Calvary Chapel Corvallis

Design Document

CS 461 Fall 2016

Kevin Stine, Courtney Bonn, Maxwell Dimm

Group #62

Abstract

The purpose of this project is to produce an iOS/Android app 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 app 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.

SIGNATURES

Client	Date
Courtney Bonn	Date
Kevin Stine	Date
Maxwell Dimm	Date

CONTENTS

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.

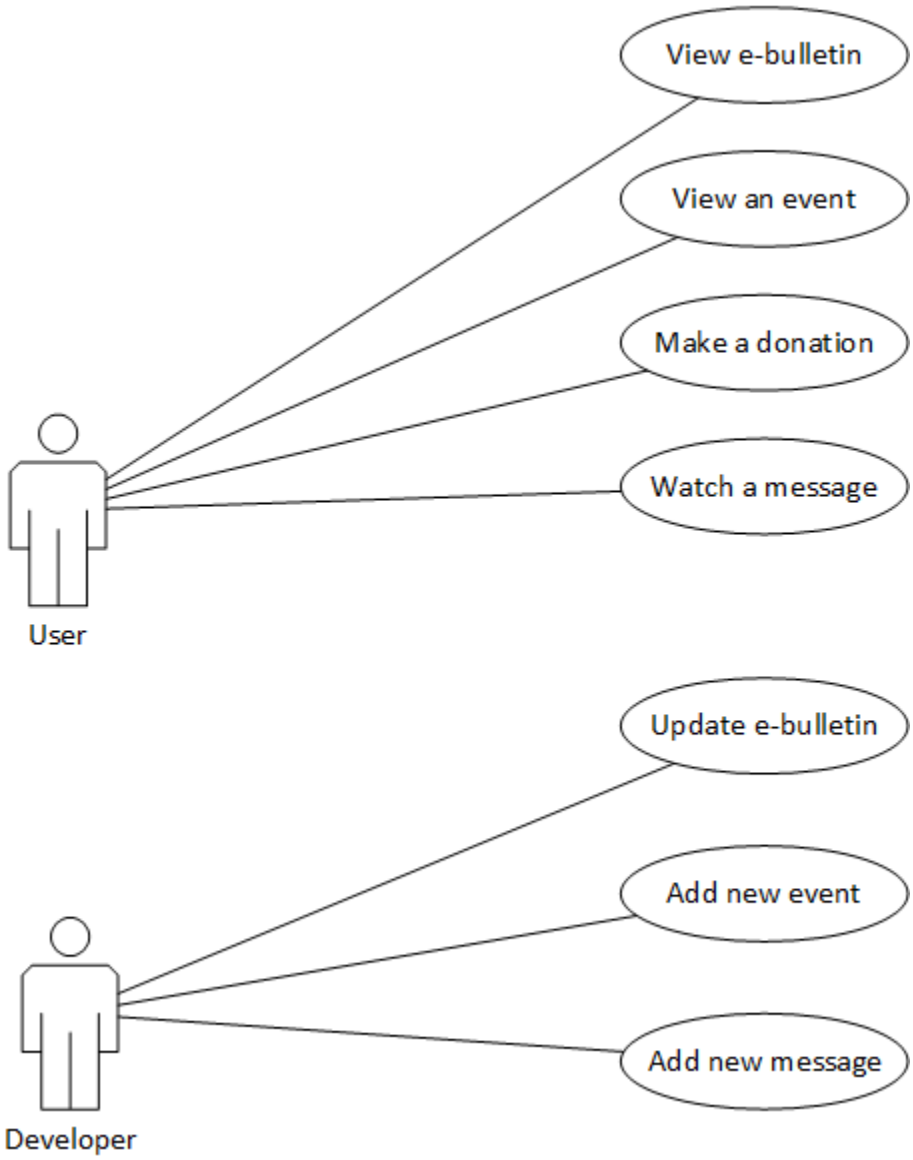

Fig. 1. Use Case Diagram

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.

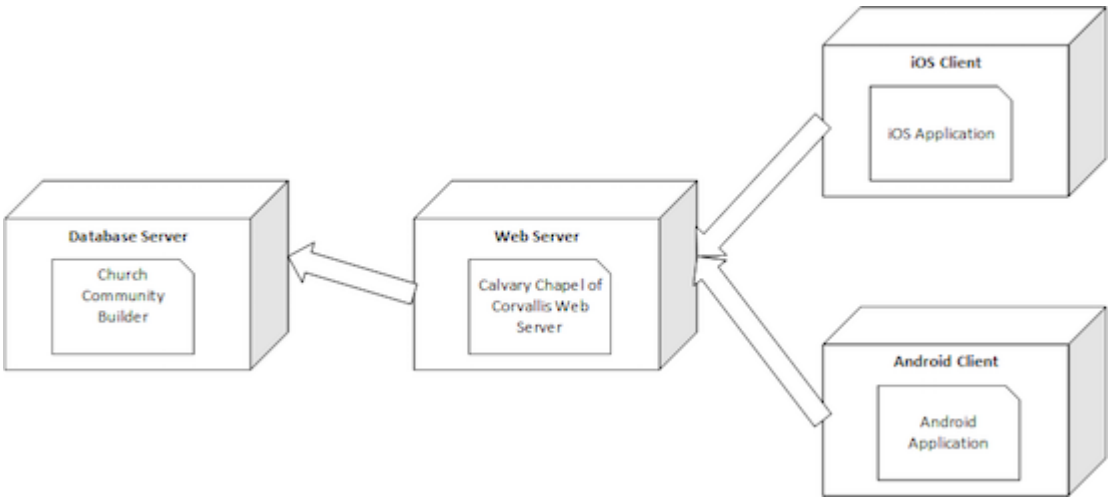

Fig. 2. Deployment Diagram

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.