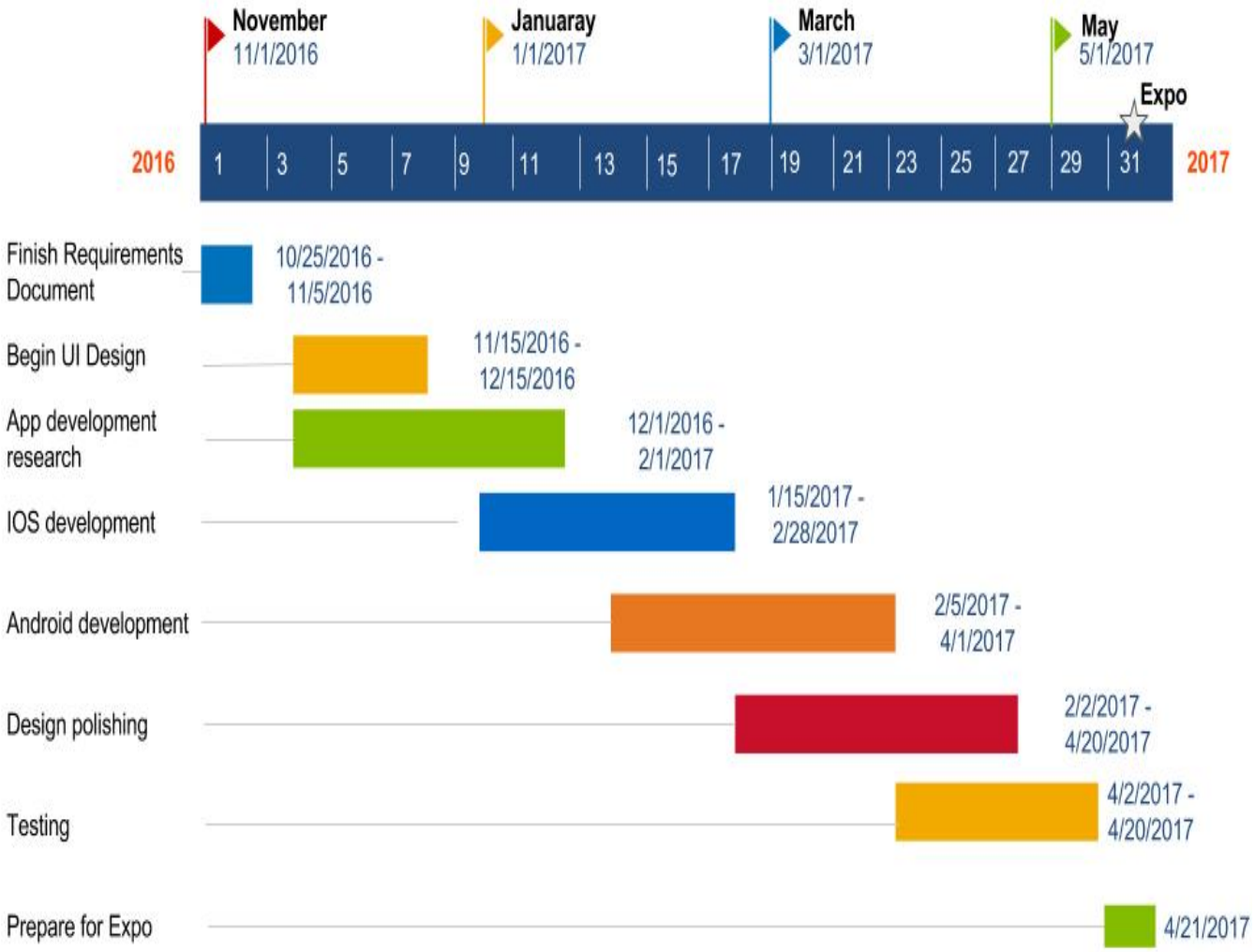


Fig. 1. Project Timeline

III. Requirement Changes

Due to having well established our clients desires early and maintaining heavy communication fall and early winter term, we were able to make little to no changes to our requirements doc. We made sure to establish exactly what our app should do and look like early on and work towards that same goal throughout our development. Our client towards the end of winter term and in spring term requested a few altercations to the design that we were able to complete such as adding additional navigation to the events page. The only requirement we did not add was the addition of push notifications due to how late in the development process it was requested. However, due to the nature of the changes, they still fit within the established requirement document and hence did not need changes.

Upon presenting the finished product to our client we got feedback on the overall feel and design of the app. While it fit all of our originally established requirements there were still a few modifications our client requested. We added a date picker to the events, changed the design of the homepage, and added loading wheels to the pages for added usability. Our client requested that we add support for multiple languages, but due to timing, we were unable to deliver on that request.

IV. Original Design Document

CONTENTS

I Overview 4

I-A Scope . . . 4

I-B Purpose . . . 4

I-C Intended Audience . . . 4

II Conceptual model for software design descriptions 4

II-A Software design descriptions within the life cycle . . . 4

II-A1 Influences on Software Design Document (SDD) preparation . . . 4

II-A2 Influences on software life cycle products . . . 5

II-A3 Design verification and design role in validation . . . 5

III Design description information content 5

III-A Introduction . . . 5

III-B SDD Identification . . . 5

III-C Design stakeholders and their concerns . . . 5

III-D Design views . . . 5

III-E Design viewpoints . . . 5

III-F Design elements . . . 6

III-G Design overlays . . . 6

III-H Design rationale . . . 6

III-I Design languages . . . 6

IV Design Viewpoints 7

IV-A Introduction . . . 7

IV-B Context Viewpoint . . . 7

IV-C Composition Viewpoint . . . 8

IV-D Logical Viewpoint . . . 9

IV-D1 Events Class . . . 9

IV-D2 Messages . . . 9

IV-D3 Donations . . . 9

IV-D4 Bulletins . . . 9

IV-E Dependency Viewpoint . . . 9

IV-F Patterns use viewpoint . . . 9

IV-F1 Model Objects . . . 10

IV-F2 View Objects . . . 10

IV-F3 Controller Objects . . . 10

IV-G Interface viewpoint . . . 10

IV-H Interaction Viewpoint . . . 11

IV-H1 Viewing the Calendar 11
IV-H2 Reading the Bulletin 11
IV-H3 Donating to the Church 11
IV-H4 Watching the Message 11

V Conclusion 12

I. Overview

A. Scope

We will be implementing a mobile app for both iOS and Android platforms. This app will provide the members of Calvary Chapel Corvallis a centralized hub that will allow them to access the most important information about the church. Users will be able to view a calendar, e-bulletins, and sermons. Users will also be able to donate to the church. Three people will be involved in the implementation of this software and it will be done during October 2016 and June 2017.

B. Purpose

The purpose of this Software Design Document (SDD) is to detail how the app software will be designed. We will discuss how we will meet the requirements for our church app and discuss the structure of the app.

C. Intended Audience

The intended audience of this design document will be our clients, the teachers of CS 461, as well as the teacher's assistants.

Glossary

Android A mobile operating system developed by Google, based on the Linux Kernel and designed primarily for touchscreen mobile devices. 1, 4, 6, 8–11

app A software application designed to run on mobile devices such as smartphones or tablet computers. 1, 4–12

iOS A mobile operating system created and developed by Apple Inc. exclusively for Apple's hardware. 1, 4, 6, 8, 10, 11

Model-View-Controller A design pattern that assigns objects in an application one of three roles: model, view, or controller. Also called MVC. 9

Acronyms

CCB Church Community Builder. 8, 9, 11, 12

MVC Model-View-Controller. 9

SDD Software Design Document. 2, 4, 5

SRS Software Requirements Specification. 4, 5

UI User Interface. 10

UML Unified Model Language. 5, 6

XML EXtensible Markup Language. 10

II. Conceptual model for software design descriptions

A. Software design descriptions within the life cycle

1) Influences on SDD preparation: The key influence on this SDD is the Software Requirements Specification (SRS) document that has previously been documented. The requirements listed in the SRS will greatly determine the design of the software and how we implement this project.

2) Influences on software life cycle products: This SDD may lead to necessary changes in the SRS. Throughout development, it's possible that there will be design changes that require us to change details in the SRS. Testing may also be changed based on the SDD.

3) Design verification and design role in validation: In order to determine if the app has met requirements, we will perform user testing. This will involve having users try to use the software in the intended manner and see if they are successful. Success will be determined by how many users are successfully able to use the app without errors or issues.

III. Design description information content

A. Introduction

Within this SDD, there are many required contents. We will identify the SDD and it's stakeholders. We will discuss design concerns and selected design viewpoints. We will also discuss design views, design overlays, and the design rationale.

B. SDD Identification

A valid SDD includes the following parts:

- Date of issue and status
- Scope
- Issuing organization
- Authorship
- References
- Context
- One or more design languages for each design viewpoint used
- Body
- Summary
- Glossary
- Change history

C. Design stakeholders and their concerns

The design stakeholders of the Calvary Chapel of Corvallis app are the developers of the app and the team at the church. Design concerns of the stakeholders include creating an app that is user-friendly and very simplistic in design features. The final app will be designed in a way that will ease this concern, as the intended design will be a very easy-to-use app.

D. Design views

We will use Unified Model Language (UML) diagrams in order to describe and represent some of the views of our system.

E. Design viewpoints

There are many viewpoints that will be covered in this document including: context, composition, logical, dependency, patterns use, interface, and interaction viewpoints. They each detail a slightly different view of the system. For example, the context viewpoint will cover what type of users will be using the app and the perceptions they should have over it. The

composition viewpoint will cover what information and content will be hosted within the app. The logical viewpoint will cover what purpose the app will serve and how it will accomplish those purposes. The dependency viewpoint will be the outside parts the app needs in order for it to work as designed and the integration of it with other apps. Finally, we will conclude with the interaction viewpoint witch will cover how people will use our app and how it will interact with various other technologies.

F. Design elements

Because the purpose of this design is to create a simplistic app that will not require the user to do much searching, the elements we will incorporate into the design are somewhat simplistic as well. When the user opens the app, the first thing he/she will see is a homepage that mimics their current website–a white background with their logo at the top of the page. There will be a banner that is displayed which will frequently change and will be pulled from their website. There will be a menu with four buttons: messages, give, bulletin, and events. The menu will be available on each additional page.

As far as design constraints go, this app is going to be best suitable for current versions of both iOS and Android smartphones. For the iOS app, the screen resolution will be based off current screen sizes, approximately 4.7 inches and 5.5 inches (diagonal), though the app will adjust to smaller and larger screens, allowing users to access the app on older phone models as well as iPads. For the Android app, the screen resolution will adapt to all screen sizes, including tablets.

G. Design overlays

When using the app, we want to create a sense of continuity between the existing infrastructure and the new app we are creating. We will be pulling the seasonal banner from the site purely as a stylistic decision. We will be keeping the color scheme simple, using approximately 3 main colors matching those on their new unlisted website. The transition screen or loading screen for the app will be a simple black with the center being the Calvary Corvallis flame logo. This will also be used for the backdrop for our main pages of the app too but with the logo faded out as to not distract from the info on screen. Additionally we can create an overlay for the schedule and calendar that are simplistic and identify the information to the user such as a title at the top of the page.

H. Design rationale

Two of the primary focuses in our design rationale are to keep it simple for the users, and to keep is simple for the church management team. The client wants the app to be as user friendly for the congregation at possible, along with keeping the back end work to a minimum. If the app is too confusing for the users or the team, then the whole purpose of this app will be voided by their inability to use it. Secondary to these points is maximizing speed and adding additional small features. We should be creating an efficient app that can have helpful functionality for the users.

I. Design languages

The design language that we will use in this document will be UML.

IV. Design Viewpoints

A. Introduction

In this section, we will discuss seven design viewpoints including:

- Context Viewpoint
- Composition Viewpoint
- Logical Viewpoint
- Dependency Viewpoint
- Patterns Use Viewpoint
- Interface Viewpoint
- Interaction Viewpoint

B. Context Viewpoint

Included in the app are four primary functionalities: calendar, sermons, donations, and bulletin. The reason being is that these are the four primary things that our client believes their users will be looking to access from the app. The schedule will be there to allow users to see what is going on within the church in the long-term. The sermon functionality will allow people to view past sermons in case they missed them or simply want to recap a message they enjoyed. In this church and many others receiving donations is a major part of what they do. Making this easier for the users of our app is crucial. Now they can do it from home or any time because the app will allow them to give without the need of cash on hand. Finally the last functionality that we want to include is the bulletin. This will be a place for the users to grab quick info that would normally be handed out on a piece of paper inside the building. This will allow them to view the bulletin from anywhere at any time and reduce costs for the church to prevent the need of printing off pamphlets.

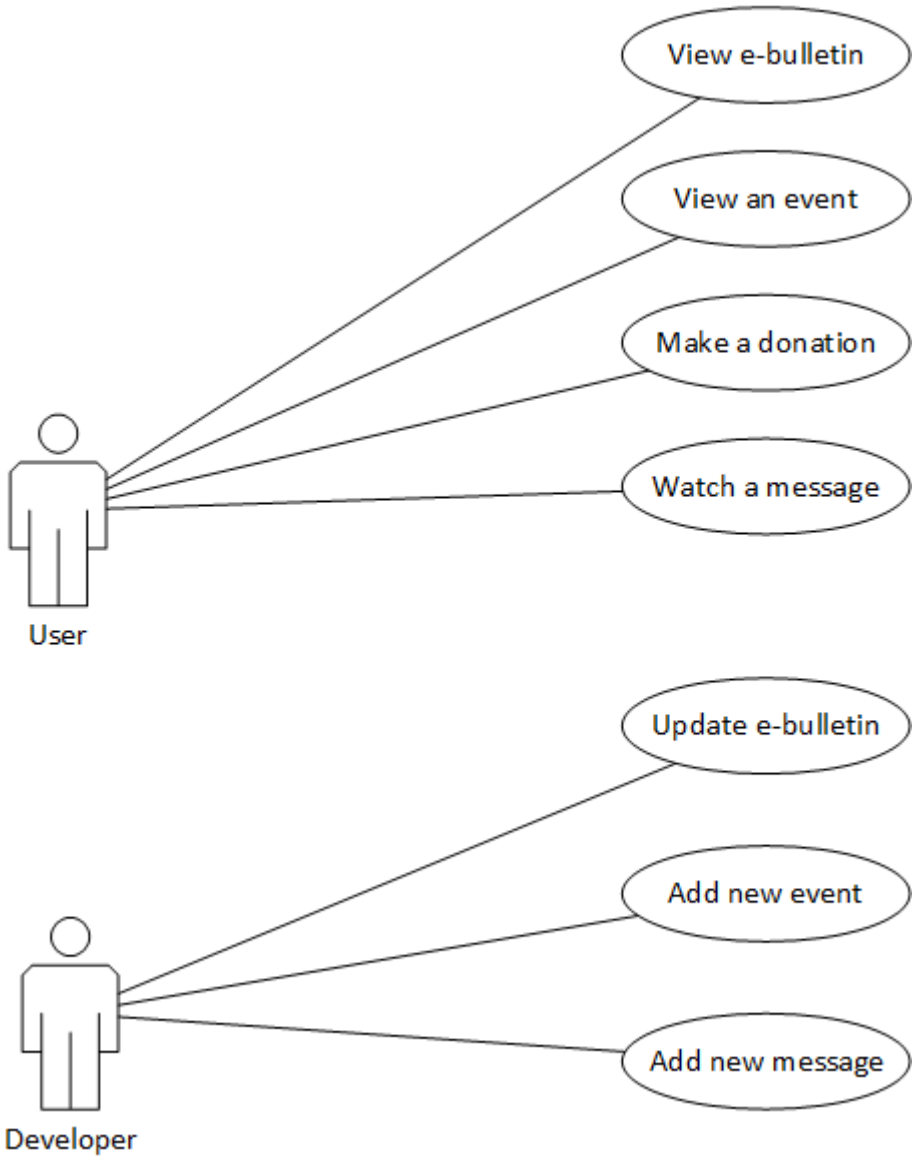


Fig. 1. Use Case Diagram

C. Composition Viewpoint

Our system is composed of four components which are an iOS Client, an Android Client, a Database Server and a Web Server. The Database Server which we connect to is through Church Community Builder (CCB). The Web Server which we connect to is Calvary Chapel's current website. Client components are comprised of smartphone apps, based on their operating system. Our iOS and Android Clients will be able to pull information from both CCB as well as the church's current website.

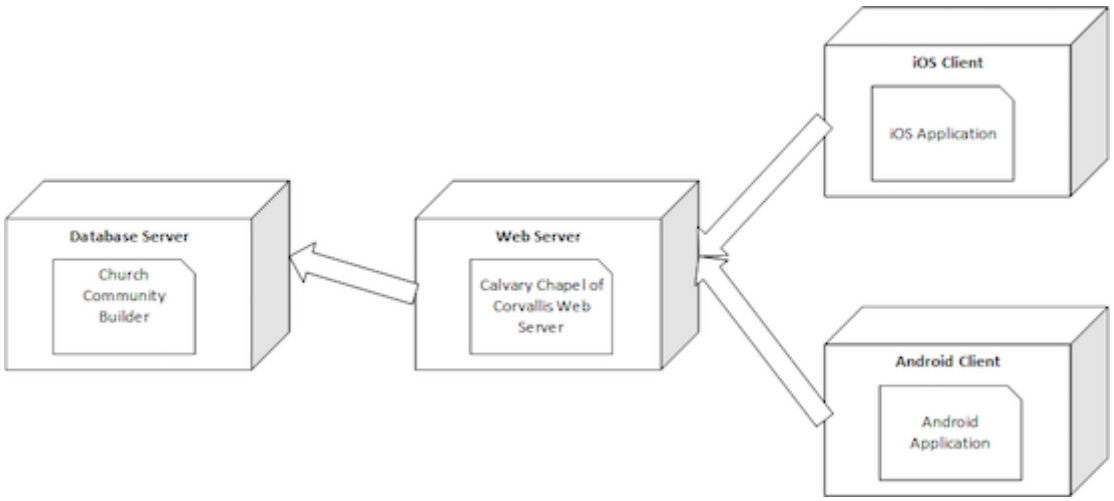


Fig. 2. Deployment Diagram

D. Logical Viewpoint

There will be four main classes which make up the bulk of this app. These classes include: events, messages, donations, and the bulletin. In this section, we will describe the details of what each class will do.

1) Events Class: The events class will handle pulling the calendar from the church's database and displaying it on the app for the user to see. The adding and updating of events will be handled by the staff on the database, CCB. All changes will then be pushed to the app.

2) Messages: The messages class will handle displaying the recent sermon in either video or audio form. The video will be pulled from the church's existing system, LiveStream.

3) Donations: The donations class will handle the interface for users to make a donation to the church. We will connect this page to the church's existing donation system, Foxycart, which will allows the users to make a secure donation.

4) Bulletins: The bulletins class will handle displaying the current bulletin on the app. The bulletin will be pulled from the database, CCB, and displayed on the app. In order to update the bulletin, the staff at Calvary Chapel of Corvallis will update the information on their database, which will push the changes to the app.

E. Dependency Viewpoint

The app will have a couple interconnections with other apps plugins. The first and foremost will be with CCB. Most of the information we will be pulling from continuously will be from CCB, such as the schedule and bulletin. Also we will need to work with LiveStream's API to get a functional video/audio player for the sermons section. Our client already uses foxycart for their donation service so we will integrate their service to our app using their published API. We will be sending information to their servers so we will also need to find a way to encrypt the data being sent because it is highly personal info.

F. Patterns use viewpoint

Our system will make use of the Model-View-Controller Android design pattern. The Model-View-Controller (MVC) design pattern assigns objects in an app with one of three roles: model, view or controller. In addition to defining the

roles the objects play in the app, it also defines the way objects communicate with each other. For example when the user refreshes the sermons page to get the latest sermons, the model object changes and notifies the controller object which will then update the sermons view objects.

1) Model Objects: Model objects encapsulate the data specific to the app and define the logic and computation that manipulate and process the data. In our app, there will be data specific to donations, calendar, e-bulletins and messages. This will strictly be the data that we pull from the CCB database and the Church's website and will not be concerned with user-interface and presentation issues. We will parse the CCB information as EXtensible Markup Language (XML) and read that into our app as XML. The model object will strictly deal with data pulled from the CCB database. Once the data has been retrieved, it will notify the controller object to begin the process of updating the user's view. This object is hidden via an abstraction layer, ensuring that the user is only able to view the information we provide them in a clean and concise view.

2) View Objects: The view object is an object in an app that the users can see. Our app will utilize the view objects to display data from our model object which is pulled from CCB. This object is what the user will see and interact with through our app. View objects will be consistent across pages to ensure the most navigable User Interface (UI) for the user. An example of the view object would be when the user wants to view the calendar for events. If they are in the month of December and want to look ahead to January, they will tap to proceed to the next month, which will notify the controller object letting the model object know to pull January's data from the database.

3) Controller Objects: The controller object will act as an intermediary between our app views and its model objects. The controller will be a conduit through which view objects learn about changes in model objects and vice versa. In addition to communication between the view and model objects, the controller object will also perform setup, coordinate tasks and manage the life cycle of other objects. When the model object is updated with the month of January, the controller object communicates the new data to the view objects so they can be displayed to the user.

G. Interface viewpoint

Our system will utilize tabbed applications to allow for easy navigation through the pages. For both iOS and Android, we will have persistent buttons at the bottom of the screen giving the user an efficient and discoverable navigation pane. This allows for the most discoverability, as the user can interact with each page directly from the home page by clicking any of the icons at the bottom of the screen. Rather than having the user swipe in from the side to access a side bar menu, we want to give the users a consistent view across platforms that it easily recognizable. By tapping on a bottom navigation icon, the user will be directly taken to that associated view, or have the currently active page refreshed.

Our application will make use of the following five navigable options:

- **Home:** The main page that users will see when they open the application. Should show some basic data pulled from either CCB or the website such as the logo, banner, and potentially any updates/news.
- **Events:** The events page will show the user a calendar view that will get pulled down from the CCB database.
- **Bulletin:** The page to allow the user to view updates and announcements in the form of a bulletin. Will pull the bulletin from the CCB database and pushed to the view object.
- **Donate:** The page to allow users to donate to the church. We will link in directly to the Church's donation vendor (Foxycart) and utilize their API to make calls when users want to donate. Our fallback method will be to utilize an

in-app browser that will bring the user to the donation page.

- **Sermons:** This page will bring up the past sermons from the Church's Database, either through CCB or a third party site such as livestream or YouTube.. This is still under discussion, however if the sermons are uploaded to CCB, we can just pull those in from the database and display them using the respective built-in video players. If the Church decides to use a different platform such as YouTube

A tabbed application is a built-in layout for Xcode and will be utilized through Apple's Interface Builder. Through the Interface Builder we will make use of the following storyboard controllers:

- Table View Controller
- Navigation Controller
- Tab Bar Controller

We will add relationships between the initial storyboard view and the other tab pages, to allow for smooth transitions between various pages.

H. Interaction Viewpoint

The Interaction viewpoint describes the high-level functionality of each part of the app. There are four different functionalities that will be addressed: viewing the calendar, reading the bulletin, donating to the church, and watching the message.

1) Viewing the Calendar:

1) **Open App:** The interaction begins with the user open the iOS or Android app.
2) **Click Calendar:** After opening the app, the user will see the menu and choose the option to view the calendar.
3) **Pull Calendar:** Once the user has clicked on the calendar, the app will fetch the calendar from the database, CCB.
4) **Return and Display Calendar:** The database will then display the calendar on the app.

2) Reading the Bulletin:

1) **Open App:** The interaction begins with the user open the iOS or Android app.
2) **Click Bulletin:** After opening the app, the user will see the menu and choose the option to open the bulletin.
3) **Pull Bulletin:** Once the user has clicked on the bulletin, the app will fetch the e-bulletin from the database, CCB.
4) **Return and Display Bulletin:** The database will then display the e-bulletin on the app.

3) Donating to the Church:

1) **Open App:** The interaction begins with the user open the iOS or Android app.
2) **Click Bulletin:** After opening the app, the user will see the menu and choose the option to donate.
3) **Load Donation Page:** Once the user has clicked on the donate button, a page will load that will connect the user to Calvary Chapel of Corvallis' webpage that will display the donation page. The user will then follow the instructions on the page to continue making a donation to the church.

4) Watching the Message:

1) **Open App:** The interaction begins with the user open the iOS or Android app.
2) **Click Messages:** After opening the app, the user will see the menu and choose the option to open the messages.
3) **Pull Messages:** Once the user has clicked on the messages, the app will fetch the message from LiveStream.
4) **Return and Display Message:** LiveStream will then display the recent video message.

V. Conclusion

The app we are building is a very user-friendly, simple app that will allow the members of the Calvary Chapel of Corvallis to access the most important pieces of their main website. This design document describes how our system for the app is going to work. It details the different parts of the system and how they will interact with each other. Additionally it describes how each part will work with the church's database, Church Community Builder (CCB). The design document provides a design framework for how the app is going to be implemented and will continue to change depending on the implementation of the project.

A. Design Changes

V. Original Technology Review

CONTENTS

I Introduction 5

II iOS Development Platform 5

II-A Xcode using Swift . . . 5
II-B AppCode . . . 5
II-C iPhone App Builder . . . 5
II-D Goals . . . 5
II-E Criteria Evaluation . . . 6
II-F Table Comparison . . . 6
II-G Discussion . . . 6
II-H Selection . . . 6

III iOS User Interface Organization 7

III-A Sketch . . . 7
III-B Interface Builder . . . 7
III-C Marvel App . . . 7
III-D Goals . . . 7
III-E Criteria Evaluation . . . 7
III-F Table Comparison . . . 8
III-G Discussion . . . 8
III-H Selection . . . 8

IV Android Development Platform 8

IV-A Android Studio . . . 8
IV-B Appcelerator . . . 9
IV-C Adobe PhoneGap . . . 9
IV-D Goals . . . 9
IV-E Criteria Evaluation . . . 9
IV-F Table Comparison . . . 10
IV-G Discussion . . . 10
IV-H Selection . . . 10

V Android User Interface Organization 11

V-A Tab Navigation . . . 11
V-B Side Bar Navigation . . . 11
V-C Bottom Navigation . . . 11
V-D Goals . . . 11
V-E Criteria Evaluation . . . 11

V-F Table Comparison . . . 12
V-G Discussion . . . 12
V-H Selection . . . 12

VI Cross-Platform Development 12
VI-A Sencha . . . 12
VI-B Appcelerator Titanium . . . 13
VI-C Apache Cordova . . . 13
VI-D Goals . . . 13
VI-E Criteria Evaluation . . . 14
VI-F Table Comparison . . . 14
VI-G Discussion . . . 14
VI-H Selection . . . 14

VII Integrating the Giving Platform 14
VII-A Authorize.net . . . 14
VII-B Pushpay . . . 15
VII-C PayPal . . . 15
VII-D Goals . . . 15
VII-E Criteria Evaluation . . . 15
VII-F Table Comparison . . . 16
VII-G Discussion . . . 16
VII-H Selection . . . 16

VIII Integrating E-Bulletins 16
VIII-A Contentful . . . 16
VIII-B Wordpress . . . 17
VIII-C Church Community Builder . . . 17
VIII-D Goals . . . 17
VIII-E Criteria Evaluation . . . 17
VIII-F Table Comparison . . . 17
VIII-G Discussion . . . 18
VIII-H Selection . . . 18

IX Integrating the Sermon Platform 18
IX-A LiveStream . . . 18
IX-B YouTube . . . 18
IX-C Vimeo . . . 18
IX-D Goals . . . 19
IX-E Criteria Evaluation . . . 19

IX-F Table Comparison 19
IX-G Discussion 19
IX-H Selection 19

X Integrating the Calendar Platform 20
X-A Google Calendar 20
X-B Microsoft Outlook 20
X-C Church Community Builder 20
X-D Goals 20
X-E Criteria Evaluation 20
X-F Table Comparison 20
X-G Discussion 20
X-H Selection 21

XI Conclusion 22

References 23

I. Introduction

This document will break down our application in 9 different parts. There will be an outline on 3 different types of technology that can be used to implement the 9 different parts of the system. We will detail each type of technology as well as discuss why it could be useful for the project. We will then compare each type of technology against each other and analyze which one would be the best for us to use in production. The 9 pieces of the app are divided among the three group members. iOS development platform, iOS user interface organization, and integrating the e-bulletins from the current website will be researched and handled by Courtney Bonn. Android development platform, Android user interface organization, and integrating the current giving platform will be researched and handled by Kevin Stine. Finally, cross-platform development, integrating the recent sermons, and integrating the existing calendar will be researched and handled by Max Dimm.

II. iOS Development Platform

A. Xcode using Swift

The main technology used to develop iOS applications is Xcode. Xcode is a program that can be downloaded on any Mac OS platform and is not able to be downloaded on Windows PCs. It makes use of developer tools to allow users to create an iOS app that will work with all updated iOS devices. This tool uses the language of Swift, which is a new programming language that has been released by Apple [1]. For people who have never developed an app before, Swift is considered the easier language to learn [2]. The tool allows the developer to use a simulator that will display the graphical interface that the user will see and makes the process of developing an app for iOS devices go smoothly.

B. AppCode

AppCode is a smart IDE that is used for iOS/Mac OS development [3]. There are many features for this IDE that allows app developers to easily and quickly develop an application without dealing with any hassle from their IDE. Some of the features include efficient project navigation, smart completion, reliable refactoring, thorough code analysis, and unit testing. Additionally, AppCode not only supports Swift, but Objective-C, C, C++, JavaScript, XML, HTML, CSS, and XPath. This wide variety of languages allows this IDE to be used for more than just iOS development all in one program. AppCode also works directly with Xcode so developing an app for iOS devices is made simple. This IDE is a paid subscription, costing $199 for the first year and the price decreasing after that. There is a free 30-day trial available to determine whether or not the program is the right tool for the wanted product.

C. iPhone App Builder

The third option for a toolkit to build an iOS app is a program that will essentially build the app for you, leaving no need to learn how to code. One program, AppyPie [4], boasts that it will take all of the work out of creating an app and allow an unskilled user to get a working app quickly uploading to the App store. This option would be useful for people who have no prior programming experience and are just looking to get an app produced with little effort.

D. Goals

The goal for comparing iOS development tool is to acquire which way is most acceptable and makes most sense for developing an iOS application. We want to determine which option is relevant to current applications and which option will produce a valid application.

E. Criteria Evaluation

The criteria we will use for determining the iOS Development Platform includes:

1) **Price:** What is the cost associated with the program?
2) **Difficulty:** Will the program do all the work for you? Is there room for learning how to build the app from scratch?

F. Table Comparison

TABLE I
COMPARISON OF IOS DEVELOPMENT PLATFORMS

Tool	Brief description	Rationale
Xcode using Swift	Using the Swift programming language to write the application using the program Xcode	Best option as it is the most recent and will allow us to actually learn how to develop an app
AppCode	Using the AppCode IDE to write an iOS app	Great option if price were not an issue as it does incorporate multiple languages into the IDE
iPhone App Builder	Using a third-party program to automatically build the app	A good option for people who have no programming experience and just want to quickly produce an app

G. Discussion

Using Xcode with Swift is a strong option for building iOS–some would say that it is one of the only options. Using a third-party IDE like AppCode is also a strong option as it does have many other features, like the ability to program in other language, that Xcode does not have. Not only that, it does integrate itself with Xcode. However, at $199 for 1 year, the price is steep and may only make sense for those who develop apps professionally.

The idea of using a third-party program to build the app automatically is tempting, but would defeat the purpose of learning how to develop an iOS app. If the objective of this project was to produce an iOS app within a matter of days or weeks, then this option could be considered useful.

H. Selection

Because our main objective for this project is to learn the detail of producing an iOS app, it is best to choose Xcode with Swift. Swift is the most updated language for iOS development and using Xcode simplifies the process of building the application. With Xcode and Swift, we will be able to properly produce an application that can be submitted to the App Store for download. It will increase the likelihood of developing an app that is testable, usable, and visually appealing on an iOS device. Considering all of the options we have for building a native iOS application, it makes sense to choose Xcode and Swift.

III. iOS User Interface Organization

A. Sketch

Sketch is a paid program exclusively available on Mac OS X 10.10+ [5]. This program gives app designers a platform for organizing the user interface. There are many features that are available in this program, including precision, objects, the inspector, tools, reusable elements, and exporting. Precision allows scalability to be present in the app. Objects allow each shape created to be easily findable and completely editable. The inspector is the tool for dimensions, positioning, opacity and everything else you would need in order to control the design of the app. Because Sketch is exclusively available on Mac, it uses Apple's frameworks so the app will be completely supported. Most importantly, there is a Sketch app available for download on iOS devices which allows the designer to view the app design on an actual iOS device, rather than just on the computer. This program is $99.00 but there is a free trial and education discounts.

B. Interface Builder

Within the Mac native program Xcode, there is a built-in editor called Interface Builder that allows the designer to build a user interface [6]. It uses a Model-View-Controller pattern which allows the designer to build the interface without worrying about the implementation of the actual app. This program will connect the user interface to the code for the app automatically. There are multiple views in an app, and this builder uses storyboards to keep the multiple views organized. This tool gives the user the option to preview the user interface without actually running the app which saves time. Users are automatically given access to the Interface Builder when downloading and using Xcode.

C. Marvel App

Marvel App is a program that allows designers to design, prototype, and collaborate on their app designs [7]. Marvel App allows users to create two projects for free, and has paid subscriptions for any additional projects. You can use images from Sketch or Photoshop to work into your design, or you can design your interface directly using Marvel. It allows for prototyping that makes your app look and feel like it is a real app–which allows user testing to begin on just the basic design of the app. The tool used for designing is called Marvel Canvas and it can be used on any computer without downloading. This tool allows users to sync their designs to Dropbox or Google Drive so designs are automatically saved. One great feature is being able to upload sketches that you have done on paper by using the Marvel app on the iPhone.

D. Goals

The goal of comparing different ways to design the user interface is to obtain the best way to organize our iOS app. There are many programs out there that will design the app for you, but we want an app that is simple, yet sophisticated, and works for the general public.

E. Criteria Evaluation

The criteria we will use to determine iOS UI Organization includes:

1) **Price:** What type of price is associated with the program?
2) **Easy integration:** Will the UI program easily integrate with our chosen app development IDE?

F. Table Comparison

TABLE II
COMPARISON OF iOS UI ORGANIZATION

Tool	Brief description	Rationale
Sketch	A paid program that allows for designing a user interface that has many features	A great option if price were not a factor
Interface Builder	Built in with Xcode so it allows for the designing of the app and programming to be done all in one program	Best option due to price and simplicity of only using one program
Marvel App	A program that combines design, prototyping, and collaborating for iOS applications	Good option despite its pricing because it allows two free projects

G. Discussion

All three options presented here would be extremely useful in designing an iOS app. The Interface Builder makes a lot of sense purely because it allows us to use one program for both the development and design of the app, rather than having to switch between programs. Sketch has wonderful reviews for designing an iOS app and could work well if price was not a factor. Unfortunately, $99.00 is not in the budget for this project. Marvel App allows for two free projects so price ends up not being a factor since we are only working on one project. The prototyping feature of Marvel App is attractive and would be useful for this project.

H. Selection

Despite the obvious positive attributes of Sketch and Marvel App, using Interface Builder still makes the most sense for this particular project. Our app is not going to be overly complicated in its design, so using the storyboards and views offered in Interface Builder should be enough tools to build the simple design we are looking for.

IV. ANDROID DEVELOPMENT PLATFORM

A. Android Studio

Android Studio is the official integrated development environment (IDE) for Android. [8] It provides the fastest tools for building apps on every type of Android device. Android Studio comes with tons of tools that will make development for Android much easier. The IDE has a feature called Instant Run, which allows you to push code and resource changes to your app while it is running on a device or emulator to see the changes instantly come to life. This feature can drastically help speed up development, as we will not have to constantly rebuild the app and start up the emulator each time we want to test a new change. Android Studio also features an intelligent code editor, allowing us to write better code, work faster and be more productive. Android Studio is built on IntelliJ and is capable of advanced code completion, refactoring and code analysis. It also has a fast emulator which will allow us to quickly get our app up and running for testing and debugging. Android Studio also has the following features: a flexible Gradle-based build system, GitHub integration, extensive testing tools and frameworks and lint tools to catch performance, usability, version compatibility and other problems.

B. Appcelerator

Appcelerator is a platform which provides everything you need to create native mobile applications - all from a single JavaScript code base. [9] Appcelerator offers direct access to native APIs using Hyperloop, delivers fully native apps for rich user experience, immediate support for each new OS release, and seamless integration to existing continuous delivery systems. Appcelerator also comes bundled with your own MBaaS (Mobile Backend as a service). This feature, called Arrow, is a powerful opinionated framework for building and running APIs. Arrow would allow us to deliver data to any app client, engage users with pre-built notification capability, trigger notifications based on user location or predetermined schedule and view notification history and details. This could be extremely helpful for the utilization of push notifications for Church members. We could have schedules that would send out a notification at 10:15 AM with the morning bulletin for users that are at church that morning. This would be very useful as we would not be sending out notifications to every app user since not everyone might be able to attend that particular day. Appcelerator also comes with real-time mobile analytics for every native app - whether built on the Appcelerator Platform or directly via native SDK (iOS & Android). This provides a mobile lifecycle dashboard for visibility into all apps and performance and crash analytics. This feature could be extremely helpful for the church so they can monitor who is using the app and get some visual feedback on its success.

C. Adobe PhoneGap

PhoneGap is a platform used to create mobile apps that are powered by open web technology. [10] PhoneGap allows you quickly make hybrid applications build with HTML, CSS and JavaScript. It also allows you to create experiences for multiple platforms from a single codebase so you can reach your audience no matter their device. PhoneGap includes PhoneGap Build, which takes the pain out of compiling PhoneGap apps. Build allows you to get app-store ready apps without maintaining native SDK's and compiles it for you in the cloud. PhoneGap includes a desktop app for creating apps without using the command line, and a mobile app to connect your device to your development machine to see changes instantly. This could be a great resource to utilize since it is free, and the apps are created using HTML, CSS and JavaScript rather than Java.

D. Goals

The goal is to determine which development platform we would like to utilize for creating our Android version of the app. Since we need to create an app for both Android and iOS, it is important to determine how we want to go about that. We can create applications using the native SDK's through Android Studio and Java, or we could utilize a third party application such as Appcelerator or PhoneGap which will work using HTML, CSS and JavaScript. The goal is to determine which method we think would be most effective for creating our application.

E. Criteria Evaluation

The criteria for determining which development platform we use will be based off:

1) **Usability:** Is the development platform user friendly, have the correct tools we need to utilize and allow for quick development.
2) **Price:** Is the development platform free and open source, is there an upfront one time cost, or a recurring cost.
3) **Language:** Does the development platform utilize programming languages which we are already familiar with, or will we have to learn a new language to create a mobile app.

F. Table Comparison

TABLE III
COMPARISON OF ANDROID DEVELOPMENT PLATFORMS

Tool	Brief description	Rationale
Android Studio	Using Android Studio and the native Android SDK	Best option for creating a native app built in Java that will be platform specific
Appcelerator	Using the Appcelerator Platform to create an iOS and Android app in JavaScript	Best option for the number of features, can be written in JavaScript but is not free
Adobe PhoneGap	Using the Adobe PhoneGap platform to create an iOS and Android app build in HTML, CSS and JavaScript	Great option for usability since we can utilize previous Web Development skill

G. Discussion

Android Studio offers the most pure Android development experience as it utilizes the native Android SDK. This makes it a strong contender for the platform we will choose for development. In addition to utilizing the Android SDK, it has features such as Instant Run which would allow us to constantly make changes to the app while it is running to see the changes occur instantly. This makes Android Studio very powerful and helpful through the development process.

Appcelerator is another strong contender as it provides a ton of functionality since it comes bundled with a few other features such as Titanium, Arrow and user analytics. While this could provide a lot of good information for the church once the app is up and running, it does cost roughly $400 dollars a year which is a bit pricey. To get the most out of this platform and the best return on investment, it would require the church to have someone using all these features by actively monitoring analytics and crash reports. We think that even though this looks like it is an incredibly powerful platform, it would be underutilized and provide more functionality than needed for the scope of this app.

Adobe PhoneGap does seem like a very promising option as it allows the app to be developed in HTML, CSS and JavaScript. This is a plus since our team is already familiar with these languages and there would be less of a learning curve then utilizing Java. PhoneGap comes with a desktop app and mobile app for easy development. Rather than having to deal with compiling the application PhoneGap handles that in the cloud. This could be a really good option as it might be simpler to use with less of a learning curve than other platforms.

H. Selection

Based on the scope of this project, we have decided that utilizing Android Studio with the native Android SDK will be the best option moving forward. Since the team joined this project in hopes of learning more about mobile app development, we think that utilizing the native Android SDK will give us the best learning experience. While developing the app in Java may have a steeper learning curve than HTML or JavaScript, we believe that we will learn the most by using the language that most Android apps are written in. We hope that we will also be able to create a smooth and well polished app using Android Studio and features that it comes with. As Android Studio if the official IDE for Android, we believe it is the best platform to use.

V. Android User Interface Organization

A. Tab Navigation

There are many different ways in which we can structure our Android app. Navigating between pages and windows is a large part of the User Interface, and we want to make sure we are using the most intuitive method for navigation. One method would be to use tabs which make it easy to explore and switch between different views. Tabs enable content organization at a high level and can be used for switching between views, data sets, or functional aspects of an app. [11] The utilization of tabs would allow for swipe gestures between the various tabs of the application. Because it uses swipe gestures, we would need to make sure to not use content which also supports swiping. Tabs provide an easy to use and recognizable view which immediately lets users know that they can swipe between the content listed at the top. Tabs are a good choice because it provides all of the content directly on the first page for users to see. This would prevent users from getting confused since the navigation pane will not be hidden.

B. Side Bar Navigation

Side bar navigation is another UI layout which creates a navigation bar to the left of the content. Side nav bars can be pinned for permanent display or can float temporarily as overlays. [12] Temporary nav drawers overlay the content canvas, which is likely how we would use a side nav bar in our application. Side nav bars are an excellent option for navigation since it essentially hides the navigation pane until the user swipes in from the left of their screen. This allows for the focus of the page to be on the content and would allow us to maximize the space that the user interacts with. Instead of taking up screen space for a navigation pane, we can hide the navigation panel and have it accessible to the user from any page. Side nav bars are extremely popular in most material design inspired applications, and would fit perfectly in our app as well.

C. Bottom Navigation

Bottom navigation make it easy to explore and switch between top-level views in a single tap. Tapping on a bottom navigation icon takes you directly to the associated view or refreshes the currently active view. [13] This is a pretty standard UI layout that can be found on the majority of iOS applications. This would be a great choice for being able to match the layout we use for our iOS application. All of the pages are neatly laid out at the bottom of the screen, so it is easy for the user to know exactly which pages are available and how to access them. The only downside of this layout would be for Android phones that have on-screen soft buttons rather than off-screen hardware buttons. This could potentially provide an issue for users trying to access certain content but rather then clicking the page they want, they accidentally click the home button instead. Despite this, bottom navigation would be a great option for continuity across both of our mobile applications.

D. Goals

Our goal is to determine which UI layout we want to use for our Android application. We want to narrow down the three main UI layouts to determine which one will be most user friendly and most intuitive to app users.

E. Criteria Evaluation

The criteria for determining which UI layout we use will be based off:

1) **Continuity:** Does this layout provide a similar look and feel to our iOS application.

2) **Usability:** Is this layout easy to use and does it provide enough context for the user to know how to navigate the application.
3) **Intuitiveness:** Is this layout simple and easier for anyone to understand and are they able to navigate the various pages without being walked through the process.

F. Table Comparison

TABLE IV
COMPARISON OF ANDROID UI ORGANIZATION

Tool	Brief description	Rationale
Tab Navigation	Layout that includes tabs at the top of the page	A great option for being able to quickly swipe between pages
Side Bar Navigation	Layout that includes a nav drawer overlay	A great option to have a hidden navigation bar that is accessible from any page
Bottom Navigation	Layout that includes bottom buttons	Col 3 A great option for continuity with our iOS app and easily viewable information

G. Discussion

The use of tabs for the navigation of the app is a very intuitive and easy layout for people that are familiar with Android devices. On Android, almost everything can be done by a simple swipe motion. This is less common on iOS devices, so it could be confusing for people that are used to iOS instead of Android. This is also true of the sidebar navigation pane, which is hidden be default unless the user swipes in from the left or clicks on the hamburger menu. For continuity, the use of the bottom navigation would make the most sense as a user switching from iOS to Android will find it similar. This does however cause issues with some Android users that have on-screen soft buttons.

H. Selection

While the sidebar can be a little less intuitive than other UI layouts, we think this will be the best option to use in our application. With the navbar off to the left side of the page, it will allow us to focus on the clarity of the content that we are presenting to the user. This frees up screen space by not using it for tabs or buttons. This also allows the user to easily get to the settings and access all other pages by swiping in from the left. We believe that the side navigation bar will be the most intuitive and easiest to use once people use it for the first time.

VI. CROSS-PLATFORM DEVELOPMENT

A. Sencha

Sencha is a platform with the purpose of creating web and application platforms for developers using a code base of HTML5 [14]. The benefits of creating apps in HTML5 can be pretty major when you look at them. The first and biggest benefit is that it provides a single codebase for all platforms, Android, IOS, and even web viewing. This means that when you want to update your code or something about your application, that you do not need to edit multiple codebases. A

strength of this is when people open your app, they will be having the same experience regardless of which platform they are viewing it on. This provides consistency between your user bases which leads to less confusion. Sencha being a HTML5 development framework also has the benefit of hosting developers and many code review services which is good for the longevity of the project. This would mean that we could get the project completed and functional to the client's desires, but later it would give our client a resource to maintain and update the app after we are gone. One of the downsides to this platform and method is that often times, cross platform interfaces can be difficult. When working with different hardware and peripherals there can become redundancies within your app. Also, when working with HTML5, JavaScript and CSS, the applications speeds might not be as responsive as native applications.

B. Appcelerator Titanium

Appcelerator is a platform to write mostly cross platform apps using JavaScript [9]. The platform claims to reuse 60-90 percent of code when developing cross platform but provides native app speeds and functionality. The platform boasts an intuitive visual design process for all operating systems. It also allows you to compare your different UI differences between OSs include visually impressive visual effects through JavaScript. It also has a framework for building and running different APIs with other applications and working with existing systems such as CCB. Some of the downsides being though that it is not truly a cross platform development strategy as different device platforms would have different codes. Also it seems that the use of this coding platform and their site requires a subscription model to Appcelerator which can become costly. One benefit to this though is they provide their own database storage, can complete API calls and have a built in file storage for development purposes.

C. Apache Cordova

Apache Cordova is a free, open source coding API that provides nearly 100 percent code re-use between platforms using JavaScript [15]. Cordova works in conjunction with visual studio for easy setup when starting out your project. The way it works from a high level is it takes advantage of the universality of HTML and JavaScript while simply managing the OSs specific API to interpret the page. This would mean, like Sencha, that the app would be available in many ways beyond just iOS and Android platforms. Cordova has CLI, which is a high-level tool they developed that allows you to develop for multiple platforms simultaneously by abstracting a lot of the functionality of lower level shell scripts. The downsides of this option is because it is a free and open source, it lacks a lot of the support and functionality of the above paid options. It does not have built in cloud storage, no database to use for the app post launch and no developers working on their end. Meaning that to modify the app after our development finished outside help would be necessary.

D. Goals

The goal here is to determine which platform we would use if we choose to develop our application to be writeable cross-platform. There are many frameworks and platforms available to us to code our application so it is best to scope out the different options, weigh the pros and cons of each option and make an educated decision based on that info. We want to develop our app in an efficient and low cost manner.

E. Criteria Evaluation

Our criteria for determining which platform to use will primarily come down to pricing. However it will also factor in the amount of features each tool or method will bring to the table. Along with that it will be noted the language being coded in and its strength within the scope of the different operating platforms.

F. Table Comparison

TABLE V
COMPARISON OF CROSS-PLATFORM DEVELOPMENT

Tool	Brief description	Rationale
Sencha	Using HTML5 and existing frameworks	Strong option for the longevity of the app
Appcelerator	Creating native apps for each OS while reusing 60-90 percent of code	Visual design process and backend features are amazing, but come with large pricetag
Apache Cordova	Using their custom CLI to code in HTML and JavaScript	Solid option with less features but the great price of free

G. Discussion

Appcelerator seems to be the strongest option in the perspective of what it provides to us. The amount of features, including cloud storage, a built in database, user analytics, and visual design process. However the price is very high and on a subscription model to upkeep the code base, and database.

Sencha appears to be a mix between Appcelerator and Apache Cordova. It provides less features then Appcelerator but they have a team of developers that you can hire to work on and maintain your application. This could be helpful less to us but our client for the longevity of their application.

Apache Cordova boasting a perk of being open source and free is too big to overlook. When working for a client that represents a non-profit organization it is crucial to keep costs low in the scope of the project. Apache Cordova would provide us the tools to get the job done while not costing us a cent.

H. Selection

Based on how heavily price will weigh in on our client we think that if we choose to go with a cross platform development strategy that we should go with Apache Cordova. It give us a lot of the tools we will need, is free and open source, and allows us more freedom to integrate with our clients existing database/back end.

VII. INTEGRATING THE GIVING PLATFORM

A. Authorize.net

Authorize.net is a payment gateway which enables Internet merchants to accept online payments via credit card and e-checks. [16] This giving platform allows you to accept credit cards and electronic checks from websites and deposit funds automatically into your merchant bank account. Authorize.net allows for all major credit cards, signature debit cards, linking of bank accounts, and digital payment solutions such as Apple Pay, PayPal and Visa Checkout. It also comes with fraud

prevention to identify, manage and prevent suspicious fraudulent transactions with address verification service and card code verification. This platform also gives you online access to manage and review transactions, configure account settings and download reports with the ability to sync to Quickbooks. In addition to online payments this platform accepts payments that are made on a mobile application and they also provide a free mobile app. Authorize.net charges 2.9% plus 30 cents per transaction with a $49 setup fee and a $25 monthly gateway fee. Authorize.net is currently what Calvary Chapel is already using as their platform for giving, so utilizing this platform will probably be the best option for our application.

B. Pushpay

Pushpay is another giving platform which allows you to make collections for your organization in seconds. [17] Pushpay is one of the fastest ways to give to your church and requires only three steps to give. You need to download the Pushpay app, choose who you want to pay and how much, and enter your passcode and the donation is complete. Pushpay allows you to pay any registered merchant and includes geo-location and search technology to find who to pay. Pushpay makes giving extremely easy, and provides the users with a simple, easy to interact with application that makes giving a breeze. This is a great option for giving as it provides a mobile application and website which you can give securely from. It hooks in directly to the Church's merchant account and can be up and running quickly. While this is a very simple and easy option, it is a bit pricier compared to Authorize.net.

C. PayPal

PayPal offers a service which allows people to donate to your non profit or organization. [18] It allows you to add one simple button to your website or app which lets you accept credit cards, debit cards and PayPal. Users do not even need to have a PayPal account in order to donate via the PayPal Donate button. This provides an extremely easy to use interface, and would allow us as developers to easily integrate giving into the application. PayPal allows you to get access to your funds quickly as the donations process within minutes. Once the donation is processed, you can then transfer it to your organization's bank account at no charge. PayPal is a great option since there are no charges associated with donating.

D. Goals

The goal is to determine which giving platform is the easiest to use by church members, provides the simplest and fastest way to donate money, and is secure.

E. Criteria Evaluation

The criteria for determining which giving platform we use will be based off:

1) **Security:** Does this giving platform have the technology to ensure that all payments are secure.
2) **Reliability:** Does this giving platform perform well and allow users to give when they want to give.
3) **Usability:** Does this giving platform have a simple, easy to understand interface for users.

F. Table Comparison

TABLE VI
COMPARISON OF GIVING PLATFORMS

Tool	Brief description	Rationale
Authorize.net	Secure and simple giving platform currently in use by the church	Great option because it is what the church is currently using
Pushpay	Simple, fast and intuitive giving platform	Great option for speed and usability for users
PayPal	Simple giving platform that does not require a fee	Great option for allowing users to give without setting up any type of account

G. Discussion

The various giving platforms that we have looked at are all pretty similar in that they all essentially provide the same functionality. When it comes to giving, you do not really want a platform with tons of features that would just dilute the process of giving. It should be something simple and easy that can be done within seconds whenever the user would like. Because giving is generally a monthly occurrence, having a platform which supports automatic recurring donations would be a great feature to have.

Authorize.net is a great option because it is what the church is currently using as their giving platform. Pushpay provides a wonderful user experience and makes giving extremely simple. Just three clicks and you are done. PayPal also is another great option as they allow you to just include a button on your website or application. This makes the integration with the giving platform very easy. This is also a great option because it does not require users to create any type of account. They can just give whenever they like using credit, debit or PayPal.

H. Selection

Despite having a lot of good giving platform options, the one that we have decided to stick with is Authorize.net. Since this is the platform that the church is already using, it makes the most sense to integrate that into our application rather than having the church switch their platform just for this application. While the other options may have great user-friendly UIs, we want to create an application which will utilize what the church currently has in place.

VIII. INTEGRATING E-BULLETINS

A. Contentful

Contentful is a content management system that allows content to be presented on a website or application through their API [19]. There are different products available from this company, but one in particular would be best in delivery the E-Bulletins to our app. The Content Delivery API is a tool that pushes read-only data to the app. This is ideal for blog updates and similar information that is not ever going to be edited within the app itself. This API takes advantage of the Contentful Web App which is the online editor that developers can use to create or update their content. Contentful has multiple pricing plans, including a free one that is described as good for small projects. An advantage that this system has is that once the API is set up, the content can be edited directly from Contentful and without any prior technical knowledge.

B. Wordpress

Wordpress is a powerful website management system that has the ability to create an entire website and manage the content all from one editor [20]. With Wordpress, you can create pages that can be uploaded to an application. This allows developers to update a page from Wordpress' editor and save the changes which will appear on the app without having to actually edit the app itself. Wordpress has an application framework that is used to integrate the pages into the application. This system is a great option for developers who are wanting to have the same information both on an application and a website.

C. Church Community Builder

Church Community Builder is a software program that is available to churches to use for their websites [21]. It uses API to integrate calendars, forms, and other features into existing websites. Because Calvary Chapel Corvallis already uses Church Community Builder, their development team is already familiar with this program. In order to integrate the e-bulletins, a group for announcements could be made and the e-bulletin could be uploaded to that group. To display the bulletin on the app, we would use the existing API network. Church Community Builder is a paid software system, but in our case the church has already paid for this service.

D. Goals

The goal is to determine the best way to integrate the church's existing information, in this case their e-bulletins, from their website onto the app. Currently they just update their website with the content whenever they need to, so we need to discover a way to push the information to the app without having to update and resubmit the app each time there is a new e-bulletin.

E. Criteria Evaluation

1) **Price:** Is the program free for our purposes?
2) **Usability:** Will people with little technology background be able to update the content?
3) **Continuity:** Will the program work well with the current church website?

F. Table Comparison

TABLE VII
COMPARISON OF E-BULLETIN INTEGRATION

Tool	Brief description	Rationale
Contentful	CMS that uses API to push updates to the app	There is a free option, but it may not be what our project needs
Wordpress	Website management system that can push updates to an app	Would be a better option if the existing website was also using this system
Church Community Builder	A Church management system that has API ability	A great option as the church already uses this system

G. Discussion

Contentful would be a good option if we were developing a more complex app. Because we are only looking to integrate this type of content for the bulletins, Contentful might be a bit too powerful for our purposes. Wordpress is a better option for a less complex app if one is already familiar with how Wordpress works and if they were utilizing that tool in their existing website. Because they are not using Wordpress at all in their website, it may be too complicated to add another tool for them to keep track of. Church Community Builder seems to make the most sense because they already use this system. Their staff already knows how to use it and there would not be as much of a learning curve to learn how to update the e-bulletins.

H. Selection

For this project, the option that will work best is Church Community Builder. While the other options offer useful features, the Church already utilizes Church Community Builder so there will be no additional costs or learning to integrate the e-bulletins using this program.

IX. Integrating the Sermon Platform

A. LiveStream

Livestream is a video broadcast and viewing platform that allows groups to distribute their events to anyone who is interested or subscribes to their content [22]. They provide the ability for their clients and users to stream their content live for the viewers to watch and then host the video for later viewing afterwards. They host integration with a variety of different systems including web access, through their app, or many popular video viewing platforms such as Roku or Chromecast. The major benefit to using livestream is that our client has existing knowledge of this platform and already is hosting a lot of their content through them. They do have an associated cost that comes with the platform, but it comes with more support than many conventional video hosting sites.

B. YouTube

YouTube is one of the world's most popular and well recognized video hosting platforms [23]. They have recently come to introduce live streaming capabilities and continue to dominate the video hosting market. Because of their sheer size they have the greatest range of integration with various video hosting platforms and a well-polished and understood API for us to use. They also have their own application that could be linked with our app for seamless viewing. They do not charge for their video hosting and can even pay the user for their content if they choose to enable adds which is YouTube's revenue source.

C. Vimeo

Vimeo is another popular video hosting service that many smaller organizations look to [24]. It allows for 500MB of weekly uploads free of charge up to a total storage of 10GB. If these totals do not allow for enough storage the user can upgrade to a paid account with fewer restrictions. They have an API for use that can be accessed upon approval from their team. They have a larger support team then the automated YouTube while having lower prices than livestream making them a mix of the two services of sorts. They also offer production assistance through their team to their team/organization accounts.

D. Goals

Our goal is to make our clients sermons available for viewing within the app in a way that is easy for the user. We also want the upload process to be as easy as possible for our client to reduce upkeep of our app. They want to continue to upload their content as they have in the past but have it available through the app.

E. Criteria Evaluation

We think the primary criteria to evaluate is how well it will integrate with their existing systems. If it requires an overhaul of how they upload videos then this would be considered a failure. The next most important criteria we believe is the pricing. Keeping costs low for non-profits is important so we are aiming to keep any recurring costs as low as possible for our clients.

F. Table Comparison

TABLE VIII
COMPARISON OF SERMON PLATFORMS

Tool	Brief description	Rationale
LiveStream	Business oriented video hosting that comes at a premium	Is the churches existing video platform
YouTube	Most popular video hosting platform with high usability	Free and well known for being a top tier platform
RVimeo	Fairly well know video hosting with high quality platform	Solid hybrid of YouTube and LiveStream

G. Discussion

So we discussed earlier that the biggest criteria for evaluation will be how well it integrates with their existing systems. LiveStream earns itself a huge leg up on the competition because it is the platform that our client is already using. This would mean that they do not have to change how they upload video from their sermons and would overall be the easiest option.

Youtube is still enticing because of the fact that it is so recognizable. On top of this they also have a giant perk of not only being free, but offering a share of their add revenue to their content creators. Their API would be usable to us as it is one of the most commonly used APIs for this style of content.

Vimeo, while being a combination of the two styles, is far less desirable because it is not already integrated with the client and it's simply not as good as YouTube. It is fairly clearly the worst option when you factor in that their API is only usable once approved so it would leave up the possibility of being denied.

H. Selection

While YouTube is such a strong option, we will be going with LiveStream. Our client is already hosting their videos through this service and they have requested that we leave their existent infrastructure as intact as possible. The price is not

awesome, but we can discount it in a way because it is not an additional cost. The client was already paying for this service so we are not adding to any costs for them. LiveStream on a functional level should complete the task just fine and work with our app excellently.

X. Integrating the Calendar Platform

A. Google Calendar

Google Calendar is one of the most widely used and recognizable calendars available to developers like ourselves [25]. It has tons of resources online to learn about it and a vocal user base to explain its integration with different apps. From what I can find we will be able to integrate it with our app without incurring any additional costs. We also know that because it is google that it will function identically with both IOS and android platforms which is a big plus. Unfortunately our client is in the process of transitioning their existing calendar off of google and onto CCB.

B. Microsoft Outlook

Microsoft has a calendar built into their email system that we could utilize for our application [26]. One of the big benefits outlook provides to us is that fact that windows is so commonly used among computer users. This means that along with our app, we could sync up the calendar with peoples computers which may be easier for some. Many users do not look to their phones exclusively for their scheduling and this would not alienate that part of our user base.

C. Church Community Builder

CCB is a development group centered on helping churches get on their feet in the complicated area of tech development [21]. Our client is moving most of their infrastructure over to CCB so integration with them in general is going to be crucial. Because most of the data we will be pulling is already from CCB it will be beneficial to us to reduce how many outside resources we are using and consolidate. They have an existing API that we can take advantage of when implementing the churches calendar.

D. Goals

The goal with the calendar integration is to make it as easy to update for our clients staff as possible. We want our app to have as minimal upkeep for our client as possible and whichever method accomplishes that while providing the necessary information will be the best option.

E. Criteria Evaluation

As stated above we want to evaluate the systems on ease of use for the clients on the backend. Alongside that the application of the calendar should be sleek and easy to interpret. As always, reducing costs wherever possible is also crucial.

F. Table Comparison

G. Discussion

In our eyes two of the big priorities when it comes to external API and including it in our project is to make it require as little work from our client as possible and work with their existing systems. This is the major benefit of using the Church Community Builder system that exists. T hey have an existing API that is functional for us to use as well.

TABLE IX
COMPARISON OF CALENDAR PLATFORMS

Tool	Brief description	Rationale
Google	Existing Google calendar integration with app	Recognizable and highly developed
Microsoft Outlook	Integration with windows and Microsoft Calendar	Allows higher integration with computers alongside the app
Church Community Builder	Church Management system with existing API	Church is already familiar and using this system

Google is a strong option and probably would have been a favorite if our client was not in the process of phasing out their existing integration with google. They have a strong background and very easy to work with API. Also the google branding is easily recognizable and would be comfortable for our users.

Finally Microsoft provides a set of tools that are interesting to talk about, but may not be super applicable to the project in front of us. Yes, it would integrate well with the windows operating system, but we are not building our app to work with computers. We are building an app that will work with IOS and Android so this may not be as relevant as you would think.

H. Selection

For the calendar we will be using CCB or Church Community Builder. Not only is it going to simplify the process by having more information being pulled from a central location, but our client specifically asked us to use this platform out of ease to their staff. Also the existence of their own API that we can make use of is excellent for us and will be a major tool in development of the app.

XI. Conclusion

Our main goal with the technology that we choose for our project is working well with the Church's current back-end system, Church Community Builder. Additionally, creating a product that the team at Calvary Chapel Corvallis can successfully continue to update and maintain is extremely valuable to this project. These goals have contributed to the technology that we have chosen to move forward with. Price of the technology was also a large factor that we had to consider.

The technology we chose for developing the iOS portion of the app is Xcode and Swift. We will use Interface Builder that is built-in to Xcode in order to organize the user interface. For the Android portion of the app, we have chosen Android Studio. We will organize the Android app using a sidebar organization layout.

Because Calvary Chapel Corvallis already uses Authorize.net for their giving platform, we have decided it is the best option for the donation page on the app as well. Similarly, Calvary Chapel Corvallis uses the Church management system, Church Community Builder, so we will continue to use this system to present the e-bulletins on the app. The technology we will use to display the sermons is LiveStream. This will work well despite its price, because Calvary Chapel Corvallis has already chosen to use this technology. Finally, we will once again be using Church Community Builder to display the calendar and events on the app.

The technology we have chosen to use will enable the church to successfully take over the maintenance of the app once the project is finished. It will also allow them to continue using their current back-end system, which will cut down on additional cost and education for the church staff. We are confident that the technology we have chosen will help us create a simple app that the members of Calvary Chapel Corvallis can use and enjoy.

REFERENCES

[1] A. Inc., "Start developing ios apps (swift)," September 2015.

[2] "How to make iphone apps with no programming experience," October 2016.

[3] "Appcode," 2016.

[4] "iphone app builder," 2016.

[5] "Sketch app," 2016.

[6] "Interface builder built-in," 2016.

[7] "Marvel app," 2016.

[8] "Android studio," 2016.

[9] "Appcelerator," 2016.

[10] "Phonegap," 2016.

[11] "Material design tabs," 2016.

[12] "Material design side bar navigation," 2016.

[13] "Material design bottom navigation," 2016.

[14] "Sencha," 2016.

[15] "Apache cordova," 2016.

[16] "Authorize.net," 2016.

[17] "Pushpay," 2016.

[18] "Paypal," 2016.

[19] "Contentful," 2016.

[20] "Wordpress.org," 2016.

[21] "Church community builder," 2016.

[22] "Livestream," 2016.

[23] "Youtube," 2016.

[24] "Vimeo," 2016.

[25] "Google calendar," 2016.

[26] "Microsoft outlook," 2016.

A. Technology Changes

Throughout the process of building our applications, the church was also busy working on their own website. The new church website used Wordpress as the software backing the entire site. Because we wanted to limit the amount of extra work required to maintain the application, it made sense to change the way we were incorporating the bulletin page into the app.

Originally, we proposed to incorporate the bulletin page using the database, Church Community Builder (CCB). The bulletin announcements were going to be handled in this database, which we could then request through the API. However, with the new website, the bulletins were no longer going to be updated on the database. This meant, should we choose to use this database, the church would have to update the database as well as the website in order to update the app.

Because their new website used Wordpress, we were able to make use of the Wordpress REST API. By utilizing this API, we could request the content on the bulletin page and using a JSON parser, we could display the results on the application. Now, whenever the website updates, the app will update as well.

There were no other technology changes.

VI. Weekly Blog Posts

A. Fall 2016

1) Week 3:

a) Courtney Bonn: This week we met with our client on Tuesday and met the rest of the staff at Calvary Church. At the meeting we talked about the basic idea of the application, as well as some bigger goals that we want to strive for. We also discussed what kind of design they were looking for. Like the functionality of the app, they are looking for a relatively simple design.

This upcoming week we are planning to meet with the church's website designer/developer. He has information about the church's management software that will assist us in integrating the calendar into the application. Personally, I plan to start doing a lot of research this upcoming week on creating iOS/Android applications.

b) Max Dimm: This week we actually went to calvary church to meet with our client and her churches existing dev team. We went over what their expectations and desires were out of the project. A lot of this info can be seen on our problem statement. However the just of it was that they wanted an easy to use app for IOS and android that gives their existing members an easy way to access weekly information. The info they wanted included on the app was things like a bulletin, schedule, videos of past messages, and the bible. There were other small things but those were the major ones.

Also this week we set up this blog and stated/finished our problem statement. For the problem statement I wrote about our solution and what we should be expected at the expo. I wrote around 300-350 words and Courtney helped me out by adding it to the tex document.

c) Kevin Stine: This week we met with our client and hammered out the details of what our project will look like. Since we are building a mobile app based on the mobile version of their website, we met with the media and web guys to get a better idea of what they were envisioning. Since last week we were able to get all the details down and write our problem statement.

This upcoming week I plan on getting more familiar with LaTeX for future write ups, looking into mobile development for iOS and Android, and getting more familiar with the back-end software that the church utilizes.

2) Week 4:

a) Courtney Bonn: We met with our client at the church and we were introduced to their website developer. This was mostly just an introduction where we gathered any additional thoughts he had on the project. He let us know that he would prefer to publish the app at the end of the year, so that way it's under his developer profile. I didn't do as much outside research on app development as I was hoping to do this week, but I am planning on doing much more this upcoming week. We also have the problem statement that we need to edit. I don't think we have too much editing to do on this paper, but I still plan on putting some more time into it. Additionally, we have the requirements document that we are going to start working on. I think the biggest goal this next week is to get a much more solid grasp onto our project, hopefully leading us into a good start on the actual project itself.

b) Max Dimm: This week we met with our client again to be introduced to their lead web developer. We talked for a bit about the problem statement along with the developers ideas for the project. He was interested about how the end product would be delivered to him and how he can be included in the dev process. He informed us he has the ability to upload apps to the app store. We didn't have too much to do besides this meeting as there were no assigned papers/works alongside this. We will begin the edits on our problem statement when we are emailed our corrected first draft.

c) Kevin Stine: This week we met with our client and got to meet their lead web developer. We picked his brain a bit to get a better idea of what he had in mind for the app, and how we could get access to some of their current resources. We got into talking about the actual deliverable and how we would be handing off the project and whether or not publishing the app would fall on us. This week I also took a look at some of the basics of developing for iOS using Swift. I started going through Apple's Swift Developer Tutorial to get a little more familiar with Xcode and Swift. Next week we'll be going through the requirements document and really hammer out the details of all the requirements we have on this project.

3) Week 5:

a) Courtney Bonn: This week we focused on updating our problem statement and drilling down on our requirements document. We reviewed some of the requirements document with our TA to check in and make sure we were headed in the right direction. I did some research on Swift and iOS programming. I downloaded Xcode onto my Mac to start trying and learn Swift. We are at the point where we need to start thinking about whether or not we are going to build two apps (one for iOS, one for Android) or if we are going to build one app that is cross-platform. This upcoming week we need to put in some time researching what might be best for our project. We plan on meeting with our client this week to go over the requirements document and fine-tune it before turning in the final document on Friday 11/4.

b) Max Dimm: This week we finished up our problem statement and began on our requirements doc. We also met our TA Vee who seemed pretty chill. We decided not to meet with our client this week because we didn't have much to deliver to her at this point, plus we had met every other week up until this point. On my end I was able to help a bit with both of the documents and deliver them on the due dates. I need to be more on top of the documents, because by the time I get involved they most of the time have already taken form. In the future I'd like to help design or help start the documents.

c) Kevin Stine: This week we finished up our problem statement and got that submitted based on what our client approved. I began going through a swift tutorial and have been looking into the basics for creating an application on iOS. This week we will be determining exactly what the client wants in terms of the application. Do they want two apps for both iOS and Android, or would they prefer an application for one platform in particular.

4) Week 6:

a) Courtney Bonn: This week we focused on finishing and polishing our requirements document. We got a rough timeline for the rest of our project. We sent our requirements document to get approved by our client, who agreed with everything except for just a few small changes. Next up is beginning to research for the design document and along with that we'll begin working on the technology review. This will consist of us looking into the different technology available for us to use on our project.

b) Max Dimm: So this week we were pretty much exclusively working on the requirements doc. We were able to get it done but it was a little tense getting our clients attention at one point. Once we were able to get her to work with us we could put the finishing touches on the doc and turn it in. I missed the one class of the week and my first class because I was out of town for the holiday weekend but don't plan on missing much more class. I need to start looking into app development more and finding out how I can contribute to the actual work part of our project from this point on.

c) Kevin Stine: This week our main focus was completing the requirements document. We really took the time to verify that we documented everything that we would need to be doing for this app. This week we'll be getting started with the design document and tech review, so we'll need to really hammer out our design on the application. I plan on looking more into the different design guidelines for both iOS and Android, and coming up with some good designs and ideas for how we want to structure our application. I've been going through a Swift tutorial and will continue to explore swift in the hopes to use it for our iOS application.

5) Week 7:

a) Courtney Bonn: We focused heavily on the technology review this week. We didn't meet with our client as we didn't have any new material to go over with her quite yet. We met up just with our group and hashed out the different parts of our system that we need to researched. Once we figured out who was going to be working on each part, we each set out to start our individual research. I took on iOS development platform, iOS user interface organization, and integrating the e-bulletins. I'm fairly confident that each technology I chose will work well with our client, as I kept in mind the requirements and exactly what the client is expecting. This next week we will start working on the design document.

b) Max Dimm: This week we are starting on our tech review which is gonna be a big project. We did not meet with our client because we did not have anything worth reviewing to go over with her yet. However we did meet as a group outside of class to go over what each of us was going to work on in the tech review. We started the process of writing but most of that will likely come down to this weekend and next week early on. I was assigned to talk about cross-platform development, sermon integration, and schedule integration.

c) Kevin Stine: This week we got working on our tech review and I spent most of the week researching and figuring out what other technologies were available for us to potentially use for our application. I focused mainly on the Android development platform, Android UI design, and integration with a giving platform. Researching and seeing what other technologies are available was really helpful to get a better idea of what options we have moving forward. Moving forward I'll be starting to get our design document setup so we can start figuring out exactly how we want to design our app.

6) Week 8:

a) Courtney Bonn: We finished and turned in our technology review this week. The next task is to work on our design document. I think we have a good feel for what we need to do, though we haven't begun working on it quite yet. My plan is to work on my portion of the design document at least a little bit before the Thanksgiving holiday. After that we will have the progress report along with the recorded presentation to work on for finals weeks.

b) Max Dimm: I got a bit of a late start on writing my part of the tech review. I spent most of the weekend and Monday working on my pieces of it but was able to get my portion done in time. It took a bit longer then I expected as the amount of sections each part was fairly large. This week we will begin looking forward and running our tech review by our client. We also did not meet as we figured the whole meeting would just be reading through the 23 page doc. We decided just sending them the doc to read on their own would make more sense.

c) Kevin Stine: This week we wrapped up our tech review and got it turned in. I spent a lot of time looking into the Android UI design that we want to use and it was good to dive deeper into the layout that we want to use in our app. Next we have the design document which will require more detail and figuring out exactly how we want to implement our application.

7) Week 9:

a) Courtney Bonn: With Thanksgiving holiday this week, we didn't meet with our TA or work on the design document very much. I began the outline for the document and began working on the first four sections. This upcoming week will be very busy with the design document as well as hopefully meeting with the client to go over our decisions. We will also begin working on the progress report and planning how we will complete the recorded presentation.

b) Max Dimm: We had a short week this week only meeting on Tuesday. We wrote to our TA Vee a short bit about what we were excited about and what we were worried about. Other then that we started our design document but I was not able to contribute much at this time due to being out of town. Next week I plan on getting my share of the work done whatever that may be. We also did not meet with our client due to the holiday weekend but we will most likely meet with them the following week.

c) Kevin Stine: This week was a short one as we had Thanksgiving. I didn't really do much for this class this week but have begun thinking about the design document. I'll be doing a lot of research into the various viewpoints that we'll want to incorporate into our application. We'll probably meet with our client next week as it'll likely be the last time before Christmas break so we can answer any questions they might have.

8) Week 10:

a) Courtney Bonn: With this last week of the term, we really drilled down and focused on the design document. I think we underestimated the depth of this document. Because the app that we're designing is supposed to be a simple design, we didn't realize how much effort and time would go into planning the design. We did end up finishing the design document and I think we did a great job. We also met with our client for the last time this term. We made sure we were all on the same page and answered any questions the client had. I have a good idea of where this project is going and I think it's going to be successful. During the break, I plan on doing a lot more research on iOS and Android development so that way when we start implementation in the Winter I know more of what I'm doing.

b) Max Dimm: We got off to a bit of a slow start on the design document but were able to finish it on time. We were not super clear on the depth that was required by some of the sections/viewpoints. We were able to meet with our client before the end of the term which was good. We were able to send her the documents we had been working on and just make sure that our visions were still in sync for the project.

c) Kevin Stine: This week we got going on the design document. Since we are all pretty new to mobile development, we weren't entirely sure how to go about mapping out our design processes for the entire application. As we begin to learn more about mobile development I'm sure we'll be able to add more in-depth information to our design document, but we had

to get as much planned out based on our current skill level. We also met with our client for the last time before break so we could make sure to get on the same page about everything moving forward. Once break begins I plan on heavily researching iOS and Android development. I'd like to have a full-functioning layout of our application for both platforms before Winter term begins so we can focus on integrating the Church's database and information into our app without worrying about the layout.

B. Winter 2017

1) Week 1:

a) Courtney Bonn: During Winter break, I didn't accomplish nearly as much as I wished I had. I ended up being very busy with work and family obligations and wasn't able to work on this project. I have started looking in to Android Studio as I am using it for another class so this will be beneficial for our project. My goal within the next week is to finish up the Swift tutorial so we can get going on our project. Kevin set up the skeleton of our iOS app and we verified we were all able to pull it from Git and run it. I've also touched base with our client. We've agreed that we don't have any reason to meet thus far and will put our first meeting off a few weeks until we have something to present.

b) Max Dimm: Over the break I really only downloaded x-code and experimented with it without much real purpose. I looked up a few guides but knowledge did not really stick. The first week I spent my time downloading getkraken on my MacBook and downloaded Kevin's skeleton. We did not meet with our client because we didn't really have much to go over with her but we plan to do so in the next week or two I think. My goals for next week are to start looking over some of the swift guides online and start learning the language.

c) Kevin Stine: Over break I really didn't have much of an opportunity to dive deeper into App Development besides for just looking at some of the iOS documentation. I was able to setup a pretty basic app with multiple pages like we want for our application for Calvary Chapel. This will work pretty well as a foundation for building up the other aspects of the app, however there is a lot that we'll need to do in order to get this app fully functioning in the next few months. My next steps will be to dive deeper into looking at the documentation for the various APIs that we wish to use in order to bring in the functionality for the calendar or sermons.

2) Week 2:

a) Courtney Bonn: This week I focused heavily on learning Swift and Xcode. I wasn't able to work on it during break, so I do feel a little behind in terms of our project. We have a basic template for our iOS app that has different pages and icon images, but our next step would be to start pulling in the actual information. I've emailed our client to get some clarifying information on the Church's new website. I believe the new website is using Wordpress (from looking at the site and the Web Inspector tool via Safari) which will allow us to use Wordpress' API to bring information to the app–allowing the web development team to only have to update the site and it will in turn update the app. I'm waiting for an answer on this now and will then start looking into Wordpress API further.

b) Max Dimm: This week I began the process of looking over the swift guides we found online. I don't think I'm currently where I want to be with learning the language just yet. It's starting to feel like it might be one of those things where I just need to start working on it and look up the parts as I go. I believe my next steps are to either start working on the sermons functionality or the donations. I will need to see how to go about how to implement the IOS media player in the app so we can show the messages.

c) Kevin Stine: This week I focused more on getting familiar with iOS development and I continued to refine the basic framework of an app that I created earlier. I've mostly been adding small features and getting a feel for how everything is laid out in Xcode so that as we move forward I'll have a better grasp on the platform we're using. I will begin working on getting the events page setup with a table view that pulls from the Church's database and populates the table with the events as well as the dates.

3) Week 3:

a) Courtney Bonn: This week I worked on getting Calvary's logo on each page of our iOS App. It took a bit longer than I expected because even though it seems like an easy concept, you have to add constraints so that way it will be in the same position on every size iPhone. I received confirmation that our client is using Wordpress for their new website and that can have access to it's API. I've been busy doing a lot of research on using Wordpress API while waiting for the Wordpress access. Once we get access, we can use REST API and parse using JSON (this will obviously be separate from the XML parsing).

b) Max Dimm: I started working on implementing the video player this week for the sermons. I have a pretty good idea of how it all needs to go down but I haven't started the implementation on our project just yet. I started by implementing it on a separate project to make sure I could do it before making edits to our actual project. I was able to get it to play a youtube video but need to make it work with livestream because that is the platform our client will be using.

c) Kevin Stine: This week I got more familiar with Xcode and configuring the build settings for our application. I also began working on getting a table view setup so that we could get the Events (calendar) view setup to have the framework before we begin populating the table with values from the database. We are still waiting on our client to get access to an admin account which will let us connect to the API, but once that's done we should be able to the XML data parsed using NSXMLParser and pulling in the correct data for the events.

4) Week 4:

a) Courtney Bonn: This week was a busy week in regards to our project. I worked on the bulletin page on the iOS app and was able to successfully get a JSON parser up and running. The parser is currently getting data from a JSON dummy site because we still don't have access to the Wordpress API. Once we get the Wordpress API credentials, I should be able to just change some variables and the URL in the current code to point to the Wordpress site and should be able to print the data from the bulletin page. I also began the Android app. I got a simple app up and running with side bar navigation drawer that goes to each page we need. This took a lot of time and was more difficult than I expected. The Android studio interface itself is not as intuitive and user friendly as Xcode. We also began work on our Winter Progress Report and sent a first draft to our TA to make sure we are on the right track. Up next we need to focus on the Android app and get that quite a bit further. We plan on meeting with our client on Tuesday 2/7 to have them look at what we have so far and get some questions answered.

b) Max Dimm: Ok, so I came upon a stumbling block this week but was able to figure it out by the end of it. I was confused about linking the visual aspects of the code to the ViewControllers. Whenever I went to connect the two it seemed to link me to an uneditable file. By the end of the week in class I was able to figure it out with Courtney's help which was awesome. So now the same video player that I made for the separate app is now working with our live project. I just wanna make it work with the livestream player which shouldn't be that hard.

c) Kevin Stine: This week I mainly stayed focused on the events page which I've been working on. I spent a lot of time getting the UITableViews setup so that the events could be displayed in a table form. I got stuck a bit stuck on creating a class for the UITableView, so I began working on the implementation of parsing the XML data from CCB. Once we were able to get the username/password for connecting to the CCB Public API, I was able to write a quick python program which connected to the database and then parsed through the XML data. I then got started on implementing a similar program in swift which is what I'm currently working on now. Once I get the XMLParser to work in Swift, I'll focus on getting the class to work so I can pass the data from the XMLParser into the UILabels.

5) Week 5:

a) Courtney Bonn: This week I started working on our Android app. I was able to create a basic app with an empty activity. After some trouble, I was finally able to get a sidebar navigation to work so now our app has the different pages we will need. I also began researching how to parse JSON data and display it in Java/Android. This is a harder job than I expected and I've had a tough time finding a good tutorial that will help me with this. We don't need a complex parser for the bulletin page and most things I'm finding on the internet are a little more complex that what our project calls for. This next week I will continue researching to try and get this part of the Android app up and running. At our meeting this week we were able to show the client the products we have so far and they liked the direction we were going with the color scheme and the simplicity of the application. We still haven't gotten the API credentials for the Wordpress website so until then, we are stalled on the bulletin page. Our client asked us to look into how easy/possible it would be to offer the app in different languages and this is something we are currently researching. They said at the very least they want to be able to have the welcome message on the homepage in different languages. We said we could do this in a scroll fashion where the different messages scrolled on the screen through the different languages. The client agreed that if we weren't able to have different languages for the app altogether, we could have the message scrolling. This next week we have our progress report due. We haven't started working on it yet and I'm a little stressed out for time but I think we should still be able to finish it by the deadline.

b) Max Dimm: I did some research about the live video embed link that livestream gives out but it seems that our client will need to upgrade their account to the premium service to access it. Because of this an a few other reasons we met with our client and I got some feedback on that. It seems they will either upgrade their account or upgrade their infrastructure and include youtube, either will work for me. So now I gotta start working on adding buttons to change the viewable video and start looking into the donations page.

c) Kevin Stine: This week I continued my work on the Events page for iOS and got the basic framework for the events to work and was able to fix the issues I was running into previously where the tableView wasn't working properly with the eventView. I was finally able to get the events page to work without crashing by recreating the class for the UITableView. It took a while of checking online to try and troubleshoot the issue, but now that I have that part working, I can work more on the XMLParser and the details page for the events page.

6) Week 6:

a) Courtney Bonn: This week we focused on writing our progress report and recording our presentation. Our apps are not exactly where we wanted them to be for the alpha release so we spent some time getting them a little further before we finished our presentation. I worked more on the Android app and added a home page fragment and then added that page to the navigation. My next step is to get a JSON parser working on the Android app. We are still waiting for Wordpress

credentials so I'm still not able to pull the correct data to the bulletin page.

b) Max Dimm: I was able to put aside most of my other work to put time into the presentation this week. I was able to write about 800 words on my experience and record voice over for the first 7 or so slides. I pushed any changes I had made to the app so that when Courtney and Kevin did their recordings of the app demonstration that they would see any changes that I had made up through this week. I need to work on implementing a few bits of functionality to the app that will increase usability.

c) Kevin Stine: This week we focused mainly on our progress report and the recording for our presentation. As I was able to get the Events table view page to work, I mostly cleaned up a few things on my code which allowed me to display test data for the various labels that I have for the event name, day and month. Next I will be working on getting the XMLParser working since now that I can display data, I just need to parse the correct data from the CCB API.

7) Week 7:

a) Courtney Bonn: This week I focused on getting a JSON parser up and running on the Android app. It was frustrating at first, but after a few hours I was able to successfully get it working. At this point, it's still pulling placeholder data because I still don't have the Wordpress credentials I need in order to access the API. On the iOS app, I changed the JSON data that was printing to the screen to see if I was able to grab other JSON data. I didn't work on the project nearly as much as I wanted to this week due to being busy with other schoolwork. Next week I'm hoping I get the Wordpress information I need and then I can make some headway on that.

b) Max Dimm: I worked on adding buttons to change the video for past sermons as opposed to leaving the video player constantly on the live feed from livestream. I think I have a method for doing what I wanna do ready, but just haven't implemented it yet. I expect by the end of this next week I will have it done and be working on implementing the donations page for the IOS app. We met with Vee and gave him a status update, he said we were perhaps a little behind schedule but not so badly that we need to panic. We just need to keep steady forward progress at this point.

c) Kevin Stine: This week I mainly focused on continuing my development on the XMLParser section of the events page. I was able to get the basic framework completed that would parse through some sample XML data and display the names of food in the label. It took a while to get this basic functionality up, but after trying to connect with the CCB API through making an URL call, I was getting error messages saying that the data stream could not be read. It looks like I've got some debugging to do to get the URL connection to work properly.

8) Week 8:

a) Courtney Bonn: This week I finally received the information needed for the Wordpress API. I was able to enable the JSON API on WP and switch out the URL I was pulling JSON data from for the correct bulletin page. It took a little bit of finessing to figure out how to display the correct data from the JSON data though. Eventually, I was able to pull just the content and display it on the iOS app. However, I figured out that the CSS styling doesn't come over and the data displayed all of the HTML tags. After some further research, I found that if you display the data in a UIWebView, you can keep the CSS styling. So I was able to still parse the JSON data natively, rather than just loading the whole page in a web view, but I was also able to load the content I needed in a Web View so that way I can style it correctly. Currently, I still need to work on the CSS styling a little. It doesn't look exactly how it does on the church's website so I do want to put more work into and make it more consistent. My next step is also to work on the Android portion and get the bulletin page displayed on that application as well.

b) Max Dimm: I figured out and implemented the buttons that I wanted to last week. I just need to repeat the process a few times for each of the buttons but thats just a matter of doing it. I'm now researching a method for implementing the donations page. I think I have exactly how I wanna do it nailed down, I just gotta give it a try. I wanna just display a html view of the existing donations page, as thats what our client thought would be best. That way we don't have to mess with their private banking info and risk having a security dilemma be on our hands.

c) Kevin Stine: This week I continued to work on the XMLParser for the events page. As I was getting an error message saying that the data stream could not be read from the XMLParser, I did a lot of digging online and found relatively few results. I tried doing everything that the posts online mentioned however I still was not able to connect to the URL. I tried connecting with a test URL and was able to connect with no issues. I was able to narrow down the issue to the way that our username/password is setup for accessing the API. The username we're using is bonnc@oregonstate.edu and the URL connection call I was making was: https://bonnc@oregonstate.edu:password@url-to-CCB.com I was able to determine that the @ is probably what's causing the issues, since when I try doing the same thing using CURL, I'm given some information from an Oregon State page. We let the client know that we need to change the username and we should be able to connect without issues after that.

9) Week 9:

a) Courtney Bonn: This week I made a lot of progress on the Android app. I finished the bulletin page after receiving the correct information from the Wordpress site. I was able to have the raw HTML code displayed on the app. Obviously, this isn't exactly what we're wanting to display. After some research, I discovered that I needed to load the JSON data into a WebView and then the CSS styling would be kept in tact as well as the necessary html tags (¡
p¿, ¡
strong¿, links, etc.). Once I correctly loaded the data into a WebView, the data then printed correctly onto the app. The links are currently being printed in a bold blue, which at some point I would like to change. But that is just polishing the functionality of this page is completed.

After I finished the bulletin page, I decided to begin working on the donations page. Our plan for this page was to just load the entire page into a WebView so that way our app wasn't dealing with any of the security factors such as storing credit card information. While this was relatively easy, I figured out by loading the whole page, it would also load the menu, the header, and the footer of the Giving page. Since we already have a header on our app, we definitely don't want two. I went back to researching, and discover if I injected javascript into the code, I'd be able to remove certain elements of the HTML. I used this to remove the header (which removed the menu) and the footer and now just the content loads.

Right now I have a few things I'm struggling with that I haven't found the answer for. 1) The bulletin page loads very slowly. The header image and the title of the page loads immediately, but there's a slight delay while the JSON is parsing before the actual content is loaded. The functionality is correct though, so this will be something I focus on a little later. 2) When removing the header and footer from the donation page, the entire donation page loads for a split second and then reloads without the header and footer. In other words, the user can see the header for just a second while it reloads without it. I am not sure how to handle this but again, the functionality is there so this isn't something I'm super concerned about. 3) I'm worried about finishing our Android app. The iOS app is much further along. Currently, only the bulletin and donation page is finished–the Events page and the Message page aren't done yet. Because we're only a couple weeks away from the end of the term, I'm nervous that we won't be able to finish this app.

b) Max Dimm: After seeing how calvary updated their website, I decided I didn't like my implementation of the sermons page and went about changing how the user would view past sermons. I went with creating a link that would open the users default browser over our app that displays Calvary's most recent videos and content on livestream. The reason I went with this is because it was near impossible to keep the links for past weeks updated without updating and relaunching the app every week. This wouldn't work because the app would need to be updated on the weekly in order to display the correct videos for the past weeks.

I need to start making moves towards implementing the sermons page on the android app. I haven't been contributing in that area at all which is not good. Worst case scenario I would like to have my IOS portion done or basically done by spring break so that I can put that behind me and work on the android app full time spring term.

c) Kevin Stine: After finding out the issue with the username for our XML parser, we contacted our client to see if she could update our username. It took a few days to get the username updated, and when I tried connecting I was getting a different error message saying invalid credentials. It looks like the client may have updated the username, however they didn't update the admin once which prevented me from connecting. Once we get the new updated admin username I should be able to finish up the iOS events page.

10) Week 10:

a) Courtney Bonn: Because of other final projects due before dead week, I wasn't able to work at all on the capstone project. Where the app stands now is a 75% finished iOS app and a 50% finished Android app. The iOS app has almost every page at full functionality, though it will still need work aesthetically. The Android app has two pages at full functionality– the bulletin page and the donations page. The events and the messages still haven't been implemented at this point. The welcome/home page on both apps are set up but could still use some personalization from the church. Because this isn't a matter of functioning, this will be part of the "polishing" stage next term. In the next week I will be working on my final report for the term, as well as the presentation as a group. Before the end of this term or early next term, the goal is to be functionally done with both apps and then we can focus on the aesthetics's and user testing.

b) Max Dimm: I fixed the sermons page to be as I want it to be. All it needs is a visual touch-up. Next I'm looking at the donations page. I have implemented it similar to how it is done on the android app but need to figure out how to strip the header and footer on swift like Courtney did. I was having trouble finding resources that shed any light on if that is even possible. However, at a very minimal level, the donations page is functional. It just has more options or into on the page then I'd like. This week has been busy as its leading up and into finals week as I have projects to finish and finals to study for so time has been in a crunch.

c) Kevin Stine: With the discovery of the issue on the credentials we were using, I was able to finally fix the issue once we got our new username. With that complete I was able to move over my code from a test app that I was using into the master branch. Since the app I was testing on was built slightly different than our main app, there are a few bugs that will have to be squashed before I can get the events page fully functioning. Once the bugs are taken care of the events page should be almost 100% complete.

C. Spring 2017

1) Week 1:

a) Courtney Bonn: This week is our first week back after Spring break. I didn't work on the project during Spring break. This week we met with Vee and gave him the summary of where we are in the project. Because my pages are functionally complete, I'm now focusing on researching how to improve my pages. The first thing I started to research was how to strip the header and footer from the UIWebView donation page in order to help Max out. Like him, I was struggling to find a solution. The closest I could find was embedding javascript in an enablejavascript function using the WKWebView. I gave the info to Max to continue researching. Next I moved on to my bulletin page on the iOS device. I need to deal with links within the bulletin. I've researched adding a back button and also having the links open in safari. So far I haven't made a decision, but I'm going to continue researching and will then begin implementing the solution I find. Once I decide on a solution, I will move to the Android app and deal with links at that app.

b) Max Dimm: Being the first week of spring term I know I got to get going on the android portion of the app. I was having trouble doing git pulls on the command line so I went and downloaded gitkraken which has helped a bunch. I got the app up and running on android studio and stopped for the most part at this point. I began doing research into how I wanted to implement the sermons page (similar to the apple app) and wanted to see if a similar method would be possible on android studio. I have not done much coding in java before so this is slightly uncharted territory for me but I think it is entirely doable.

c) Kevin Stine: Continued working on the events page now that we have the proper username which is actually working so I'm able to connect to the database and parse the XML. I was able to get the correct data parsed from the XML however I ran into some issues when trying to pass the variables that got saved from the XML to the actual UI labels in the app. It looks like the data gets parsed correctly however I'll need to do more research on getting the labels updated.

2) Week 2:

a) Courtney Bonn: This week I made progress both on the iOS and Android app. On the iOS app I added a function that will force links on the Bulletin page to open in Safari, rather than in the app itself. I also helped Max out on the donation page and figured out how to remove the header and the footer. I've already done this on the Android page and the code was very similar, but I had to work out how to convert it to Swift. Also on the bulletin page I was able to update the CSS to match (as closely as I could) the stylesheet on the corresponding website page. The font is slightly different because Calvary is using a custom font, but it's very close. I also fixed the fonts/color of the headers on each page to match the website. Additionally, I helped Kevin out on the Events page. He wasn't able to get the events printing on the app, just in the console so I debugged and worked through the code until I was able to get it working. The event name and the date are now printing to the app in a Table View.

On the Android app I got the base files ready for the Events page, which Kevin will take from there. I found a good resource for an XML Parser and got the layout coded as well. This next week I am waiting to hear from our client on a question about their LiveStream account. According to my research, we may be able to access the LiveStream API and with our client's secret key we could pull the most recent video (past event) to display on the app. Currently the video playing is hard coded. I've contacted the client and she is checking with the web developer.

b) Max Dimm: This week I don't have a ton to report on because I did not get a ton done. I was able to find a guide online that was exactly what I was looking for. I had a lot of school work I had put off and ended up not spending a ton of time on the app. I began working on implementing the steps I found in the guide but did not finish. The app was not building at the time which was concerning but I was confident in my ability to get it to work without too much struggle.

c) Kevin Stine: This week Courtney was able to help me out with the events page and she was able to get the events to display on the events page and parse through the different dates. Now that we're able to pull in one specific week, we need to update the way that we parse the URL so we can accept more dates than just that week. This will require a bit more research on the CCB API to figure out how to specify an end date for the API and also figure out how to specify that end date based on the current days date.

3) Week 3:

a) Courtney Bonn: This week began cleaning up some of the code for my bulletin page. This included fixing the CSS and making it match the current website. I added the CSS code to both the iOS and the Android page, meaning these pages are completely finished on both apps. I also fixed the icons that were in the navigation drawer on the android page–they are now matching the icons we have in the iOS app. I then removed unnecessary files/xml layouts that were in the Android app but we weren't using. I set up a meeting with our client for April 27 to discuss the project and where it's currently at. I asked our client to see if we could get a LiveStream API key, but they're having issues with this so we may or may not be able to complete this.

In the next week I am going to help work on the XML Parser on Android to get that finished. Because of the May 1 deadline, I'm going to do my best to work a lot on the project to get it completed and ready to test. Testing will more than likely take place after the May 1 deadline but before expo.

b) Max Dimm: This week I was able to get the sermons page up and working. The functionality is there but I'm currently having issues with getting the video to fit in the window I have created for it. I spent a lot of time working on resizing the video and the window but no matter what I do it seems there will always be a scrolling option which I would like to avoid. From this point I am looking to fix that issue and have a button implemented that will give users the ability to watch past sermons. From there I can start helping fix other small issues the app may be having.

c) Kevin Stine: This week I was able to implement the logic which allowed me to set an end date to the url based on the current days date. I'm now able to display more than just the current weeks events but up to a month which is as far out as we thought would be necessary. From here I'll just need to figure out the events details page and pass the data for that particular event to the next viewController using a segue.

4) Week 4:

a) Courtney Bonn: This week we worked on trying to get all of the little details finished before the code freeze on May 1. Because we were still having issues finishing some of the bigger pieces, I did feel a little behind before this week started. This week I focused on the Events page on the Android app. Originally this was assigned to Kevin, but he was debugging and having issues with this page on the iOS app and since I had already finished my pages, I took on the android page. I implemented an XML Parser and was able to successfully pull over the events. Now when you click on an event, you can get more information about that specific event. I also made it so it would automatically pull one month of events from the current date. I also discovered that the back button on the Android app didn't work properly – it would actually exist the app. After some research, I determined that I needed to add the fragments to the back stack so that way when the back button was pressed, the fragment would know where to go. Once that was working properly, the back button no longer exited the app. At this point the android app was mostly complete except for a few little things. I was able to move on to working on the details. I was finally able to fix the constraints on the messages page on the iOS app. I ended up needing to delete almost everything on that layout and reading the components in order for the logo to show up in the correct position.

I also added activity indicators on the bulletin and donation page (iOS app only) that will show while the page is loading.

After meeting with our client on Tuesday, we discussed some additional features that the church would like if we are able to add them as well as some changes to the existing interface. The first one was removing the extra sections on the donation page. Currently the entire history on where the money is going was on the page and our client decided that they just want the ability to donate and users can visit the website for more information. The second was adding a date selection to the events page – giving the users the ability to change the month of events. I was able to complete this on the Android app. A user can now press ”previous” or ”next” to change the month of dates. There is no selection for the specific day because the client just wanted them to change the month.

The other requests our client had were seeing about the possibility of push notifications and adding forms for event registration. As these were not on our original list of requirements, we decided that we could look into this after the code freeze but focus on the requirements until then.

This upcoming week (at least before May 1) I have a few goals in mind:

1) Try to get the bulletin page to load faster. I am not sure if this is possible. I've been trying to figure out if I can ”call” the bulletin page from the home page so that way when the app starts up it automatically calls the bulletin page and starts loading it but so far I have been able to implement it. This could just be dependent on the users internet as I don't believe the JSON can parse/display things any faster.
2) Lock the orientation on both apps. The user shouldn't be able to go into landscape mode. Neither app was designed for this and when it does go into landscape, certain features don't work right.
3) Assist Kevin with the event details on the iOS app. It's currently crashing the app and I have some time to help so I'll see if I'm able to figure out why it isn't working.
4) Figure out why the back button isn't working on the event details in Android. Currently when you press back it doesn't return you to the calendar you were viewing before, it reloads the page and goes to the current date.

b) Max Dimm: Ok, this was a big week and a lot was done on my end and in general on the app (well duh its the week of the code freeze). I got the sermons page fixed so that the scrolling issue is no longer a thing which was a big relief. I was able to implement the button to watch the past sermons fairly smoothly which was very pleasing as it works just as I wanted it to. Now its on to keep working on bug fixes wherever we see them existing. I've fixed the scrolling on the donations page (similar process to fixing the video scrolling) and added readme pages to both the android and iOS folders. Other QOL things i'd like to fix are adding loading wheels to some of the pages if possible and helping make sure the app stays locked upright.

We met with our client for the first time in a long time and we got some good feedback on the app. However they wanted to add a few functions to the app which we said we would give an attempt but couldn't make any promises. This was due to getting the word on them 3 days before the code freeze. Had they let us know about this functionality at least a few weeks earlier it would have been more feasible.

c) Kevin Stine: With the events page now working for up to a month out, we focused on getting the details page working. Courtney helped debug some of the issues I was having where the variables would not actually be passed through the segue to the next view controller. She was able to get that implemented so now the details page displays everything from start time to leader phone number.

This week we also met with our client for the first time in a while and got some feedback. They wanted to add a few

more features which we said we would look into but for the sake of getting everything done for the code freeze we would mainly be focusing on the parts of the app that we specified in the requirements document. One feature that they wanted implemented was a date picker which would allow the user to specify a month or a year that way they would be able to look into the future (or past if they wanted) and view all of the events. Initially this got setup using a pickerView, however it was a little clunky and required us to hardcode the years. I was able to switch this over to a DatePicker which now allows the user to select the month and year in which they would like to view. From here until the code freeze it's just fixing bugs and making everything is functioning properly and looks good.

5) Week 5:

a) Courtney Bonn: This week I worked on getting the event details printing on the iOS app. Once I got some details printing on the new view controller, I was able to parse additional XML information from the event and print those details as well. I also added back and forward arrow pictures on the Android events page. I then worked on implementing date selection on the iOS events page. After I finished, we decided to go a different, simpler route and Kevin ended up redoing the date selection. After that the bulk of the code was finished and we just had a few finishing touches before the code freeze. I updated the welcome message per the client and removed unnecessary code. On the bulletin page on the iOS app, I added a pull to refresh function as well as removed a grey background that showed when pulling the web view down. I also received a new photo to put on the home page from the client.

b) Max Dimm: This week I did a few usability changes. Small things like changing the messages button on the IOS app and removing a text field that was not intended to be there. I changed how the button looked on IOS so that the users would be more able to identify it as a button. Outside of this there was some research I did into push notifications but I did not delve too deep into that. I went and added readme's to both the android and IOS folders so that anyone looking at our code would know how to run and compile our apps.

c) Kevin Stine: This week I did some more work on the Events page. Courtney had implemented a way for the user to pick a date, however it required our years to be hardcoded which would require the future developers to go in and manually update the app. I began working on a different implementation which utilized the built in DatePicker with a toolbar for selecting the month and year. While the date picker also includes a specific day, our client didn't want the users to be able to specify the day. So now when you select a month and year, it will default to automatically loading that entire month's dates, rather than starting with the particular date the user specified. Since this was an addition that our client wanted to add a week prior, we implemented it as best we could and if they do want to allow for specific date selections in the future it'll be pretty easy to implement. I also went in and cleaned up a lot of the Events page that had debugging statements or old code that we were no longer utilizing.

6) Week 6:

a) Courtney Bonn: Now that development is essentially finished, I took a step back from this project this week to focus on other classes that I had put on the back burner. The next step for this project was to begin work on the midterm progress report and presentation. I got our report set up and began working on the overview sections for Fall and Winter terms as well as the future work section. In the future work section I described the additional content the client may add after we had the project over. In the next week I will work on getting the progress report finished and focus on preparing for expo.

b) Max Dimm: Truth be told, not a ton of focus was given to capstone this week. I figured we had finished development so It was bumped down a notch in my priority lists. We began working on the midterm progress report but I think I will work on it further next week as I am heading back to Portland for the weekend. I need to handle the writeup for my pages and record voice for my part of the slides but that should not take too long.

c) Kevin Stine: With all our major development completed (everything on our requirements document), I used this week to focus on other classes that I had put off since we had a big push to finish everything for this class. Next is getting our midterm progress report and presentation completed which I'll probably work on this coming weekend. Other than that just preparing for expo is all that I have for this week.

7) Week 7:

a) Courtney Bonn: Now that we have basically finished our project and expo is now over, we are at the point where we are getting ready to transition the ownership of our project to our client. Our client has indicated they are going to tweak and continue developing the apps for a little while before publishing them.

If I were to redo the project starting Fall term, I would tell myself to spend more time researching mobile development in the early months, rather than waiting until Winter term. This would have allowed me to begin developing more confidently earlier on and could have given us more time to add on features or do other cools things with the apps that we ended up not having time for.

I'd say the biggest skill I'm taking from this project is how to work with a team and a client over a long period of time. Before this class, the longest I'd have to work with people is a few weeks–a term at most and then I'd be done. Learning to cooperate and help each other while dealing with requirements and requests and meetings with the client was a great way to prepare myself for future projects I may work on at a job.

My favorite part about the project was learning how to develop apps in general. This has always been my main interest and now I have the actual experience to go along with it. The only part of the project I would say I didn't like is not being able to actually publish the apps. Since the client wants to continue working on them, we can't truly say the project was "complete" since they aren't published yet.

I learned a lot from my teammates but the biggest thing was just how to work in a team. Like I said earlier, I wasn't used to having to work with the same people for such a long period of time. But in this type of project, you have to learn to work with each other and find a good balance. I think we found a great balance between us. We each had our strengths and weaknesses and we could use those to help each other out.

I think I would be satisfied if I was the client. While there is an issue with one of the pages still, that problem has been adequately explained to the client and until they upgrade their system, there isn't anything we can do about it. Because we know about this issue and know how to fix it (we just can't control that part), I am satisfied with the project as is.

If the project were continued, I think the messages page would need worked on more. Once their system is upgrade, I think it'd be great to pull the last 5 or so events that were on the livestream page and have them all available on the app, rather than just the most recent. I also think adding notifications and the ability to register for events within the app would be cool features to work on.

Overall, I'm really pleased with our product and I'm proud at how much we were able to accomplish.

b) Max Dimm: I am so glad we are done with expo, we just need to stay focused for a few more weeks and we're done. Looking back on this project, it was a lot of work, very frustrating, and I feel like I learned a ton. If I had the ability

to go back and give myself advice back in fall term I would tell myself to just download Xcode and android studio and spend 2 hours a week just trying to implement a few things. I think this in and of itself would have taught me a ton and would have made the process of development much more smooth winter term.

In regards to the biggest skill i've learned, I would say it is version control. I had not used git (as it was intended) before. Seeing how it can actually save us when we make mistakes is amazing. Also, being able to coordinate code between three people without having to re-do things or overwrite code is awesome. This is definitely something that will come in handy down the road. Also, I feel like knowing the foundations of swift and java will be super helpful down the road and could easily be applied to future jobs.

In regards to my enjoyment of the project, I liked being able to see the results of my code so quickly and to port it onto my phone. This gave the project a very real feel to it as opposed to other projects I've done. The IOS app did not have the same feel to me because the simulator on my laptop was so poor. I enjoyed being able to show my friends the app we were working on and being able to show them the differences over time. In regards to things I did not enjoy about this project, I did not enjoy not having many solid resources to use. At times it felt like if the internet did not yield the answers I needed, that I was screwed because I didn't know who to ask on campus.

I learned a lot from my teammates throughout this. I think the biggest thing though was communication. We used slack to stay in contact with one another and I felt, at least, that our communication was fairly spot on. We let each other know what we were working on and were able to answer one anothers questions fairly easily. I think our group worked very well together and we ended up with a solid product.

I think I would be. Not because the end result was what we ended up wanting, but because the end result was what we asked for initially. Its hard to judge a group based on results when the desired result changed so much throughout the project and so late in the projects development life. I would have aimed to communicate better with my group if I was the client in this particular project.

I think moving forward this project may revive more focus. I know our client was interested in a few things like adding push notifications and geofencing. I think the client will need to upgrade their systems on their end to get live functionality added, but will eventually want someone to change that line of code for them. I think the design could use perhaps a tad bit of polish. I think the app in itself works great, but a few things added to it to give it more functionality would be nice.

c) Kevin Stine: With Expo done it feels like a pretty big weight has been lifted off of my shoulders. Now that our app is fully functioning and working properly as per our requirements, we've been able to relax a bit more and take it a little easier. While our client still has a few extra things that they would like to see implemented that they requested a few weeks ago, the main functionality is done so at this point anything else we add would be during our free time. This week there wasn't much to do besides Expo, which I thought went really well and it was fun to see everyone's projects after a long year.

If I were to redo the project from fall term, I would probably have started learning Swift and Java sooner, probably during the middle of fall term. While we had a lot of documents and things to write, I think having a better understanding of how Swift worked, getting familiar with the newer version (3.0 vs older iterations) would have been helpful. It seemed like once I actually got to developing for iOS, a lot of the issues I ran into were due to following guides that were based on an older iteration of Swift. I think reading through the documentation earlier and really reading into the different functions that we would be trying to implement would have been beneficial. Once we got into Winter term I spent a lot of time researching

the Date class in Swift and utilizing that versus NSDate, Calendar or NSCalendar. It seems like if I had focused on one particular implementation it would have made development a lot smoother.

I think the biggest skill I've learned would be problem solving. We were essentially given a problem with no guidelines besides what they wanted to accomplish. From there it was up to us to determine exactly what it was that the client was requesting, turn that into requirements and develop an application that would fit those requirements. Essentially the entire process was one big long problem solving exercise. I think it really helped develop my long-term problem solving skills. Up to this point most projects I've worked on had been at the most, a two week ordeal. However when you think in the context of months rather than weeks, things start to get more complicated and I think I was really able to hone my long term planning skills. I see utilizing those same problem solving skills in the future as I apply them in the workplace.

I really liked being able to develop a mobile application as it has always been something I've been interested in. I think it was really cool to take a concept and turn that into an actual product which could potentially be on the marketplace. I think for me, iOS development was a lot more interesting since it seemed like Xcode was a lot more polished than Android Studio. I think the portion that I liked least about this project was debugging and trying to figure out issues and bugs that I wasn't able to find an easy workaround or solution online for. It took a long time for me to get the events page working due to a lot of those types of issues.

I learned how to communicate and work well with a team. Through using Slack we were able to keep everyone updated and communicate freely throughout the three terms. I think this really helped us as we didn't have to meet in person too often (since that would have been inconvenient as we live in three different cities), and didn't have crazy long email threads that got convoluted and messy.

If I were the client I would be satisfied with the work that has been done on the project. Based on the initial requirements and feature requests, everything is implemented at a functional level. While some of the new features and requests aren't on there since it was requested last minute, I think that apps do everything that the requirements specified and in a manner which would be easy for the congregation to utilize.

While I don't necessarily think this project has enough potential ideas to make it a good candidate for continuation, I think it could be a bit more polished with some of the newer features that the client requested. Things like notifications (which require a developer account) and other languages would be possible to implement, but I think all of the main functionality for the app is there and at this state, a pretty good app.

VII. Final Poster

CALVARY CHAPEL CORVALLIS

Helping Calvary Chapel connect their congregation

Computer Science Capstone Project

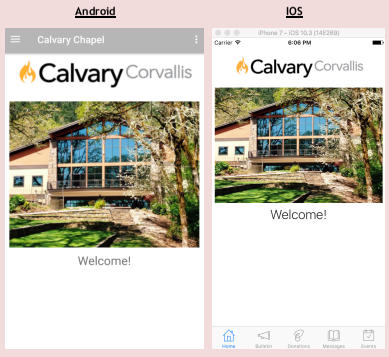
What Was the Problem?

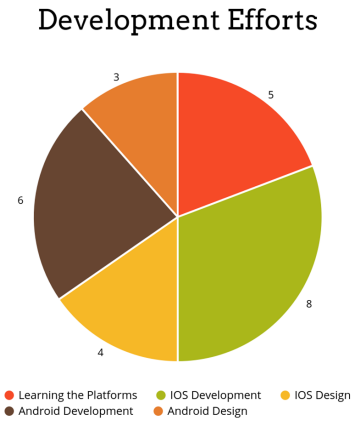
WEBSITE BUT NO APP

- Purpose: Produce an iOS/Android application for Calvary Chapel of Corvallis.
- Current website does not provide an interface where current members of the church can very quickly access important information such as events, bulletins, and messages from the service.
- The issue is that the church believes that people do not have the time or the knowledge to open up the website to access the information they need.
- Printing out the bulletins every week to hand out at services can be expensive and ineffective.
  - The desired application will be simple enough for anyone to use while providing back end access for staff to easily upload new information to the app.
- The priorities lie in maximizing the usability of the app and providing bulletin, schedule, video, and donation.

OUR SOLUTION

- To create an iOS/Android app that will provide the most pertinent information for the churches core of membership.
- Application will include an E-bulletin, the church schedule, the ability to watch past sermons, and the ability to donate.
- The end goal is a highly usable product that even the least tech savvy individuals will be able to easily navigate.
- Incorporate data from the churches database/website for minimal outside maintenance.





PROJECT DESCRIPTION

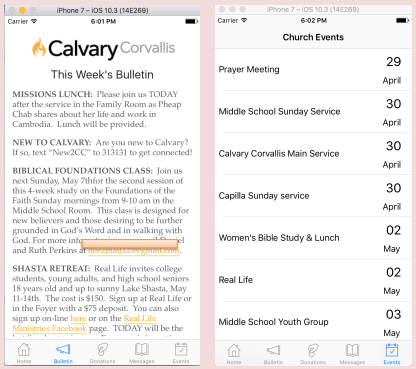
In order to deliver a product we knew our clients would be satisfied with, we made the difficult decision to create two native applications. Our other option was to create one application that would work as a cross-product; however, we decided that not only would it be beneficial to the client to have both native applications, it would benefit our education as well. After thoroughly researching mobile development, we immediately began producing the iOS application. After we created approximately half of the iOS app, we began working on the Android app concurrently. Because of time constraints, we made the decision to work on both apps at the simultaneously, rather than finishing one and moving on to the other.

Both applications were divided between the three of us in the following way:

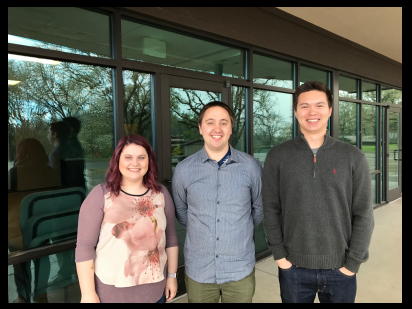
- Courtney was assigned the bulletin page and the home page on both platforms, and the donations page and events page on Android.
- Max was assigned the messages page and the donation page on iOS.
- Kevin was assigned the events page on iOS.

SOLUTION IMPLEMENTED

- Two native applications for both iOS and Android
- The pages on each application pull information directly from the corresponding website page
  - Bulletin: the Bulletin page on the website is sent through a JSON parser and the response is displayed on the app through a WebView. Using a WebView allowed us to use CSS styling to format the JSON response similarly to how the website is styled. This creates consistency between the website and the apps.
  - Donation: the donation page is loaded through a WebView. This decision was made in order to protect the security of any person using the app. The app will load the existing donation page from the website and will take advantage of the security measures already in place so that members can feel confident providing their credit card information.
  - Messages: The messages page loads a mini video player using a windowed WebView and using Livestream's live embed link. Buttons for past sermons links to a webpage listing all the pervious sermons for the user to search from. The Live video will auto play the most recent or any live services.
  - Events: The events page uses a URL connection to interface with the CCB public calendar API. The connection returns an XML file which is then parsed through to return any pertinent information such as event name, date, time, etc. The parsed data is then passed to the various corresponding labels to display that information to the user.



Meet the Team and Our Client



THE TEAM

- Courtney Bonn: *bonnc@oregonstate.edu*
- Maxwell Dimm: *dimmm@oregonstate.edu*
- Kevin Stine: *stinek@oregonstate.edu*

OUR CLIENT

- Desiree Gorham: *Administrative Assistant*
- Ryan Smith: *Social media/Design interface*
- Reuben Wai: *A/V Media Interface*
- Ryan Gardner: *Calvary Website Interface*

Calvary Corvallis started at a small church in 1995, led by Senior Pastor Rob Verdeyen. It began with a small group of 15-20 people and has grown into a congregation of over 800 attenders. Where it sits now on 53 acres in Corvallis has been its permanent home since 2007. A college ministry was formed and the church became a spiritual home to many Oregon State University students.

Communication with its members is very important to the staff at Calvary Corvallis. Their main motivation for wanting to create mobile applications is to improve communication between the church and all those who attend. The mobile applications are intended to communicate important messages, events, and bulletin news in a concise manner. The web development team will continue to improve and add features to the applications to make them even more desirable to their audience.

Oregon State UNIVERSITY

VIII. Project Documentation

Our project works in two different ways, depending on it is begin ran as a developer or as a consumer. Because the end result is an application, available on both iOS and Android platforms, the project's structure is very simple. There are two sets of project files, one for each platform, and within the files the code that can be executed in order to run the application is available.

As a consumer, one can download the application from the app store once it has been published. After downloading, it will be available to use on the corresponding phone. The app itself takes up less than 40 MB of space on a phone.

As a developer, the applications can be ran on a computer using two different softwares: Xcode and Android Studio. These softwares are required in order to run the code. They can be downloaded from https://developer.apple.com/xcode/ or https://developer.android.com/studio/index.html. One caveat, though, is that to be eligible to download Xcode, one must be on a Mac computer. Android Studio can be downloaded from both Mac and Windows computers.

Once the required software is installed, the code can be downloaded from https://github.com/ikaikastine/capstone-group-62 with the code living in the folders iOS and Android. The projects can then be opened in the corresponding software and ran accordingly. The software allows one to run the code on either a built in simulator or an actual device if it is connected to the computer.

IX. Learning New Technology

Because mobile development was new to all of us, we had to spend a good portion of our time researching and becoming accustomed to using the mobile development software, Xcode and Android Studio. The main sources which we used to learn these platforms, were produced by Apple and Android Studio. Apple had a website in which it walked us through how to begin developing an iOS App on Xcode. It taught us how to develop in Swift as well as how to use the program [1]. Android Studio had a similar website, as well as sample apps we could walk through designing [2].

Besides learning the main platforms, we had to do a copious amount of research on how to implement each part of the app. A website that was instrumental in learning how to parse the Wordpress pages into JSON objects and display the JSON response was Grok Swift [3]. Because we also needed to know how to handle JSON in Android, a thread on Stackoverflow was very helpful [4].

A tutorial on www.androidbegin.com was used to understand how to handle XML parsing and displaying the data on an Android app [5]. We needed this information to handle the events calendar through CCB.

While there were other websites that we searched and read, the above websites were the most helpful in learning the new technology. As far as print documentation, we did not utilize any reference books in our learning. Our assigned teacher's assistant, Vedanth, did provide some suggestions to help us with our learning, as he had some experience in mobile development from his own senior project.

For the iOS Events page, a lot of time was spent looking at the Apple Developer API reference at developer.apple.com/reference to get familiar with the references for various portions of the application. For the Events page specifically there was a lot of time spent looking at the DateFormatter reference, the NSDate reference, and UIDatePicker reference. In addition to utilizing the developer API reference, Stack Overflow at stackoverflow.com was pretty critical in helping with debugging the code. I found a lot of discrepancies between Swift 3.0 and older versions, so when my code had a bug, it was tough to find specific help related to that version of swift. However I think that stackoverflow was instrumental in my success in debugging and with adding certain features like the UIDatePicker that utilized a particular type of object with the toolbar.

X. Team Reflection

A. Courtney Bonn

Because I had not worked with mobile development before I decided to choose this project, I was given the opportunity to learn an abundance of technical information. I went into this project with little understanding as to how an app worked and how to create my own. Fast forward to the end of the year and I know feel comfortable developing in both iOS and Android platforms, a skillset I am excited to get to add to my resume. Specifically, I learned how to code in Swift and brushed up on my Java language skills. I learned two new platforms to code in, Xcode and Android Studio. I also learned how to work with a REST API and JSON objects, as well as XML parsing and responses. Finally, I learned the whole process of creating an app from start to finish.

The main non-technical bit of information that I learned this year was how to produce accurate and helpful documentation for a large project. While documentation can feel time-consuming, it ends up saving a lot of time because there is already a roadmap for the project. Beginning development with no documentation would have felt overwhelming and chaotic and more likely than not would have caused development to take at least twice the amount of time. Knowing how to produce the necessary documentation is something I am very grateful to have learned in this course.

Until this class, most assignments I had worked on were weekly assignments, with a few term projects here and there. Even then, term projects did not usually take up the entire term, but perhaps the last couple of weeks. I have never experienced working on the same project for a year before. Learning to keep focus and interest in a project for this amount of time, was a valuable skill that I am taking away from this project.

The biggest lesson I learned about project management, was that it is very easy to fall behind in a project if you go even one week without working on it. There were times where I could not work on the project for whatever reason, and then I felt I was behind in where I should be in project progression. The lesson I will take away is that even putting an hour into the project every week or a few minutes a day will help keep the project going smoothly.

In my college career, I have worked in many group projects. More often than not, these projects lasted a few weeks to a term at most, and then we went our separate ways. To work with the same group for almost an entire year was a new experience for me and one that I enjoyed. I learned that you have to help one another because you have to work as a time–you cannot leave anyone behind. I also learned that you have to be patient with each other because not everyone is going to be able to commit the same amount of time or effort at all times. Because we all had different classes to focus on as well, we had to work together to make sure our project was not falling behind if we needed to work on other classes.

B. Max Dimm

C. Kevin Stine

There were a few reasons why I choose to partake in this project for senior design. Firstly, I really wanted to get some hands on experience with mobile development as that was an area of programming and design I had always wanted to explore, but hadn't really had formal instruction on how to do so. I decided that this could be a great opportunity to learn the fundamentals and basics of mobile development to add to my portfolio. Second, this project intrigued me due to the fact that Calvary Corvallis was using Church Community Builder, and the local church plant that I'm a part of also uses that particular backend software. I thought this would be the perfect opportunity to learn how make a mobile app that integrates

with CCB in the hopes that one day I could create a similar type of application for my church. This project was a great opportunity for me to learn more about mobile development, and really learn the intricacies of Swift as well as some Java.

I learned a lot of technical information through this project. I learned how to use Xcode more in depth as well as Android Studio as the platforms in which we developed for. Surprisingly I spent a lot of time with Xcode just learning how to use the inspector to change the various attributes of variables and of elements on the storyboard. While I didn't end up doing too much Android development, I did still learn how to use Android Studio, learned how to utilize the xml files that make up and Android project, and learned how to setup an Android device to use for debugging and demoing purposes. I learned a ton on Swift as I mainly focused on iOS, and really feel like I have a solid foundation now for being able to create my own applications. I learned how to utilize the storyboard, inspector, created a tabbed application, utilize the NSDate and DateFormatter references and create a UIDatePicker with a toolbar. While I obviously learned more than just those few things, I think those are the most notable technical things I learned how to use. In addition I was able to continue to enhance and refine my skills at using git as it was something we utilized throughout the entire process of the project.

On the non-technical side, I learned a lot about how to write good documentation and how to take the information, needs and ideas from our client and formulate that into specific requirements that we could use as benchmarks. With Fall term being mostly centered around writing documents, I found that it also helped enhance my writing skills while improve on my ability to analyze and think through specific processes and ideas. In addition I was able to learn a lot about time management through this project. Having deadlines for documents really helped me set easy goals, however once we got to Winter term where deadlines were more fluid, it really made me have to manage my time well and make sure that I was staying on track to reach our goals. I learned that when it comes to project work, it's really important to make sure you set your requirements and make sure that you have goals to reach throughout the entirety of the project.

In terms of project management and working with a team, I learned that communication is key to the success of the project. Utilizing tools such as Slack really helped boost our communication as a team, allowing us to make sure we were all contributing and striving towards the same goals. It also made sharing the work easier as we could just ask for help debugging or to see if someone else could take a look at the code to see if they could figure out what was going wrong. I also found that working with a team really requires everyone to put in their best effort, while we all have different things going on outside of school, it's important to keep in mind that we're on the same team and have the same goals. It also really helps to designate who will do what parts of the project so everyone has a clear idea of what is required of them. Knowing that we each have a part to play in the whole project really helped us moving forward as we knew that once we all finished our individual pieces, it would come together to make a nice application.

If I could do it all over again I would definitely do some more research on the front end. While I had a general idea of how to use Xcode and a basic idea of what the format for an iOS app was, I didn't really have a good understanding of how it all fit together. While doing research I learned about the Model View Controller and it really helped me understand more of what was going on behind the scenes. I think if I had known more technical aspects of a mobile app going in, I would have been able to better understand how the various pages linked together. I also would have started development sooner as it seemed we kind of had to rush towards the end to make sure we fixed all of our bugs before the deadline. While we did do development throughout all of Winter term, I think if we had gotten the basics and easy stuff out of the way first, rather than researching how to implement the whole XML Parser in my case, things would have gone smoother. I also would have broken down my section a bit more and mapped out how I was going to approach development. I basically

started development by going after the whole XML Parser without realizing that I would need to connect to the CCB API in order to access that information and would need to create some sort of date to keep track of whatever the current date was. I think if I had started with the Date portion I would have been able to implement the XML Parser a bit easier.

REFERENCES

[1] A. Inc. (2015, September) Start developing ios apps (swift). [Online]. Available: https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html#//apple_ref/doc/uid/TP40015214-CH2-SW1

[2] (2016) Android studio. [Online]. Available: https://developer.android.com/studio/index.html

[3] (2016) Simple rest with swift. [Online]. Available: https://grokswift.com/simple-rest-with-swift/

[4] (2016) Get json data from url using android? [Online]. Available: https://stackoverflow.com/questions/33229869/get-json-data-from-url-using-android

[5] (2013) Android xml parse images and texts tutorial. [Online]. Available: http://www.androidbegin.com/tutorial/android-xml-parse-images-and-texts-tutorial/

APPENDIX A
ESSENTIAL CODE LISTINGS

Code 1: iOS EventViewController Snippet

```
func createDatePicker() {
    // format the picker
    datePicker.datePickerMode = UIDatePickerMode.date
    //toolbar
    let toolbar = UIToolbar()
    toolbar.sizeToFit()
    // bar button item
    let doneButton = UIBarButtonItem(barButtonSystemItem:
                        .done, target: nil, action: #selector(donePressed))
    toolbar.setItems([doneButton], animated: false)
    changeDate.inputAccessoryView = toolbar
    changeDate.inputView = datePicker
}
```

This code creates the datePicker which allows the user to select the month, day and year.

Code 2: iOS EventViewController Snippet

```
func donePressed() {
    // format date
    let dateFormatter = DateFormatter()
    dateFormatter.dateFormat = "yyyy-MM-dd"
    pickerTracker = true
    startDate = dateFormatter.string(from: datePicker.date)
    updateTable()
```

```
    self.view.endEditing(true)
}
```

This code is the selected action for the DatePicker.

Code 3: Android XML Parser

```
public String getXmlFromUrl(String url) {
      OkHttpClient client = new OkHttpClient();
          final String basic = "Basic " + Base64.encodeToString(CREDENTIALS.getBytes(),
          Base64.NO_WRAP);
          String str = null;
          Request request = new Request.Builder()
                  .url(url)
                  .header("Authorization", basic)
                  .build();

          try {
              Response response = client.newCall(request).execute();
              str = response.body().string();
          } catch (IOException e) {
              e.printStackTrace();
          }

      return str;
   }
```

Code 4: iOS JSON Response

```
{
  "id": 1038,
  "date": "2016-10-27T19:22:53",
  "date_gmt": "2016-10-27T19:22:53",
  "guid": {
    "rendered": "http://www.calvarycorvallis.org/?page_id=1038"
  },
  "modified": "2017-03-18T11:48:50",
  "modified_gmt": "2017-03-18T18:48:50",
  "slug": "bulletin",
  "status": "publish",
```

```
  "type": "page",
  "link": "https://www.calvarycorvallis.org/bulletin/",
  "title": {
    "rendered": "This Week&#8217;s Bulletin"
  },
  "content": {
    "rendered": "<p>all bulletin content would be here...
    ...
    },
    Additional, unrelated JSON returned below here...
}
```

Code 5: iOS JSON Parser

```
do {
        guard let bulletin = try JSONSerialization.jsonObject(with: responseData,
        options: []) as? [String: AnyObject] else {
                print("error trying to convert data to JSON")
                return
         }

               guard let bulletinContent = bulletin["content"]?["rendered"] as?
        String else {
                  print("Could not get bulletin content from JSON")
                  return
         }
         let actualContent = bulletinContent.replacingOccurrences(of: "<[^>]*.", with:
         "", options: .regularExpression, range: nil)

         DispatchQueue.main.async{
                  self.jsontext.text = actualContent
         }
} catch  {
         print("error trying to convert data to JSON")
         return
}
```

Code 6: Android JSON Parser

```
if (response != null) {
```

```
        try {
                JSONObject jsonResponse = response.getJSONObject(TAG_CONTENT);
                 String jsonData = jsonResponse.getString(TAG_RENDERED);
                 textView.setText(jsonData);
                  Log.e("App", "Success: " + response.getString("yourJsonElement"));
        } catch (JSONException ex) {
                    Log.e("App", "Failure", ex);
        }
}
```

Code 7: iOS Load into WebView

```
let htmlCode = "<!DOCTYPE HTML><html><head><style> body {color: #5b5e5e; font-family:
'Lora', Palatino;} a { border-bottom: 1px solid #fbaf17; color: #fbaf17;
text-decoration: none; }
.staff a { border-bottom: 0px none; } a:focus, a:hover { border-bottom: 1px solid #fbaf17;
color: #b17b0e; }</style></head><body>" + bulletinContent + "</body></html>"

self.bulletinWeb.loadHTMLString(htmlCode, baseURL: nil)
```

Code 8: Android Load into WebView

```
String bulletinContent = "<!DOCTYPE HTML><html><head><style> body {color: #5b5e5e;
font-family: 'Lora', Palatino;} a { border-bottom: 1px solid #fbaf17; color: #fbaf17;
text-decoration: none; } .staff a { border-bottom: 0px none; } a:focus, a:hover {
border-bottom: 1px solid #fbaf17; color: #b17b0e; }</style></head><body>" +
jsonData + "</body></html>";

myWebView.loadDataWithBaseURL("file:///android_asset/", bulletinContent,
"text/html", "utf-8", null);
myWebView.getSettings().setAllowFileAccess(true);
```

Code 9: iOS Donation Page

```
        let donateURL = URL (string: "https://www.calvarycorvallis.org/give/")
        let requestObj = URLRequest(url: donateURL!)
        donateView.loadRequest(requestObj)
        donateView.delegate = self
        donateView.scrollView.delegate = self
        donateView.scrollView.isScrollEnabled = false
```

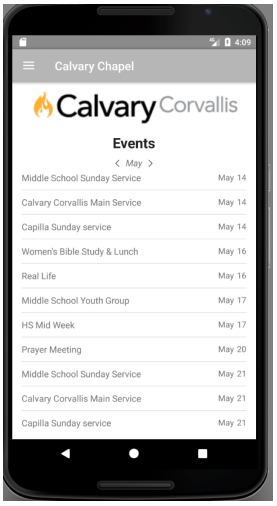
Code 10: Android Donation Page

```
@Override
    public void onPageFinished(WebView view, String url) {
            myWebView.loadUrl("javascript:(function() { " +
            "document.getElementsByClassName('site-header')[0].style.display='none'; " +
            "document.getElementsByClassName('footer-widgets')[0].style.display='none'; " +
            "document.getElementsByClassName('content')[0].style.display='none'; " + "})()");
            myWebView.setVisibility(View.VISIBLE);
            myWebView.getSettings().setLoadWithOverviewMode(true);
            myWebView.getSettings().setUseWideViewPort(true);
        }
    });
        myWebView.setVisibility(View.GONE);
        myWebView.loadUrl("https://www.calvarycorvallis.org/give/");
```

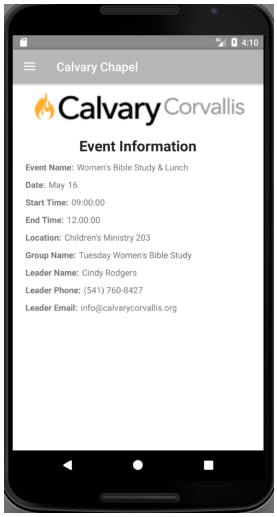
Code 11: Android Messages Page

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        myView = inflater.inflate(R.layout.fourth_layout, container, false);

        String videoLink = "<html><iframe id=\"ls_embed_1493363421\" src=\"
        https://livestream.com/accounts/18343788/events/7279945/videos/154327352/player
        ?width=960&height=540&enableInfo=false&defaultDrawer=&autoPlay=true&
        mute=false\" width=\"960\" height=\"540\" frameborder=\"0\" scrolling=\"no\"
        allowfullscreen> </iframe></html>";

        myWebView = (WebView) myView.findViewById(R.id.messagesView);

        WebSettings webSettings = myWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        myWebView.getSettings().setLoadWithOverviewMode(true);
        myWebView.getSettings().setUseWideViewPort(true);
        myWebView.getSettings().setBuiltInZoomControls(true);
        myWebView.loadData(videoLink, "text/html", "utf-8");

        return myView;
```

APPENDIX B
PHOTOS

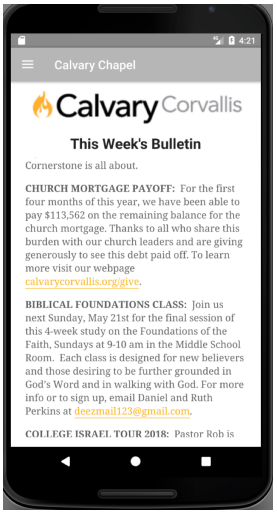


(a) A List of Events

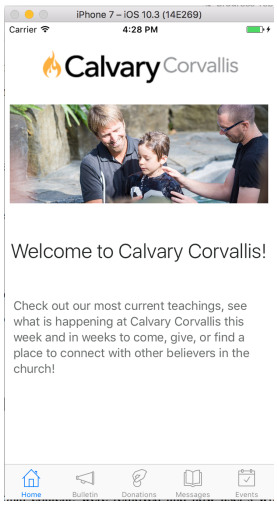


(b) An Event Details

Fig. 1: Android Event Page

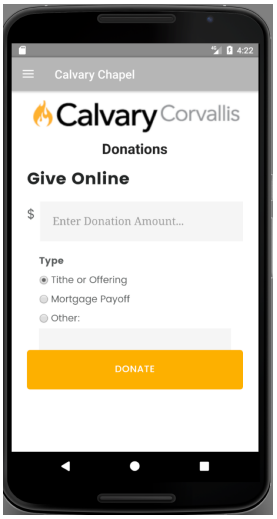


(a) Android

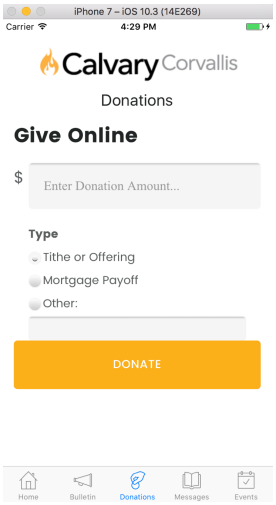


(b) iOS

Fig. 2: Bulletin Page

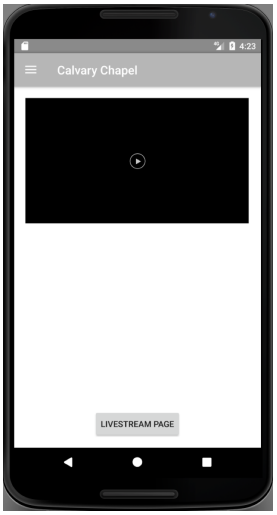


(a) Android

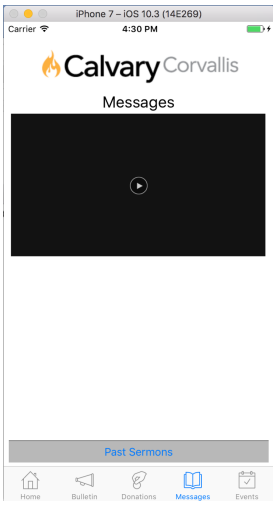


(b) iOS

Fig. 3: Donation Page



(a) Android



(b) iOS

Fig. 4: Messages Page