

Life Directions Mobile Application

Software Systems Capstone Project Spring 2020

Group Report

Christopher Garcia
College of Aviation, Science and Technology
Lewis University
Full Stack
Romeoville, Illinois, US
christopherygarcia@lewisu.edu

Flamur Kelmendi
College of Aviation, Science and Technology
Lewis University
Full Stack
Romeoville, Illinois, US
flamurkelmendi@lewisu.edu

Peter Alvarez
College of Aviation, Science and Technology
Lewis University
Team Leader
Full Stack
Romeoville, Illinois, US
peterjalvarez@lewisu.edu

Abstract

The goal of this project is to develop a mobile app for Life Directions, a non-profit organization. The target audience for this app is younger, religious students enrolled in a religious retreat offered by Life Directions. This mobile app is similar to a planner and other prayer apps. The app will allow the user to read from nine different scriptures as they attend the nine day retreat. Specifically, the app is meant to keep track of what scriptures have been read and when. The user will clearly be able to see when they have missed out on a scripture reading and notify them. The app will help members in the retreat navigate and access any bible scripture and other content throughout the program.

Table of Contents

Abstract.....	2
Theory and Background.....	4
Introduction.....	8
Approach.....	10
Design, Implementation, Testing, Security, and Documentation.....	12,13 18
Balancing Requirements with Budget.....	12
Application of Prior Knowledge.....	18
Conclusion.....	20
Works Cited.....	21
Appendices.....	22

Theory and Background

SDLC (Software Development Life Cycle)

Software Development Life Cycle – The process of implementing and executing an information system. Included in the life cycle is planning, analysis, design, implementation, and maintenance.

A. The concept of software engineering is heavily tied to the software development cycle.

Software engineering is the planning, creation, and upkeep of a software solution

a. Agile software development process and management

- Incremental software development and planning - Requirements are recorded on “story cards,” and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development “tasks” (p 78).
- Pair programming - Developers work in pairs, checking each other's work and providing the support to always do a good job ” (p 78).
- Simple Design - Enough design is carried out to meet the current requirements and no more ” (p 78).
- Small Releases - The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release. ” (p 78).

b. Software requirements and specifications

- Understanding and managing the software specification and requirements (what the software should do) are important. You have to know what different customers and users of the system expect from it, and you have to manage their expectations so that a useful system can be delivered within budget and to schedule (p. 26)
- Requirements engineering is the process of developing a software specification. Specifications are intended to communicate the system needs of the customer to the system developers. (p 69).
- Functional requirements - These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the

functional requirements may also explicitly state what the system should not do (p 105).

- Non-functional requirements - These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services (p 105).

B. Software engineering ethics are social and legal obligations software engineers have regarding their work. These ethics include confidentiality, competence, computer misuse and intellectual property rights.

- a. Confidentiality - should be kept between your client or employer whether or not an official agreement is made
- b. Competence - you shouldn't promise a level of skill that you do not possess
- c. Intellectual property rights - you should acknowledge and respect laws governing intellectual property like copyright and fair use.
- d. Computer misuse - You shouldn't use your technical knowledge in a malicious way (hacking, exploits etc.)

C. The software development process is the steps taken to create software. The steps include analysis, specification, and validation (p. 55).

- a. Requirements and analysis - This step is where you gather requirements from your client and potential users so that you can develop your system. Also included in this step is the creation of system models and diagrams.
- b. Requirements and specifications - After analyzing your requirements, you should narrow down the information you gathered into a document that specifies your requirements.
- c. Requirements and validation - At this stage you check your specifications for achievability and progress. Here you will need to make corrections caused by errors in your document. (p. 55)

D. Agile development differs from standard plan-based development by incrementally developing software. Agile is most commonly used for small to medium scale projects and for situations where there is heavy customer involvement in the development process. The Agile method works well in these situations because it can easily and quickly adapt to a customer's needs, and because it makes launching a project quicker.

- E. Software requirements and specifications are the necessities and guidelines software developers must keep in mind when creating software. Software requirements may include:
- a. Hardware - The physical devices that software will run on including computer systems and network requirements
 - b. Software - Other software that your software may depend on. For example, your software may require you to use a UNIX based operating system
 - c. User needs - What your client expects or needs from your software
 - d. Budget requirements - The total cost of all requirements
- F. Software system modeling techniques are abstract ways that software engineers model their software. These models give different perspectives of the system (like user side models, development planning models). System models are not full representations of the finished system, but simplified models that help users understand the system. (p 319).
- G. Unified Modeling Language is a commonly used and well-defined set of diagrams for creating system models.
- a. UML diagrams are composed of UML and show the process of a model for the development of software. The diagrams show the flow of and components that make up the software. UML diagrams aid the software development process by (p. 198):
 - i. Helping developers understand internal and external components in the system
 - ii. Designing the system structure
 - iii. Discovering the most important components in the system
 - iv. Create design models
 - v. Describe interfaces
- H. Software testing is used to ensure that software works as planned before it is made publicly available.
- I. Software evolution is the long term planning and development of software to ensure it is not out of date and keeps up with client needs. Software evolution follows a cyclical pattern of change and starts with a new system (p 259). This pattern includes:
- a. Change identification process
 - b. Change proposals

- c. Software evolution process
- d. New system

Introduction

Client and Project

The client is aiming to increase young adult agency through mentorship and guidance. To achieve this, the client works with a group called Life Directions that focuses on social-emotional variables that may affect young adults including low income, school dropout rates, and violent communities. Programs provided by Life Directions include the Peer Mentor Program and the Neighborhood Enrichment program. The Peer Mentor Program is a cyclical process in which students who complete the program go on to guide and teach the program to incoming members. The Neighborhood Enrichment program helps connect youths to summer job opportunities. So primarily, the app will be used in religious environments that promote growth through religious readings.

The goal of this project is to provide aid to the client's' nine-day retreat program where students learn about the holy divinity. With this app, students enrolled in the program will access the app and can follow along with the lectures of the course. The course covers nine days that cover the Trinity, creation, marriage, fallen creation, the immaculate conception, incarnation, the cross, resurrection, and communion. Students will be able to access each day and get an overview of what will be taught on that day. After a student has read through a day in the application, the day will be marked off to show that they have completed that day. Also the user will have a profile that he or she will be able to access. With this profile, they will be able to share their progress and connect with other young adults on the retreat.

In regards to the future of the app the client has predicted his app is to only be used for these specific retreats. So, the app will be built in a way that doesn't have the ability to add more days to it. Although the app can be redesigned to add more days and select specific scriptures in the future. There are no plans of adding an admin facing version of the app for Life Directions staff to monitor. Overall, the client must reach out to the developers to request any changes, additions, or modifications besides the ones that have already been discussed prior to the release of the app.

The app utilizes React Native, an open-source mobile application JavaScript framework created by Facebook, It is used to develop applications for Android, iOS, Web, and UWP by

enabling developers to use React along with native platform capabilities. We decided to use react native because of our experience in JavaScript and specifically React (Web Framework). However, we did not have experience with React Native. One key factor about React Native is its ability to be cross-platform, this saved us a great deal of development time by allowing us to simultaneously develop for Android and IOS. For our database and back-end, we used Firebase. Firebase is Google's mobile platform that helps you quickly develop high-quality apps and grow your business. It includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer. Using Firebase improved and quickened our development process.

Approach

The Organization

Before we began development, the group decided to focus on learning more about Life Directions, a non-profit organization focused on “motivating young adults, ages 13-35, to mature into responsible, productive adults through self-direction” (Life Directions). The team has kept in contact with Van Bensett who works with Life Directions and has been discussing developing a mobile app for a nine-day retreat program. The nine topics covered in the retreat include the Holy Trinity, creation, marriage, fallen creation, the immaculate conception, incarnation, the cross, resurrection, and communion.

Users

From there, the group planned out what kind of users would use the application. At first, there was a plan to incorporate an admin view that would allow an Admin to add more days to the retreat. This was eventually scrapped due to time constraints and the fact that our client did not think of a case where he would use this feature. In the end, we decided that our app would only have one type of user which we called general users. General users are able to sign up and use all parts of the app with no restrictions and with any email.

Existing Software and Hardware that Relates to the Project

To start off, the group decided on the tools, software, resources, different components, and phases of the system. The main focus of the app is to be a mobile app that will be used for a nine-day retreat program in which it will display bible scriptures for every day of the retreat. To write this app, possible options were discussed such as Java or Swift. Java is known to be a robust language that can be used to develop apps in Android Studio. Swift, on the other hand, is a language that is used to develop apps solely on iPhones. Ultimately, since most team members had Javascript knowledge, we decided to use React Native. React Native is an open-source mobile application framework created by Facebook. It is used to develop

applications for Android, iOS, Web. This was a crucial reason for picking React since it is cross-platform. This means it can be written once and run on multiple operating systems.

Another major resource used in development was GitHub. GitHub allowed the group to source control of the program and ensure that each team member could access an updated version of the app. Since each team member was working remotely, GitHub provided a way for the team to see, edit, and add changes. Since each team member had experience with Git and GitHub, it was an easy decision to use it. As for software used to develop GitHub, it was crucial that each team member had installed node.js and React.js. As these were the frameworks that were used to build the application, it was important that each team member had them installed. Software used to write the application (IDEs and text editors) were developer independent. As long as the team member could write and compile code, any IDE or editor could be used. One last crucial tool each team member needed was a means to debug code and run code. Since the mobile app runs on phones, it was important that each team member had a smartphone emulator or an actual smartphone to run the app on. Each team member had access to an IDE that could run an emulator, and the team was also able to run the app on their own Android device.

To communicate the group used Discord. Discord is a free chat client that allows multiple people to interact via text and voice chat. This tool was crucial in the development of the application because each member could update others on their progress, share images related to the application, ask questions, and share links and resources. This sped development time up since members were able to get in touch with each other easily. Discord was a good alternative to email which would make it harder to talk in real-time with other group members.

Another important aspect the group researched was the hardware and software requirements of the user. Since the project is a mobile app, it was important to make sure any modern smartphone could run the app. This is why the decision was made to develop with React since apps created with it can be run on both iOS and Android operating systems. The app will be compatible with iOS 10.0+ and Android 4.1+ which allows for a wide range of users to access our app. As for hardware, users should have at least 2GB of RAM and a 1.6 GHz quad-core CPU. The dimensions of the phone display do not matter since our app will dynamically resize itself to the required resolution.

Budget

Overall, the app would run entirely free based on certain criteria. In the U.S the app can have to 10k requests a month for authentication. Right now, authentication requests are made when the user logs into the app. So with the small group that is projected to use the app, the free or “spark” version is more than enough. The group then analyzed how much storage the app would use over time. For the free version, firebase limits 1GiB total which the group projected the app would never reach. Right now, with all scriptures and 10 users, our database is using roughly 3.5mb of space. The only time our client may consider upgrading the database is if he decides to incorporate more days to the app.

Security Concern

The only data directly related to the users that are stored on the app is the first name, last name, email, and password. In terms of security, firebase uses a token system that prevents data from being stolen. Another layer of security is added by hashing the passwords so they are not saved in plain text. Since there are no plans of storing any sensitive data like credit cards our group decided that the security that was in place through firebase was enough. Also, the code for the project will always be secure since it is stored through GitHub and requires the correct credentials to access. Firebase also prevents the same IP address and device associated with it to make multiple accounts at a given time, this improves security greatly. There is a possible security concern that involves any user being able to make an account with an appropriate email. The client did not specify whether he wanted to approve every account before creation. Within Firebase and our app implementation, this is a possible feature.

Design

The overall layout of the project stems from a side panel consisting of four main pages. This includes the home page, scripture, profile, and account page. The group wanted to keep the number of panels on the navigation screen as minimal and simple as possible.

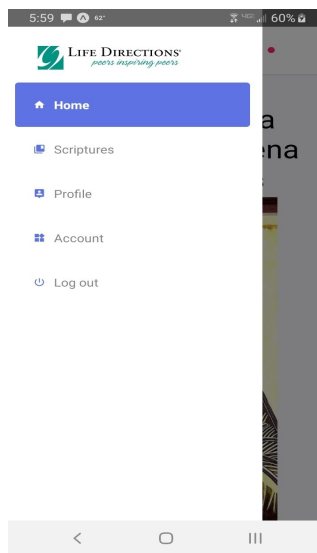


Figure 1. Nav panel showing the home, scripture, account, and logout button

The actual scripture page contains nine buttons with each respective day on them. At first, the group had the button redirecting directly to the scripture but the user was flooded with too much text. Instead, the group decided to create another page that split the day into morning, afternoon, and evening scriptures. So when the user clicks on the first day, a new page will appear with three more buttons. We also considered the user's experience as well when splitting the days up. The group decided that a user may not want to read all of the scripture in one sitting so splitting the day up would be beneficial in this sense.

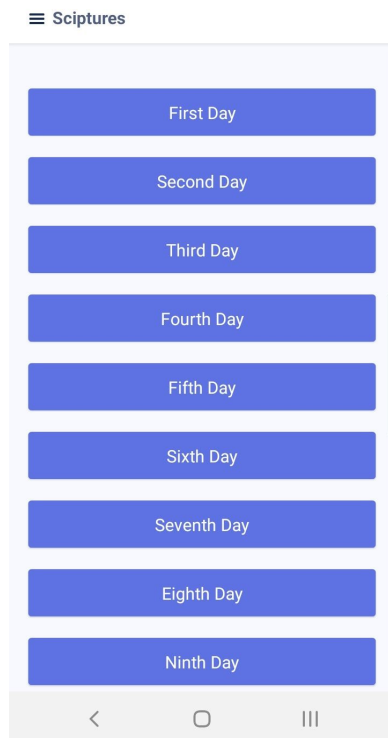


Figure 2. Days page. Shows all nine available days

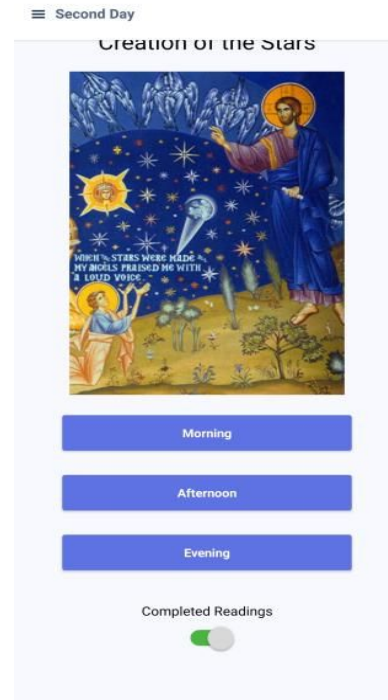


Figure 3. Displays the options for a specific day's page with the confirmation checkbox

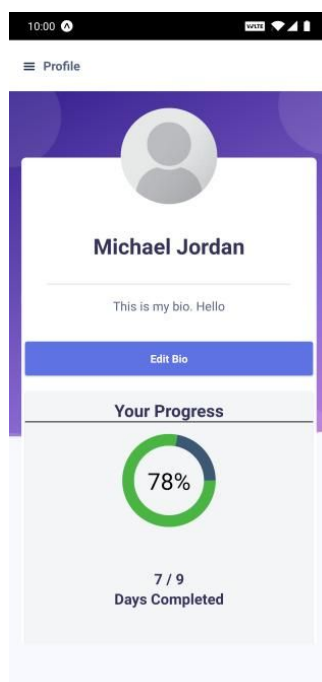


Figure 4. Profile page with progress bar

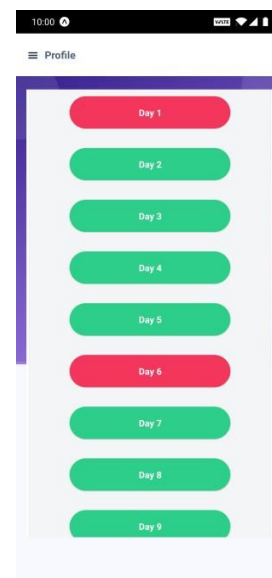


Figure 4.1 Profile page with days missing and completed

In addition to this, a checkbox was added to the bottom of each of the day's pages so that users could mark off when they have finished reading which you can see in figure 4. This checkmark is saved and added to the students' overall progress on their profile page. The profile page itself displays the user's first name and last name as well as their progress bar. The progress bar will fill as the user checks these boxes and show what days are missing like in figure 4.1

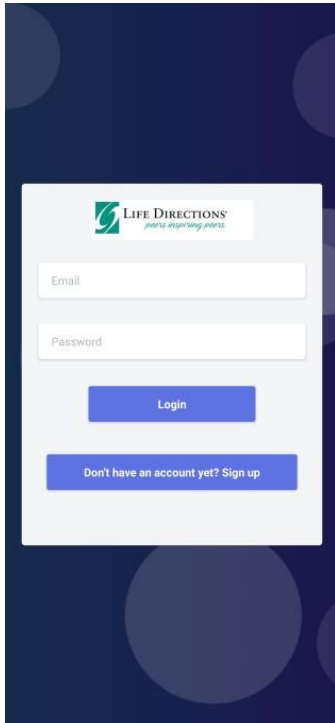
The sign-in page features a dark blue background with large, faint, overlapping circles. A white rectangular form is centered on the page. At the top of the form is the Life Directions logo, which consists of a green stylized 'L' icon followed by the text 'LIFE DIRECTIONS' and the tagline 'peers inspiring peers' in a smaller font. Below the logo are two white input fields: the first is labeled 'Email' and the second is labeled 'Password'. Under these fields is a blue button with the text 'Login'. At the bottom of the form is another blue button with the text 'Don't have an account yet? Sign up'.

Figure 5. Sign in page

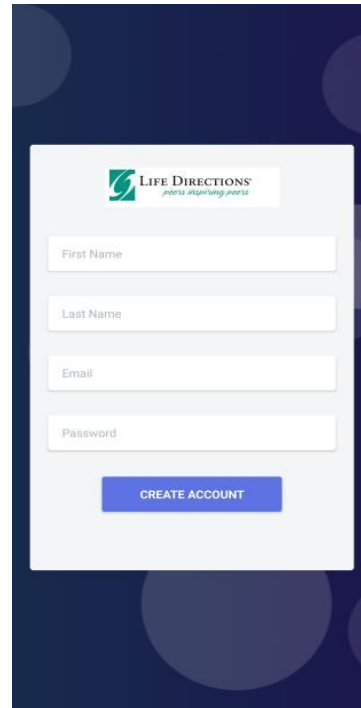
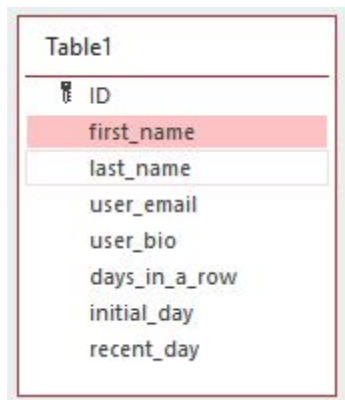
The sign-up page has the same dark blue background with faint circles as the sign-in page. A white rectangular form is centered. It features the Life Directions logo at the top. Below the logo are four white input fields: 'First Name', 'Last Name', 'Email', and 'Password'. At the bottom of the form is a blue button with the text 'CREATE ACCOUNT'.

Figure 6. Sign up page

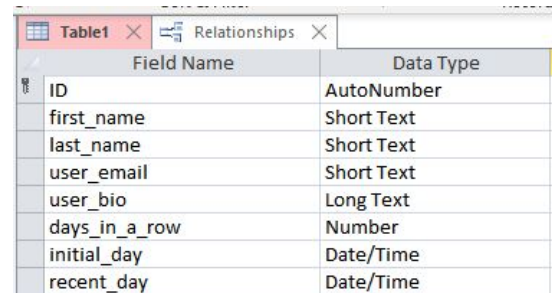
Database Design

To create the initial design of the database the group used Microsoft Access. From there Firebase was used to manage our data. The group wanted a good template to work off and see if all fields were necessary. Eventually, the group removed the `days_in_a_row`, `initial_day` and `recent_day` fields as it was not used in the application. Instead, the group added a progress field to keep track of the user's progress while using the app.



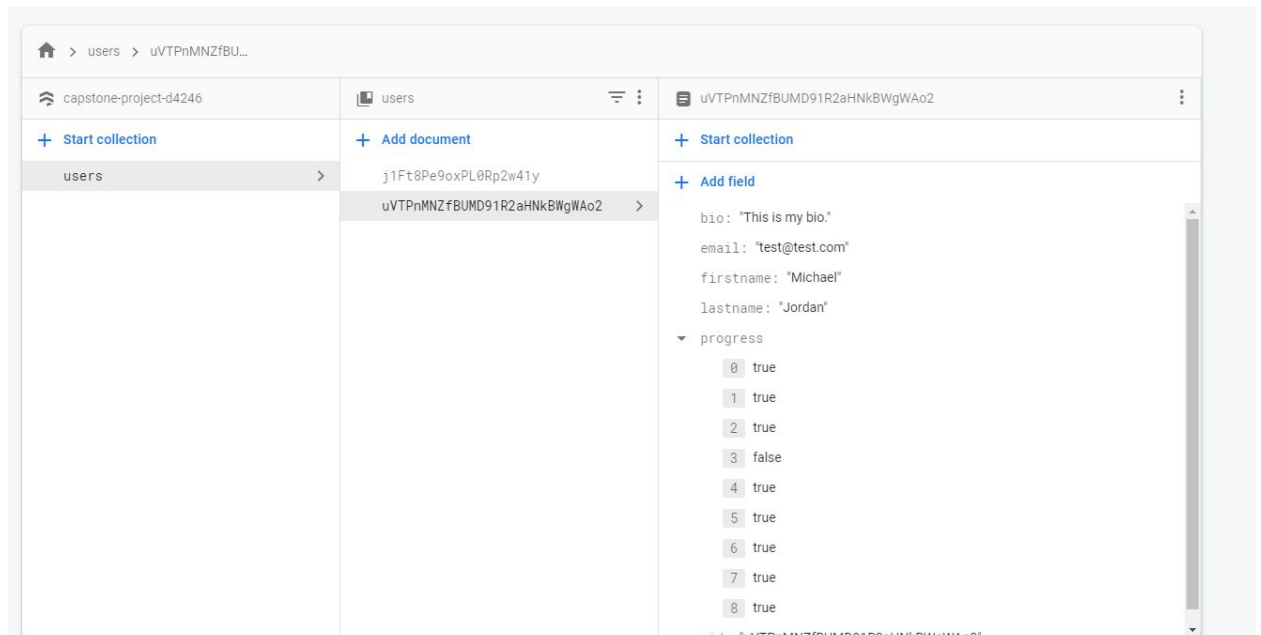
ID
first_name
last_name
user_email
user_bio
days_in_a_row
initial_day
recent_day

Figure 7. Access Database design template



Field Name	Data Type
ID	AutoNumber
first_name	Short Text
last_name	Short Text
user_email	Short Text
user_bio	Long Text
days_in_a_row	Number
initial_day	Date/Time
recent_day	Date/Time

Figure 8. Access Database design template



capstone-project-d4246	users	uVTPnMNZfBUMD91R2aHNkBWgWAO2
+ Start collection	+ Add document	+ Start collection
users	j1Ft8Pe9oxPL0Rp2w41y	+ Add field
	uVTPnMNZfBUMD91R2aHNkBWgWAO2	bio: "This is my bio."
		email: "test@test.com"
		firstname: "Michael"
		lastname: "Jordan"
		progress
		0 true
		1 true
		2 true
		3 false
		4 true
		5 true
		6 true
		7 true
		8 true

Figure 9. Completed Firebase database example

Hardcoded Values

With all projects, there are certain concerns that may arise through development. The main concern discussed was how the text was coded in the app. At first, the app hard-coded the text for the scriptures. Although this was fine for the nine scriptures, it would cause issues in the future if our client decided he wanted more days. After a milestone discussion, the group decided to use the “storage” feature in Firebase to store text files that would be pulled by using a generated URL created by Firebase. This fixed the hardcoded value.

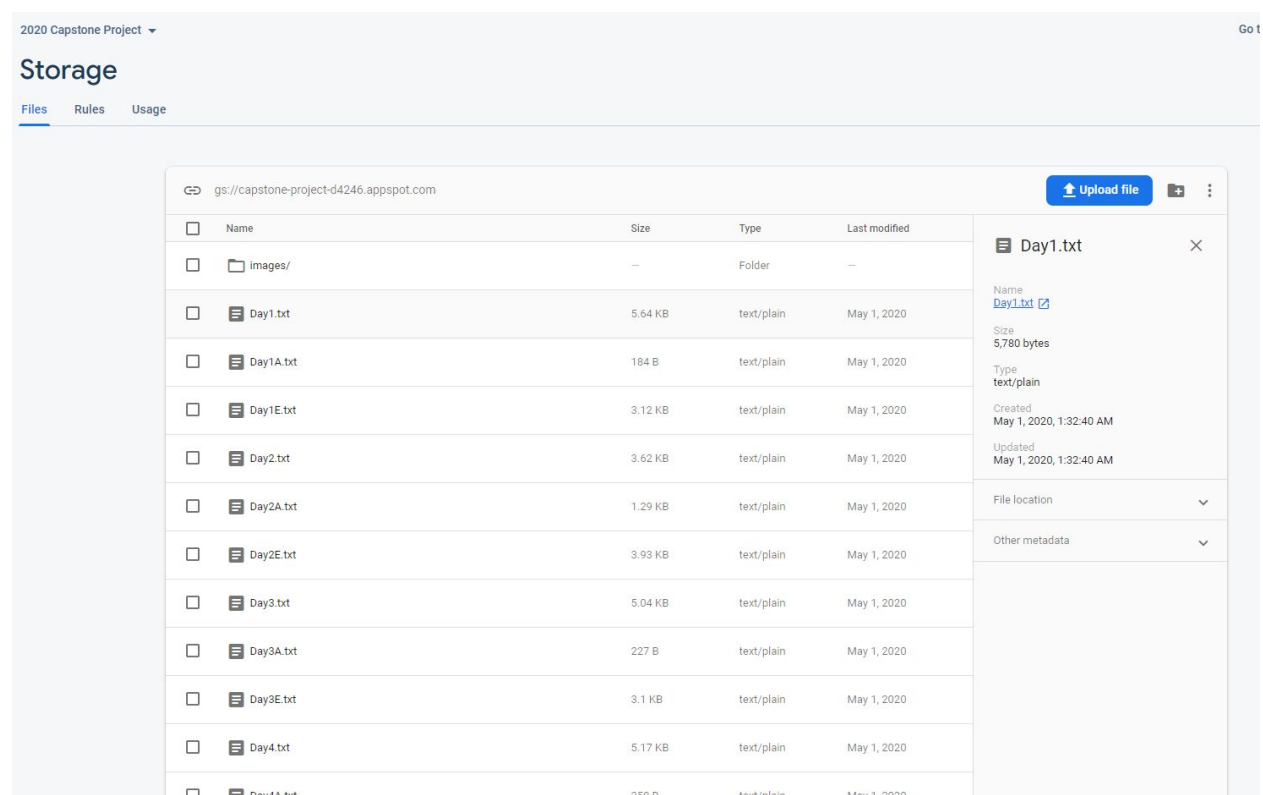


Figure 10. File storage using Firebase

Testing

Testing was done in increments as the group added features. For example, when the login feature was added the group had to test this feature on both IOS and Android. The main things the group would test for were related to user experience. This includes speed, design, and accessibility. The group tested for time by timing our requests and JavaScript/CSS would take to load. Since the data was small, speed was never an issue. The group tested for design by emulating different phones using emulators in Android Studio. Accessibility was tested by determining if the app was straightforward to use with no prior knowledge. The group did this for every large commit that was made. This involved the group to be up to date with the codebase at all times to not cause merge conflicts.

Potential Concerns

Adding More Days

Unfortunately, the group was not able to add the ability to add more days to the app. So if the client decided he wanted a 10-day retreat he would need to consult the group. On the other hand, the project is set up in a way that allows for this to be added anytime if more development time is given.

Learning Outcomes

Through our four years of computer science education, we have been introduced to different aspects of software engineering. This helped our development process, especially when we were learning new tools to complete our project. The various courses we had to take involved in introducing new programming languages developed our learning capability. Courses that involved object-oriented-programming, system development and database design aided us in this project when we dealt with the initial design of this project. We consistently used principles taught in those classes.

For example, the database design class at Lewis allowed the group to make a well-versed database structure. In the database class the students learn to create a relational

database with the according field. Similarly, in the project, the team needed to create a database that stores user information. This database consisted of fields like “first_name” and “email.” Without prior knowledge, the group would have had a hard time deciding what kind of data types they were working with. Ultimately, the database class in Lewis made this task a simple one.

The object-oriented programming class taught the principles of object oriented programming and its uses. An important principle of object oriented programming is to break code into maintainable and reusable components to help organize data and limit redundant data. This is something that the team tried to adhere to when developing the application, but was faced with the problem of data that was not uniform. For example, the group had the “navigation” panel which consisted of several nested pages which routed the user. The group made these pages in a way so that it limited the amount of redundant code. The team also broke the pages apart in a way so they could be modified without interfering with each other. In the end, the object-oriented-programming class at Lewis taught us to code more efficiently.

Another important thing learned from object oriented programming classes is that certain data should be kept private. Sensitive data like user passwords and payment information should be kept private so that it will not be maliciously or accidentally tampered with. Keeping components of objects private lessens the likelihood of a developer accidentally accessing components that should not be edited like user IDs. The group utilized these concepts when developing user account information. It was crucial that sensitive data like passwords could not be accessed or read unless they had access to the database itself.

Another useful class that we took was “Algorithms and Data Structures”, which was helpful to solve a single program in different methods and sometimes data structures also play a great role in solving a problem. It's important to study these structures because in complex computing problems such as search, sort, hashing, etc many of such structures are used. The coding problems we dealt with on a daily basis were based around designing and implementing data structures and algorithms in such a way that data in processes and accessed in minimal space and time complexity. This would be useful if the group would have decided to add some sort of “search bar” functionality to their project.

Conclusion

Using previous knowledge gained from classes and coursework the group created a functional application for the Life Directions organization. Specifically, the group developed a prayer application to be used by students enrolled in a retreat program offered by Life Directions. By carefully assessing the needs of the organization and making sure to adhere to the organization's core values, the group was able to successfully develop the app in a timely manner.

Works Cited

Sommerville, Ian. *Software Engineering*. Harlow, England: Pearson Education Limited, 2016.

Life Directions, *Life Directions Peers Inspiring Peers - Life Directions*, www.lifedirections.org/., May. 01, 2020 [05/01/2020].

Appendix - See Attached files

Screens.js

The screens.js file is used for creating and routing the user when they click on certain buttons. For every button that redirects the user there needs to be a corresponding navigator component that exists. So, all the redirection of pages is in this file. The logic of the code is divided into three parts. The first part is the importing of all screens or “pages” into the screens.js file. So, for the first day of the prayer app which directs the user to morning, afternoon, and evening prayer, the group had to import all three of these files. Then, a function is built that references these import statements. The purpose of these functions is to navigate the user to the screen with certain parameters. The last part is to let react-native know what components these screens will be using when they load. For example, the screen FirstDayMorning would have the FirstDayMorningStack Component. Overall, this file specifically is rather redundant and could be shortened potentially. Currently it stands at 1000 lines of code.

Package.json

The package.json file is responsible for tracking all versions of open-source software libraries/tools that are currently being used in the app. The file itself should be generated automatically with the node package manager. The file itself is useful because as many programmers know, syntax is always changing. This is especially true when frameworks get updated. A good example is when you look at Python 2.7 and 3.0. So when another group member or programming looks at the codebase for the first time they are

User.js

The user.js file consists of functions that are throughout the project that pertain to any user information. This includes email, password, first name, last name, login, and signup. For example, in the file there is an updateEmail() function that updates the users email. A more complicated function is the login function which requires the parameters email and password. After receiving these parameters it will require authentication from Firebase to validate the request. These functions are used throughout the project when any information is needed about

the user as well. A good example of this is when the progress bar is updated and the `getUser()` function is called.

Signup.js

This file holds the code for your sign-up page. This is the page a user sees when they opt to make a new account. The design is influenced by the initial login screen. It is a modal-like block of form inputs behind a design image. The form inputs have an `onChangeText` property that signals a function to a global function called `updateFirstName`, this is what updates the global state of the parameter `firstname`. This is the same for `updateLastName` and `updateEmail`. Once the user has finished their form inputs, they click the `Create account` button. This will fire the `handleSignup` function that handles the form object data. `handleSignup` is a function included in our actions file `user.js` among many other form handlers.

Login.js

The `login.js` file holds the code for the login page. The page itself is called when the user clicks on the `login` button. They are then required to have an existing account which is composed of an email and password. In the code, when the user submits their requests to login, they are making a request containing two fields which are username and password. The function that handles this requests is in the `users.js` file. If the user has entered a valid username and email it will log them into the app, otherwise it will display an error message letting the user know the information they entered was incorrect.

All Scripture Day Files

For all the nine days in our app the group needed a large amount of text and some specific images based on the day that was being displayed. A good example of a file like this is the `ThirdDayEvening.js` file. Since all the files and images are stored in a storage container in Firebase the group had to pull these files to be used in the code. In a file like `thirdDayEvening.js`, there is a function that uses three hard coded variables. This includes the url of the storage container, text file, and image file. The function itself will pull the text file and append each line of text one by one to a predefined variable. Another function will take the

image and store it in an image variable. Once the variables have been filled with the correct image and text file it will use these variables in a React Text component. Finally, the text will be resized with CSS so it is displayed to the user cleanly.

Profile.js

The “Profile.js” file contains the code displaying the profile page for the user. The profile page displays user information like profile pic, first name, last name, bio, and progress. This file is one of the more complex ones because we are handling and manipulating many of the user data. As the component first renders, it loads up the component “state” with the global app “state”. This includes the user’s name, bio, progress, and profile pic. The profile page has an option to add a new profile pic, this button enables a user to access the gallery on their mobile phone and upload the picture. Firstly, The image gets sent to the firebase storage, it then returns a link that is able to be downloaded and displayed within the component itself. The component retrieves the download URL from firebase. When a user presses on “Edit Bio”, a modal pops up, enabling the user to type out their new bio or add to the already existing one. When the user clicks submit, the “handleBio” function is executed. HandleBio changes the state within the local component and the global app. A pattern that most of our handle functions implement.

Scriptures.js

This file displays our “Scriptures” component which is the first screen that shows up when users press on the “Scriptures” option in the navigation bar. The screen includes a list of button options, signifying the different scripture days provided by the client. The only logic implemented in this file was the different navigation that is executed. We implemented a Javascript Object that includes the “title’ of the navigation per button, this allowed us to differentiate the different navigation options.