**C868 – Software Capstone Project Summary**

**Task 2 – Section A**



|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | Superior Church CRM Mobile Application |
| **Student Name:** | Andrew Merzwski |

Table of Contents

[**Business Problem** 3](#_Toc43202209)

[**The Customer** 3](#_Toc43202210)

[**Business Case** 3](#_Toc43202211)

[**Fulfillment** 3](#_Toc43202212)

[**Existing Gaps** 4](#_Toc43202213)

[**SDLC Methodology** 4](#_Toc43202214)

[Requirements 4](#_Toc43202215)

[Design 4](#_Toc43202216)

[Implementation 4](#_Toc43202217)

[Verification 5](#_Toc43202218)

[Deployment and Maintenance 5](#_Toc43202219)

[**Deliverables** 5](#_Toc43202220)

[**Project Deliverables** 5](#_Toc43202221)

[**Product Deliverables** 6](#_Toc43202222)

[**Implementation** 6](#_Toc43202223)

[**Validation and Verification** 6](#_Toc43202224)

[**Environments and Costs** 7](#_Toc43202225)

[**Programming Environment** 7](#_Toc43202226)

[**Environment Costs** 7](#_Toc43202227)

[**Human Resource Requirements** 7](#_Toc43202228)

[**Project Timeline** 8](#_Toc43202229)

# **Business Problem**

**The Customer**

The customer is Superior Church, a small local community church servicing a congregation of less than 100 members. Due to an increase in the average member age over the past 10 years, they have expressed a goal of attracting new members. To achieve this goal, they have decided to implement a mobile application that will help them internally with customer relationship management (CRM) and act as an information hub for users.

## **Business Case**

Currently, the only technology in use by church staff is email, along with a handful of personal computer spreadsheets spread out across members of the congregation. Most of the customer information is stored on paper media and often changes hands due to a regular turnover rate in staff (2-year service terms). Data silos are rampant, and a good deal of member information is estimated to be lost/misplaced on a regular basis. Communications are normally sent out as a mass group email, and subgroups are formed and disbanded without any formal recordkeeping. Church staff have noted that the lack of organization has contributed to losing members in the past and deterring prospective new members.

The mobile application will, first and foremost, help centralize the accumulation of member information by storing data in a single database. Church staff will also have more complete access to all data using the application. Because the application allows for direct member-member communication through email and phone/text, quality, accuracy, and timeliness of church communications should increase. The application allows the administrative user (< 5 users) to distinguish between active members and persons that are not officially part of the church membership, so maintaining a relationship with casual visitors and prospective members will be easier.

## **Fulfillment**

The application will be developed for Android devices (version 4.3 and higher) using Java. Currently, the application will store data using the on-device SQLite database, but the church has plans in the near future to upgrade to a dedicated cloud-based server for centralized data access. With this in mind, the application will be developed with a repository abstraction layer to easily connect to an external database.

The application entry point is a login screen where users will be required to authenticate with the database in order to gain access to the rest of the app. Guest users will be able to bypass this login, but will be restricted to only viewing church events (which are public in nature and often advertised on social media). After a successful login, the user is taken to a home screen where they can reach further information regarding other church members, church groups they are involved with, their donation history and, of course, church events. This upper layer of aggregated information will be presented as a scrollable, top-down list on the screen. Admin users will be able to add, update and delete entries in all of these categories. Navigating to the screen that displays a single member’s information, which displays street address, phone number and email address, a user will be able to contact them through phone/email (using the device’s native capabilities) with a single click. Users can also search for member by name and search for event by keyword.

# **Existing Gaps**

The existing method for tracking membership and member donations/transactions is to have the member fill out a small paper slip and return to a staff member. The slips are collected in the office, and eventually given to a non-staff member to add member info to a local spreadsheet. Financial information is tallied in a physical ledger, and general donation history has been collected and categorized, but is heavily dispersed throughout decades of physical records (many of which are most likely lost to time). No staff member currently has direct, on-demand access to the member list. Event information is currently presented on the church website as well as the church’s social media page, but most staff members do not have access to update these sites. There currently exists no collected data on church subgroups.

# **SDLC Methodology**

The project will be developed using the Waterfall methodology for the following reasons:

* The church provided a thorough list of requirements in advance
* It is not expected that the requirements will change significantly during development
* The developer has past experience with similar projects, in form and size

The waterfall development method can be broken down into 6 distinct phases, described in more detail below.

## Requirements

The first phase of development, gathering requirements, focuses on what the church wants and needs the app to accomplish. As stated above, some of the main objectives of the application were summarized as: easier communication between members, better access to data, better maintenance of data, and centralized control of data. The waterfall methodology worked here because the requirements were simple and well developed.

## Design

High-level and low-level design takes place next. The application will be designed using the View Model (MVVM) pattern using the Android Room architecture, with a data repository class. As stated previously, the repository will enable the church to easily connect to another data source in the future. An entity-relationship diagram (ERD) will be developed during this stage, as well as the general methods by which the data is accessed from the user interface. The bulk of the application involves displaying simple data, however some aspects including user authentication and admin user identification will be extremely important to design well and implement correctly. Due to the personal data being stored and accessed by the application, it is imperative that database access is kept secure.

## Implementation

At this stage, the program will be converted into code using the requirements and design decisions to guide development. The entity models will be created first, followed by the database and repository aspects. After, the intermediary aspects of the design pattern, including the view model and adapter classes (for displaying lists of real-time data), will be developed. At this point, the user interface is ready to be developed and functionality added to the program. The graphical aspect of the application will be developed using XML files that are associated with a particular screen.

Code refactoring (within project time constraints) and unofficial testing will occur side-by-side with project development at this stage. Testing will primarily involve finding and fixing bugs during the execution of the program, via integration and regression testing. Once the bugs have been removed to a reasonable extent, and the program includes all the functionality that it needs to, the next stage is verification.

## Verification

Official testing occurs here, with unit tests that concern specific, small sections of the program. Because the testing during implementation generally deals only with expected input, it is especially important to inspect how the application handles invalid data inputs. These tests will be in a class separate from the executable code. An example of the testing conducted is to take sample/existing data, perform a database operation on it, and test to see whether or not the database version matches the sample data.

The application will also be tested across a number of devices, using both real and emulated devices. The final test, of course, is the customer acceptance test. The project will not move forward to completion until the church agrees that the application meets the requirements.

## Deployment and Maintenance

After the application passes the test cases to the satisfaction of the customer, it can be officially rolled out to the church users. Users will be able to install the application on their phones and use it instantly on compatible devices.

# **Deliverables**

Provide information about what deliverables are related to your SDLC method. List and describe those deliverables. Also, include examples to help clarify what specific type of artifacts will qualify.

For example:

There are 2 types of deliverables that are associated with the Waterfall SDLC that the customer has requested. They are project and product deliverables.

## **Project Deliverables**

* Project Schedule
  + A granular schedule created and maintained by the project manager, which will be used to track development milestones and help ensure the project completes on time and within budget
* Requirements Document
  + The church’s wants and needs will be translated into a requirements list, that will serve as the measuring stick for customer acceptance as well as guide the overall development of the application
* Test Plans
  + The project manager will assist the church in developing a test plan that will be used to determine if the application meets the requirements from the customer’s view
* Wireframes
  + A series of hand-drawn wireframes will be created to determine the acceptability of the general UI layout design
* Mockups/Layout
  + High-fidelity representations of the various app screens. The church will be able to give feedback on color schemes, fonts and font sizes, button layout, navigation options, etc.
* Budget
  + A list and detailed breakdown of expected costs incurred during development

## **Product Deliverables**

* Fully functional GUI
  + Includes searching, filtering, and reporting features
* Security features
  + Login authorization required to access specific member/group data
  + Programmatic restriction of data and admin functions
* Special features for admin users
  + Add/modify/delete data entries
* Limited guest access
  + View church events
  + Redirect to contact church staff through device functionality
* Custom functional database

# **Implementation**

Initially, the separate login database will need to be populated with user credentials. The church, at its discretion, does not currently want users to be able to create their own login account. The project manager and database administrator will assist church staff, at a time and date of their choosing, with entering all member info and denoting admin status where applicable, as well as transferring any remaining hardcopy data to the database. Because this is a new system, there will be no service outage for the church. At this point, another round of acceptance testing will be conducted on staff personal devices.

Immediately after customer acceptance of the application, the application will be digitally signed and an APK file will be generated. The APK will be submitted for review by Google to add to the Google Play store for public installation. The database administrator will be available for installation/setup assistance if needed, although this will likely be unnecessary as the church will send an email to members with instructions.

# **Validation and Verification**

A battery of unit tests will be developed and employed by the developer to discover any bugs that would prevent the requirements from being met. The unit tests will be contained in a separate package from the executable file and can be run as an automatic, all-in-one test on the Android Studio IDE. The test environment will create a non-persistent, dummy database and will focus primarily on database functionality, specifically ensuring that all SQL commands are working correctly, and the system is responding appropriately to database transactions.

A security test will also be performed on the application, focusing on login activity, and testing to verify that application guests (non-authenticated users) cannot gain unauthorized access to data.

The church, with the guidance of the project manager, will develop the criteria of the acceptance test. Any tests dealing with program logic will be added, if possible, to the developers’ automated tests. The acceptance test will be developed according to the 3 main use cases: admin, general member, and guest. It will involve overall system testing, including login/logout, data persistence testing, database transactions (CRUD). Accessibility testing, including UI layout and font size/color will also be important due to the relative age of the user base. One of the most important tests for the customer will be testing for valid/invalid authentication on the login screen.

# **Environments and Costs**

## **Programming Environment**

It is expected, and mostly confirmed from the church, that the hardware infrastructure is already in place to allow users to use the application as delivered. The application was developed with device compatibility in mind, and the *minimum* requirements are as follows:

|  |  |
| --- | --- |
| **Operating system** | Android 4.3 Jelly Bean |
| **Processor** | Intel Atom® Processor Z2520 1.2 GHz, or faster processor |
| **Storage** | Between 850 MB and 1.2 GB, depending on the language version |
| **RAM** | Minimum of 512 MB, 2 GB is recommended |
| **Hard Disk** | * 2 GB of available hard-disk space for [operating system] installation; extra free space is required during installation. * You cannot install using a removable flash storage device. |
| **Browser/**  **Internet** | * This application is designed to work offline. * To download and launch Google Play\* Store apps within the application, a high-speed Internet connection is recommended. |

Table . Minimum System Requirements for Android\* 4.2 and 4.4. (Intel Corp. 2017)

## **Environment Costs**

Environment costs are currently minimal for this project. If and when the church migrates to a cloud server, a small monthly charge from the cloud provider will be incurred for upkeep. Please visit <https://calculator.aws/#/> for an estimated pricing structure from Amazon Web Services. Optional assistance with app installation on personal devices after acceptance testing is available at no extra charge (see Table 2 below).

## **Human Resource Requirements**

The bulk of the development costs will fall under this category, for paying the wages of the project employees (see Table 2 below).

| Resource | Type | Description | Total |
| --- | --- | --- | --- |
| Project manager | Human Resources | Cumulative payroll – 4 weeks | $6,000 |
| Software developer | Human Resources | Cumulative payroll – 4 weeks | $4,500 |
| Database administrator | Human Resources | Cumulative payroll – 2 weeks | $2,000 |
| Cloud storage | Environment – data storage | Monthly service charge | < $1.00/mo |
| Produce license | Intellectual Property | One-time purchase | $10,000 |
| Application support | Environment – tech support | 6 months | (included in product license fee) |
| Grand Total |  |  | **$22,500** |

Table . Estimated Pricing Structure

# **Project Timeline**

Due to the low risk of project changes and unexpected delays, the following timetable, while still an estimate, will represent the likely overall project duration. Per the waterfall development methodology, each phase will occur sequentially, with minimal overlap.

| Phase | Milestone/Task | Deliverable | Description | Dates |
| --- | --- | --- | --- | --- |
| Pre-development | Gathering requirements | Requirements document | Meeting with customer and procedure review | 5/25/2020 |
| Design | Class design/UI design files, app screen flow diagram | Low fidelity wireframe,  High fidelity interactive mockup,  ERD | Create the UI that relates the look and feel of the project, create database structure | 5/26/2020 – 5/28/2020 |
| Development | Software implementation | Completed project code, Database setup, Unit tests | Programming the requirements into Java code | 6/1/2020-6/13/2020 |
| Verification | Unit tests, user acceptance testing | Customer acceptance document | Final customer sign-off received upon satisfactory completion of tests | 6/15/2020-6/17/2020 |
| Google Review | Product release | 3rd party acceptance of product for public distribution | Google will review app for consumer safety and legal/policy compliance | 6/18/2020-6/19/2020 |
| Deployment/Rollout to Play Store | Product release | Finished product | App is officially located on the Play Store | 6/22/2020 |

Table . Estimated project timeline (summarized)