UNT DALLAS
UNIVERSITY OF NORTH TEXAS AT DALLAS

Department of Mathematics and Information Sciences
University of North Texas at Dallas
Dallas - Texas

University of North Texas at Dallas Mobile Application

IT Capstone II

Jared Giles

Ana Sanchez

Miguel Torres

Edinho Lopez

Supervisor:

Dr. Saif Al-Sultan

May 2, 2019

Abstract

The university mobile application is a system that will allow student users to access a multitude of features. Students will be able to easily navigate the university, discover events and organizations, and have access to their classes among other features. The motivation behind this project is to help connect the community to our campus through a mobile application. We will add and improve the basis upon which the current University of North Texas at Dallas Mobile Application works. The application will be developed using Kotlin as its programming language, SQLite as its Database Management System, and the Incremental Model for its software development process.

Contents

List of Figures

0.1 Acronyms

DML Data Manipulation Language

DBMS Database Mangement Systems

DCCCD Dallas County Community College District

DFD Dataflow Diagrams

ERD Entity-Relationships Diagrams

GUI Graphical User Interface

IPO Input/Process/Ourput

JVM Java Virtual Machine

MVC Model View Controller

SQL Structured Query Language

UML Unified Modeling Languages

Chapter 1

Introduction

Objectives:

- Provide an introduction to our proposed system.
- Outline each chapter of the report.
- Explain the motivation and assumptions associated with the system.

1.1 Introduction

This chapter will introduce the motivation behind proposing our software system. The chapter will also discuss some assumption which were made when proposing the system. This chapter will outline the other chapters of this proposal alongside a short description of each one. Finally, a time chart for each major component of the software system.

1.2 Problem Definition

Our application will help students at the campus manage their student life in various ways. Most students in the modern world are using their phones often, so they can constantly communicate with others, and as a result there are many applications available for many colleges. However, many of these apps are not optimized for one reason or another. Our solution is to build an application which will be used for UNT Dallas, which students can download and use in order to connect to their classmates, as well as use many other features that students may need to be successful in class.

Our proposed system will have the ability for students to use either guest features or log in and use more personalized features. They will have the ability to access many different university-focused features like blackboard, emergency services, and info about their classes, as well as social-focused features like organizations, social media, and connections to other students. Our system will use a database to store student information as well as information about the different organizations and other information.

1.3 Motivation

The motivation behind this project is to help connect the community to our campus through a mobile application. We will add and improve the basis upon which the current University of North Texas at Dallas Mobile Application works. The current mobile app does not fully integrate the student body or the community as well as one would hope when trying to get the students and community involved on campus.

We compared the current application with others from different campuses, and we were able to distinguish some differences as well as some similarities between each of them. Our intention is to implement the basic functions that most of the applications offer, as well as build upon them in order to make the application unique and provide a holistic approach to our campus life.

1.4 Assumption

For our project, there are some assumptions that our group has made in order to have our system working efficiently. With a project that would depend on the university's database, we assumed that the users will be able to have access to the information provided through the university's database through the different tabs we would provide.

1.5 Project Outline

The following describes the proposed outline of the project.

1.5.1 Chapters organization

- **Chapter 1: (Introduction)** This chapter serves as an introduction to the system we plan to develop and the report of the system.
- **Chapter 2: (Literature Review)**
  This chapter presents similar system to ours and tools needed for implementing our proposed system.
- **Chapter 3: (Requirement Specification and System Modeling)**
  This chapter defines the requirements of the system in terms of user and system requirements.
- **Chapter 4: (Design)** This chapter shows the user interfaces that the system will use. It also shows the modeled database.
- **Chapter 5: (Implementation, Testing and Evaluation)** This chapter presents all the code used for the system's implementation.
- **Chapter 6: (Conclusion and Future Work)** A conclusion of the proposed system will be given in this chapter. In addition, this chapter will present the possible directions of the future work.

Chapter 2

Literature Review

Objectives:

- Provide background information related to the system.
- Describe the tools available for implementing software systems.

2.1 Introduction

The first section of the chapter provides some related work to the project we plan to develop. The section also discusses similar software systems that have been developed. The next section addresses software development. The section introduces some software development processes and presents our choice. The following section addresses the tools that could be used to develop the system such as the programming language and the Database Management System. Finally, the chapter ends by presenting our choice of tools. This last section discusses the programming language, the Database Management System and the Unified Modeling Language.

2.2 Related Work

This section introduces similar software systems that have been developed. There are also academic articles related to the relevance and justification of our project.

The Texas Tech [1] mobile application is a very useful tool that many university students can really rely on as it provides access to academic and social information related to the campus. Their features range from campus emergency services, university's menu, and academic needs. Campus emergency services provides important contact information in case of emergencies. The university's menu feature provides students with information regarding food being sold at the different cafeterias. The academic needs features provide students with access to their assignment, information about their courses and important campus announcements.

Some of the features the application offers are shown in different categories that the user can choose from, and narrow down their choices. The app's categories are: Announcements which provides information regarding campus announcements, Texas Tech University which provides general information about the campus, Go to Texas Tech which provides information regarding applying to the university, MyTech which provides access to the university's email and student profile, Faculty/Staff which serves as a directory for faculty, and an Employee tab which provides information regarding university's employment. They offer a variety of sub-categories for each of the tabs mentioned above. Their menu options are quite extensive, giving students many options to choose from. The application provides important campus information is easily accessible.

[2] is a standard mobile application. Students can regularly check their grades and check their classes.

There are many features, for example, the athletics tab. This feature provides information about topics related to sports, such as videos and images of the football team. Not only can you just see information about them, you can also purchase tickets to attend certain sports events.

They also offer a radio station feature which allows you to hear news about the campus and sports teams through your phone. Their social media pages seem to be sports inclusive, and shows only events that the university is hosting or has hosted. There is a scheduling tab, that only gives you information about scheduled games going on at the school. The app lets you log on or create an account so you can keep track of all the sports teams that you follow.

This application [3] is a university-based mobile application. This application gives several features that adapt the functionalities of the university's website and other applications. Several unique features have been implemented in this application, ranging from the ability to take school related polls to listening directly to the school's radio. Every feature is incorporated within the application, it does not require the use of external applications and does not redirect the user to the university's website.

The University of Oklahoma has a mobile application [4] that provides students with information about the campus. The application is straightforward, providing only a key set of features. Features include access to Canvas, the student's campus email and enrollment for classes, among others. Each feature is presented with an icon and a title, each square icon is then laid out in a grid. Some of the features require students to provide their log in information to proceed while others do not. The application incorporates most elements of the university's website but also provides some extra features for student commodity.

The Texas State app [5] is like most college apps, with a good number of features. It has a simple design, where the app connects to the school's website and displays the mobile version in the browser of the app. It has the basic features necessary for current or future students such as the course catalog, directory, and the academic calendar. It has a good map feature for the college campus which displays all areas for parking, bus routes, and the different buildings on campus. There are social features available that connects the mobile application to other social media apps. There is a library page where it connects the user to their online database, where they can see the books that are available to them, as well as events going on in the library. When it comes to the events, there is no centralized place for the events, since there is a tab for a series of events called Innovation weeks, an events tab, and the events under the library tab. Finally, there are notifications, resources for future students, and other minor features such as photos, emergency info, and athletics.

The Louisiana State University mobile application [6] had one interesting feature that makes it stand

out, the fact it runs on top of a platform that many colleges or other institutions use for their mobile app. It is called Modo and provides a framework for apps to use. The app uses the hamburger menu design that is recommended to be used with Android applications, allows for different personas such as for a guest, student, alumni, or staff, and has a grid menu for each persona with their own features. While some aspects of the app are like other college apps, where features are just a redirect to a webpage with the built-in web browser, this app's uses are better than the average college app since it has more built in features. When it comes to the Modo platform used, it provides an environment for apps to connect different data sources into the app. While it is easiest to use for premade sources of data such as social media, some institutions have even been able to connect their bus stop or parking lot data by using the platform.

The application [7] provided by the Dallas County Community College District (DCCCD) to its college students include plenty of useful features all in one app. Some of the features offered for current and future students are things such as course catalogs, degree plans, and where to apply. Aside from that, the app also offers some other functions such as map views, using google maps, to help students get around campus. The feature also shows some of the events that are going on at the school. The other main feature which is similar to a directory, is the emergency tab, which houses all the emergency contacts in the event of an emergency.

The app from the University of Texas in Arlington [8] is sleek looking and more modern. There is consistency in the theme and icons throughout the app. When the app is first opened the user is welcomed to several options that the user can select which consist of Events, map, news, emergency, photos, and some other options. When the user taps on one of the options the user is taken to another page that displays nicely into a mobile friendly format. The app does a good job displaying content throughout the app, and rarely redirects the user to full web pages. As a student using the app, the app does a great job to find content available in the school, and around the campus. However, there are no options in the app for the student to sign into their school account and check their schedule or see what assignments they might need to take care of.

2.3 Our proposed mobile application

There are several features that our proposed system would incorporate. Some of them are: the ability to obtain information regarding courses and grades, a feature that provides a detailed map of the campus, and a feature that allows access to the school's social media. Several other features will also be adapted, some of

them are implemented in other similar systems. There are other features that our system would implement which are not found in any of the other similar systems. These include, but are not limited to, a feature that provides information to clubs and organization within the campus and a feature that allows students to make posts and socialize in a shared page.

2.4 Software Development

Software engineering deals with the design, implementation and maintainance of complex software systems. There are different approaches to developing software, each with specific guidelines and procedures. The structure in which tasks required to develop the software are selected and completed is called a software development process[9][10]. This section presents different approaches to the software developing process:

2.4.1 Incremental Model

The incremental model is a software development model, which can actually be considered two models, divides the process into different sections, called increments. The incremental model divides the product into different components, where the components are usually the major requirements, and each component is developed separately. Each component goes through the main phases of the development process, including design, implementation, and testing, and each phase can go through multiple iterations. Once each component is completed, they are joined together and continuously integrated, and the complete system is tested until the process is complete. The other model included under the incremental umbrella is the incremental release model. In this model, the core functions are developed for the first release, with additional features being added in new incremental releases, as well as any bug fixes from previous releases. Both models use the divide-and-conquer methodology since their process is divided into parts, worked on separately, and then joined together in order to solve the problem. However, both require a higher level of understanding than some other processes, since the developers must know how each section will connect to the others as well as when dividing the problem into parts[10].

2.4.2 Spiral Model

The spiral model is one of the most important development life cycle models, which focuses on risk handling. The model diagram is basically a spiral that has multiple loops. The spiral model also does not have a set number of loops since the loops can vary depending on the project. Every loop in the model is a different

"Phase" in which the project is traversing in the software development process. Another factor that dictates how many phases are needed to develop the final product will vary depending on project manager and the project risks. One way to determine the cost of the project at any point in the software development process is to get the radius of the spiral. Each phase of the spiral model is divided into four sections, a typical traversal through the four quadrants is as follows: Identify the objectives, alternatives, or constraints for each cycle of the spiral. Evaluate the alternatives relative to the objectives and constraints. In performing this step, many of the risks are identified and evaluated. Depending on the amount of and type of identified risks, develop a prototype, more detailed evaluation, an evolutionary development, or some other step to further reduce the risk of achieving the identified objective. On the other hand, if the risk is substantially reduced, the next step may just be a task such as requirements, design, or code. Validate the achievement of the objective and plan for the next cycle

A key part of the spiral model is the review of all the activities and products completed in each cycle by all the people that were involved in the project. The reason behind the review is so that every party is committed to the project and agree to be on the same page to advance into the next phase of the project. The spiral model is focused on risk reduction by repetition, which includes several benefits such as: The model incorporates prototyping and modeling as an integral part of the process. It allows iterative and evolutionary approaches to all activities based on the amount of risks involved. The model does not preclude the rework of an earlier activity if a better alternative or a new risk is identified [10].

2.4.3 Scrum Model

The Scrum model is very popular among the Agile methodology. The process consists of three main categories: Roles, Artifacts, and Time Boxes. The Scrum Roles are the Scrum team, the Scrum Master, and the Scrum Owner. The Scrum Artifacts include the Product Backlog, the Sprint Backlogs, and the Increment. The Scrum Time Boxes are fixed amounts of time that are given in order to complete the activities given to you. The Scrum method is most commonly used when there is a large and complex software development that needs to be completed. This process uses iterative and incremental releases to publish the software that meets the expectations of the organization. Some of the benefits that the Scrum process brings to the organization is that the process allows in an increase in the quality of the software being delivered, and since the process is a subset of Agile, it allows for adaptability when changes are constantly being made. The Scrum process is also simple for others to understand, which is a benefit since most companies have groups of people that are working together to produce the product that is required. Scrum implements its approach

based on the organization and the people that are involved in the product development and creation. It helps simplify the complex problems that may arise into more simple terms that the team can understand[10].

2.4.4 XP Programming

XP programming is an agile software process model. The model involves the use of pair programming and unit testing to complete the project in a timely manner. Pair programming involves the pairing of two programmers when writting code. This ensures that the code is also revised while being written because errors and easler to detect by a member. There is a need for constant feedback from users, but the model compensates for this by embracing constant changes in the design and the requirements. This process model is best suited for small to medium projects. The topics these projects deal with should not be error sensitive due to the high possibility of bugs [9] [10].

2.4.5 Software development process chosen for this project

Extreme programming is one of the more popular Agile Processes. It has been more successful because it emphasizes customer satisfaction. Instead of receiving a product with many features right out of the box, although with a very far away release date, the customer receives a smaller version of that software at an early date with the main and fully functional features. After that newer updated versions are pushed out that help fix and patch bugs, and new features on the newer versions. That way the customer still gets a functional app with working core functions, which will eventually get newer features and get better with time. Extreme programming is also very heavy on teamwork. Every staff member is equal partners to one another in the team. Having a good team that can work together and able to be effective will also create more productive environment in which problems are tacked by the team as whole instead of everyone tackling many different problems on their own[9][10].

Some of the main core values that are emphasized to ensure that extreme programming is going to work for your team are:

- Rapid Feedback: Use pair programming, unit testing, integration, and short iterations and releases.
- Simplicity: Try the simplest possible approach. Don't worry too much about considering cases that may or may not occur in the future.
- Incremental Change: Don't try to make big changes; try small changes that add up. This is applied to design via refactoring, planning, and team composition as well as the adoption of XP itself. Code

refactoring is a form of code modification to improve the structure of the code.

- Embracing Change: Try to preserve options for the future while actually solving your most pressing problems. Delay decisions that commit you to a path until the latest possible moment.
- Quality work: Try to create as good a product as possible. Do the best work all the time. This is assumed to be a natural tendency for most programmers, and it is encouraged by many of the practices

2.5 Software Development Tools

There are different tools used to develop software. Evaluating each option and determining which will be chosen for the project is important. Various options that are available in the market are showcased below:

2.5.1 Programming Languages

Programming languages are a set of agreed-upon languages and tools which are used to produce an output. While they can be used to do simple functions such as math or display words, they can also be used to build complicated programs that are used everyday like word processors or web browsers. An overview of some languages is provided below:

2.5.1.1 C++

C++ is one of the most popular programming languages currently used on a wide array of devices and software systems. C++ is an object-oriented programming language that was developed by Bjarne Stroustrup. C++ is an extension to the "C" programming language, due to this nature It can be coded in either way and it will still be effective. Some other highlights which make C++ so popular for system/application software, drivers, and embedded software is its ability to create classes and objects. Some other concepts within the C++ programming language include polymorphism, visual and friend functions, templates, namespaces and printers[11].

2.5.1.2 Java

Java is an object-oriented programming language that allows programmers to write code that is easily implemented on multiple types of media that can range from cell phones, to computers, as well as web-based applications. Java applets were created to be implemented with web-based applications, and Java programs

were designed to be used on mobile applications. While Java is easy to use, which is why it is simpler to use compared to other programming languages with more difficult syntax[12].

2.5.1.3 Swift

Swift is an object-oriented and general purpose programming language developed by Apple Inc. The language is typically used to develop iOS desktop and mobile applications. The language has become very popular since its release due to its intuitive approach to common programming errors. [13]

2.5.1.4 Kotlin

Kotlin is an OO programming language, which is based on Java and was first developed in 2011 by Jetbrains. It is like Java, since it can use all the libraries Java has and can use its code, so it has all of the stability and security features that such a popular and long lasting programming language has. It also runs on the Java virtual machine, so that it can run with other languages just like Java can[14][15].

2.5.2 Database Management Systems (DBMS)

A database management system (DBMS) is a piece of software used to create and maintain a database. Different DBMS' have many different features, but generally they can create a database, store it in a centralized place, show different pieces of data to different users, and allow the administrators to alter the data or how it interacts. The Structured Query Language (SQL) is used to interact with a DBMS. SQL is a computer language which is divided into two parts: the Data Definition Language (DDL) and the Data Manipulation Language (DML). DDL is used to create the structure of the database while DML is used to manage the data in the database.[16][17] An overview of some DBMS' are provided below:

2.5.2.1 Microsoft SQL Server

Microsoft SQL Server is a relational database management system, or RDBMS, that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. It's one of the three market-leading database technologies, along with Oracle Database and IBM's DB2. Like other RDBMS software, Microsoft SQL Server is built on top of SQL, a standardized programming language that database administrators and other IT professionals use to manage databases and query the data they contain. SQL Server is tied to Transact-SQL, an implementation of SQL from Microsoft that adds a set of proprietary programming extensions to the standard language[18].

2.5.2.2 Oracle

Oracle is a database that is used to store data that can later be retrieved. Oracle is one of the first database that is designed for companies to be able to use that is able to be expanded upon depending on the size of the company's needs. It is also cost efficient because companies have a whole system that allows them to have a full access to the server as well as the software that helps them access the data as well as manage the database[19].

2.5.2.3 Apache

The Apache Software Foundation offers a multitude of open-source database management systems. Each one of them specializes on different types of platforms. Apache Ignite allows the database to be stored within the system without outside contact. This is a relational database management system with support for SQL[20][21].

2.5.2.4 MySQL

MySQL is a database management system, founded by Michael Widenius in Finland in 1995. It is open source, but has been acquired by Oracle, who has tightened control and it is not as open as it once was. While MySQL isn't as strong as it once was due to it losing supporters to other DBMS, it is still a very established database system with good support available[22][23].

2.5.2.5 SQLite

SQLite is an open-source database management. It can implement databases which are compact and self-contained meaning that they are stored within a device itself. It is one of the most popular database management systems due to its wide use in mobile phones and some computer applications.[24]

2.6 The Tools that will be used in this Project

In this section we will describe in detail each tool that will used in the implementation of this project. We will discuss the chosen programming language and the Database Management System.

2.6.1 Kotlin

Kotlin is an OO programming language, which is based on Java and was first developed in 2011 by Jetbrains. It is like Java, since it can use all the libraries Java has and can use its code, so it has all of the stability and security features that such a popular and long lasting programming language has. It also runs on the Java virtual machine, so that it can run with other languages just like Java can. While there are a few minor differences, such as different variable types, it can directly convert Java code to Kotlin code and make it run, in some cases even reducing the amount of lines by a large degree and improving readability. It also has features that can save time compared to Java, such as automatically detecting types and putting them into the code, better treatment of equals(), and a better handling of nulls. It also has JavaScript support, so it has increasing support for creating web applications as well, although its implementation is not as strong as Java's. Due to the improvements over Java, Google chose it to be the new default language for Android in 2017, increasing its popularity even more than those the added features did. Since we are familiar with Java, we will use it as our programming language of choice to develop our Android application. It adds enough features to Java to help us improve our skills, while not requiring us to learn a completely new language, and we may also be able to write our normal Java code and then convert it to Kotlin[14][15].

2.6.2 SQLite

SQLite is an open-source relational database management system. The first version of the system was released on the Spring of 2000 and tt was developed D. Richard Hipp. Unlike other databases, SQLite is not a client-server database engine, instead, SQLite is directly embedded within the system. This allows the databases to be accessible only within the device they are stored in.[24]

We will use SQLite in our program due to its simplicity and free cost. SQLite will allow us to easy develop a database for our system without worrying about hosting it somewhere. Furthermore, SQLite is widely used in mobile applications so there should be plenty of tools available to implement the database. Finally, since it is free, we will be able to run the server without problems of cost.

The information included in our database will consist of the events and news provided on campus by the university, the course catalog, and a detailed campus map with any relevant information that will be useful to our college students and staff. It will also be able to have the contact information for the faculty and staff, as well as the directory, which will be useful to everyone on campus. In our database, we will also be able to show the user a transportation schedule of the different busses as well as the DART train schedule. We will

also provide the information of the different social clubs and organizations that are registered on campus. The users will also be able to access their grades and look up their schedules all in our app, with information provided through the school's database. The school's basic menu will also be stored in our database in order for the students to familiarize themselves with our school's cafeteria food.

2.6.3 Unified Modeling Language

The Unified Modeling language is an Object-Oriented modeling language that provides the elements and relationships to model software requirements and design [9]. This graphic design notation was standardized by the Object Management Group (OMG) and it is widely used in the industry[10].

This section goes over the many different UML diagrams that are commonly used for specifying defining a graphical language for visualizing, constructing, and documenting object systems. There have been several versions of UML design with previous versions deprecating older versions of UML diagrams for newer and more efficient diagrams. The current version is UML 2.5, it was published by the Object Management Group in 2015. Different diagrams have different purposes when implemented in a system. We will implement several UML diagrams throughout the course of our project. UML Structure Diagrams: show static structure of the system and its parts on different abstraction and implementation levels and how those parts are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. Structure diagrams are not utilizing time related concepts, do not show the details of dynamic behavior. However, they may show relationships to the behaviors of the classifiers exhibited in the structure diagrams[25].

- **Class diagram:** Shows structure of the designed system, subsystem of component as related classes and interface, with their features, constraints and relationships – associations, generalizations, dependencies, etc.
- **Object diagram:** Instance level class diagram which shows instance specifications of classes and interfaces (objects), slots with value specification, links (instances of association).
- **Package diagram:** Shows packages and relationships between the packages.
- **Model diagram:** UML diagram auxiliary structure which shows some abstractions or specific view of a system, to describe architectural, logical or behavioral aspects of the system. It could show, for example, architecture of a multi-layered (aka multi-tiered) application.

- **Composite structure diagram:** Diagram could be used to show: Internal structure of a classifier A behavioral of a collaboration
- **Internal structure diagram:** Shows internal structure of a classifier – a decomposition of the classifier into its properties, parts and relationships.
- **Collaboration use diagram:** Shows objects in a system cooperating with each other to produce some behavior of the system.
- **Component diagram:** Shows components and dependencies between them. This type of diagrams is used for Component-Based Development (CBD), to describe systems with Service-Oriented Architecture (SOA).
- **Manifestation diagram:** While component diagrams show components and relationships between components and classifiers, and deployment diagrams – deployments of artifacts to deployment targets, some missing intermediate diagram is manifestation diagram to be used to show manifestation (implementation) of components by artifacts and internal structure of artifacts.
- **Deployment diagram:** Show architecture of the system as a deployment (distribution) of software artifacts to deployment targets. Specifications level deployment diagram (also called type level) shows some overview of deployment of artifacts to deployment targets, without referencing specific instances of artifacts or nodes. Instance level deployment diagram shows deployment of instances of artifacts to specific instances of deployment targets. It could be used for example to show differences in deployments to development, staging or production environments with the names/ids of specific build or deployment servers or devices.
- **Network architecture diagram:** Deployment diagrams could be used to show logical or physical network architecture of the system. This kind of deployment diagrams could be called network architecture diagrams.
- **Profile diagram:** Auxiliary UML diagram which allows to define custom stereotypes, tagged values, and constraints as a lightweight extension mechanism to the UML standard. Profiles allow to adapt the UML metamodel for different: Platforms (such as J2EE or .NET) or Domains (such as real-time or business process modeling.
- **Behavioral Diagrams:** show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

- **Use case Diagram:** Describes a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors) to provide some observable and valuable results to the actors or other stakeholders of the system(s).
- **Information flow diagram:** Shows exchange of information between system entities at some high levels of abstraction. Information flows may be useful to describe circulation of information through a system by representing aspects of models not yet fully specified or with less details.
- **Activity diagram:** Shows sequence and conditions for coordinating lower-level behaviors, rather than which classifiers own those behaviors. These are commonly called control flow and object flow models.
- **State machine diagram:** Used for modeling discrete behavior through finite state transitions. In addition to expressing the behavior of a part of the system, state machines can also be used to express the usage protocol of part of a system. These two kinds of state machines are referred to as behavioral state machines and protocol state machines.
- **Behavioral state machine diagram:** Shows discrete behavior of a part of designed system through finite state transitions.
- **Protocol state machine diagram:** Shows usage protocol or a lifecycle of some classifier, e.g. which operations of the classifier may be called in each state of the classifier, under which specific conditions, and satisfying some optional postconditions after the classifier transitions to a target state.
- **Interaction diagram:** Interaction diagrams include several different types of diagrams: Sequence diagrams.
- **Communication diagrams (Known as Collaboration diagrams in UML 1.X).** Timing diagrams. Interaction overview diagrams.
- **Sequence diagram:** Most common kind of interaction diagrams which focuses on the message interchange between lifelines (objects).
- **Communication diagram (Also known as Collaboration diagram):** Focuses on the interaction between lifelines where the architecture of the internal structure and how this corresponds with the message passing is central. The sequencing of messages is given through a sequence numbering scheme.
- **Timing diagram:** Shows interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among lifelines along a linear time axis.

- **Interaction overview diagram:** Defines interactions through a variant of activity diagrams in a way that promotes overview of the control flow. Interaction overview diagrams focus on the overview of the flow of control where the nodes are interactions or interactive uses. The lifelines and the messages do not appear at this overview level.

2.7 Summary

This chapter introduced related work similar to the system we plan to develop. The different approaches to the software development process were discussed and XP programming was determined to be the best choice for the project. Several programming languages were discusses and Kotlin was determined to be the best option. Various Database Management Systems were listed and described but ultimately MySQL was chosen. Finally, the chapter described the Unified Modeling Language. This modeling language will be used to model important aspects of the project.

Chapter 3

Requirement Specification and System Modeling

Objectives:

- Explain the requirements engineering process.
- Define the user and system requirements.
- Introduce use-case diagrams.

3.1 Introduction

This chapter introduces the requirements engineering process. The engineering process has certain steps that a software developer must go through when creating an application. The steps include: Elicitation, Analysis, and Documentation. This chapter includes the User and System requirements. The diagrams that showcase the different activities and options of our application are also found on this chapter. Since we are the only party involved in creating the software, there is no client to set the requirements, and as such, we are heavily involved in the requirement creation processes.

3.2 Requirements Engineering Process

Requirements engineering is the process of gathering, documenting and understanding the requirements for the system. The requirements serve as a guide as to how and what the system needs to achieve in order for the product to be considered successful. This process is divided into three parts which are elicitation, documentation and analysis. These steps are crucial as they serve to form the outline and structure of what the final product is going to produce. Each requirement can be presented as a user or a system requirement. The user requirements present a general descriptioni of the requirement while the system requirement give a thorough description with plenty of detail. Throughout this chapter, we took the different steps, and explored how they relate to our system. Each of these steps are presented and describe in detail in the following sections.

3.2.1 Elicitation

Elicitation is one of the first steps in the requirement engineering process where we are first obtaining the requirements necessary for the project that our team is currently working on. Normally, one would gain such requirements through client interviews, brainstorming with the team, and through meetings with the clients. By asking open ended questions, we can determine what parameters we would need to establish before beginning on the project and moving on to the next step in the process. The team would also be expected to have brainstorming meetings to be able to induce ideas that would be used to guide the creation and facilitation of the software being produced. The ideas need to be documented in order for everyone to be able to see and look back and to build upon. The different requirements that will be introduced in this step are the normal requirements that are proposed by the client that may be added to the software once all

of the expected requirements are met. The expected requirements are those that are expected by the client to be implemented in the program, and the normal requirements are those that are not required to be in the program but they can be if time allows them to be implemented. Once all of the requirements are gathered and are finalized, they are made into more formal and detailed requirements.

3.2.2 Analysis

After the requirements have been documented, the team would organize the requirements to understand which of the requirements have a higher priority than the rest. Within the analysis step, there is the Categorizing and Prioritizing steps to ensure that we are grouping and separating the requirements to match the different stages of the software's development. When we categorize the requirements, we make sure that we group them into different categories so the team can be able to understand where each of the requirements will be utilized and where they need to be implemented. When we prioritize the requirements in this stage, we ensure that the requirements with the highest importance are done first, and the rest of the requirements are done as the project development progresses. Most of the non-critical requirements are not always implemented in the finalized products due to the different restraints that one is given, whether it is due to time or due to a budget.

3.2.3 Requirements Definition and Documentation

In this section, we define and document each step of the requirements. The requirements definition is the process of adding further information to the given requirements for the development team to be able to understand the requirements using more technical language instead of informal language. In the documentation phase, the team must document all related instructions, any documentation that is used to keep the software updated as well as the documents that will be used by the users in case they need some help with the software. In the documentation stage we must also represent our ideas to the client using different forms of visual representation methods such as the unified modeling language diagrams. We can use two different types of visualization techniques for visualizing the user interface. The two ways to display the user interface could be by using low fidelity and high-fidelity prototypes of the interface. The low fidelity models are simple representations that will be used in rough drafts of what the user interface will look like, as opposed to the high-fidelity models that are professional quality to display exactly what the final version of the user interface will appear to be. Once the prototypes are completed, they are shown to the client for approval.

3.2.4 Requirements Specification and Requirements Agreement

By having the prototypes approved by the client and having the documentations of the finalized requirements looked over, a new finalized requirements document is created. The last document related to the requirements is called the Software Requirements Specification (SRS), and it is what the whole design team involved with the software will use as a reference guide. The SRS documents are given to everyone involved in the creation and implementation of the software. The document's purpose is to explain what the software is, the software's goal, the requirements and constraints that the software must meet, and any other documentation that demonstrate what the user interface will be. Some of the types of recipients of the SRS documents could me the managers, stock holders, engineers, and any other relevant party that may need these formal documents. The last step is the requirements agreement, and it is necessary because it is the final agreed upon version of the software plans that the client and the design teams approve upon and therefore becomes a contractual obligation to produce a finalized product for the client based on the approved time constraints and budget constraints.

3.3 Requirements Documentation

The requirements for the software system are presented in this section. The user requirements identify the general requirements of the system while the system requirements give additional detail for their implementation.

3.3.1 User Requirements

1. The system shall allow the user to be able to login or enter as a guest.
2. The system shall allow the user to be able to use a username and password to access some features.
3. The system should be able to allow the users to customize their profiles by listing their preferences.
4. The system shall allow the user to access information about their current classes.
5. The system should be able to allow the users to enroll/drop classes currently being offered in a given semester.
6. The system should be able to obtain information regarding assignments from their current classes.

7. The system shall allow the users to be able to access information regarding organizations within the school.
8. The system shall allow the users to be able to redirect them to another application (ex. Blackboard).
9. The system should be able to allow the user to be able to view the school's social media accounts.
10. The system should be able to let the user access to a calendar which highlights the upcoming events and important dates.
11. The system should be able to access information regarding the school's library.
12. The system should be able to access the inbox of their school's email within the application.
13. The system should allow the users to access a detailed campus map.
14. The system should send the user notifications on special occasions.
15. The system shall present the user with the school's menu offered by the cafeteria.
16. The system shall provide all the contact information for the campus emergency services and emergency hot lines.
17. The system should provide the user with a detailed schedule of the local bus and the DART train station.

3.3.2 System Requirements

1. When the user first opens the mobile application, the user will be greeted by a welcome page where it will ask the user to log into their account if they have one. If they do not, they can create an account within the application where they can choose a username of their liking and a password. Or they can skip the log in process altogether and open the application as guests. Illustrated in 3.1
2. If the user is logged in as a guest only certain features will be available to them. If the user wants to unlock all the features they must be logged into their account or create one if they have not done so.
3. The user will be able to tailor their account to their liking. Such as adding a profile picture, study plan, interests, hobbies, and organizations they will be interested in joining or are a part of.

4. The user will be able to log into their blackboard account and have the ability to view what classes the user is enrolled in or has previously taken in the previous semesters. The user will also be able to see the notifications, assignments due, and grades that they have received in the classes they are currently enrolled in.

5. If the user needs the ability to enroll in new classes for the future semester or drop classes that they are currently enrolled in the user will be able to click on a link and they will be taken to the "my.unt.edu" website where they can drop or add classes as the user needs.

6. With the application being able to see what classes the user is enrolled in; the user will also have the option to see what assignments are due for their classes as well as notifications sent out by the professor. The user will also be able to see future assignments that will need to be completed in future dates as well.

7. The mobile application will also give the user the options of seeing what organization are currently active and have the ability to see the active members in the organization. The system will also provide the user with a summary of information for what the organization stands for, and what events the organization has taken a part of.

8. The system will give the user the options of being redirected to other websites that are important to them such as the ability to be redirected to blackboard.edu for example to turn in assignments, or to access their student email accounts.

9. With schools interacting with the student body through social media, the system will also be able to provide the schools social media accounts username and links to the respective website. Once the user clicks on the link of a social media account (Instagram for example), if the user has the Instagram application on their mobile devise it will open up on the schools account page.

10. The user should be able to choose the calendar, either as its own separate tab or as an option found within the menus. When the user accesses it, it should default to the monthly view so that the user can see an overview of the important dates, although other views can possibly be added that the user can change it to one they are familiar with. The user should be able to select certain dates and see which events are available on that day. A feature could possibly be added to where if they are part of an organization, those events will be visible to that user but not to other users. While it seems like it

would be a good idea to add the ability to add reminders or your own events, this feature can already be implemented by other calendar apps. Illustrated in 3.2

11. The application should be able to access the features available with the library. It would mainly have the ability to view info about the library such as hours, their blog, and events with the library. Additional features that would be good to add would be the ability to check out books from the application, access UNT and UNT Dallas databases in order to read online books, and reserve study rooms at certain available times on the application.

12. This should be a simple redirect to the Outlook application or accessing the Webmail features from the web browser, either redirecting to the actual web browser or a built-in web browser. The user will have to login by themselves, since our application will not originally have access to the database that contains student log-in information.

13. The system could either use Google maps, or an interactive version of an image of the map of the campus. Using either option, the user could move throughout the map, and tap on certain points of interest, such as each building and DART station. They could get information about each point of interest as well as pictures, and if Google maps is used, we could add in street view or photo sphere options so there can be more interactivity with different points of interest. Finally, there could also be floor plan images of the inside of each building so that students can have a map if they get lost and don't know where to go.

14. The application should be able to interact with the Android notification service system in order to send notifications about certain things. Not all features of this application, such as blackboard or E-mail, will be able to have notifications, but ones that we can control should be able to have them, such as events or organizational information.

15. The system should have the ability to access the menu items and prices available at the cafeteria. If they tap on the items, they could possibly access a picture or nutritional information, although at the minimum it should have a scrollable list for the menu items available each day. Illustrated in 3.3

16. This should be able to have a list of all available emergency services, complete with phone numbers and email if necessary. It should at least list them, and if the user taps on them, it will redirect them to the phone dialer or email application so that they are able to contact the emergency services if necessary. This section will also describe the different services provided so they know who to contact, mainly

campus police. The GPS location of the phone could also be used if a feature could be implemented similar to the blue emergency stations around campus, so the police may have an exact location of the user.

17. This feature would preferably have its own tab, since it is an important part of the campus. The first part of this feature should have a connection to Google maps, where it can show a DART system map which will show the possible paths the train could take. It could also do the same for the buses, although only nearby destinations should be necessary for the bus stops, since the train can reach farther than the buses. The next part of this feature would have the schedules for the buses and trains. For the trains, since the application is for our campus, it should only require the UNT Dallas station schedule. Since ours is at the end of the blue line, it should only list departure times. For other stations, they will need the DART application. The bus schedules can be expanded, since bus stops nearby can be included since they are relevant to the campus. It will include their schedules and possibly their paths. Finally, the feature can possibly be expanded if the campus grows and has its own shuttle system expanded past the current golf carts.

3.3.3 Use Cases

Some user requirements have been modeled and turned into use-case diagrams. The following are the use-case diagrams chosen:

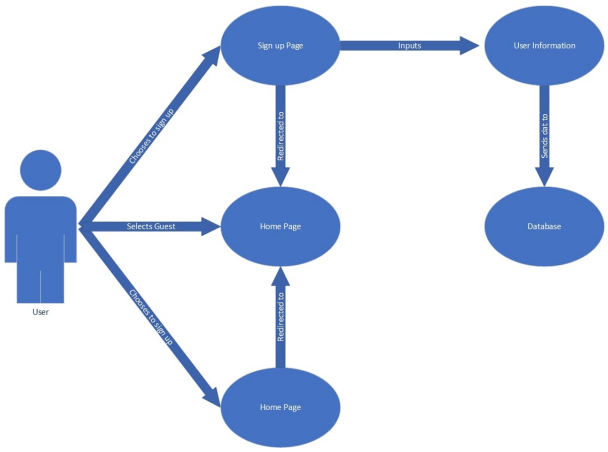

Figure 3.1: This use-case shows how the user is able to access to the system.

This use-case shows how the user is able to access to the system. When the user starts the system they are required to enter through logging in, signing up or being a guest. Signing up prompts the user to

complete the sign up process. Logging in will only require users to provide their log in information. Finally, accessing the system as a guest will not require the user to provide any information but will make some features unaccessible.

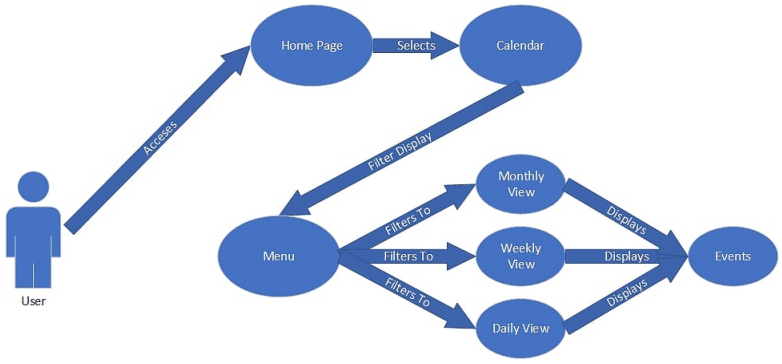

Figure 3.2: This use-case shows how the user could access the events feature.

This use-case shows how the user could access the events feature. Accessing events through the home page will prompt the user to select a filter either monthly, weekly and daily. Selecting the monthly option will display all event for the current month. Selecting the weekly option will display all events for the current week. Selecting the daily view will display all the events happenning in the current day.

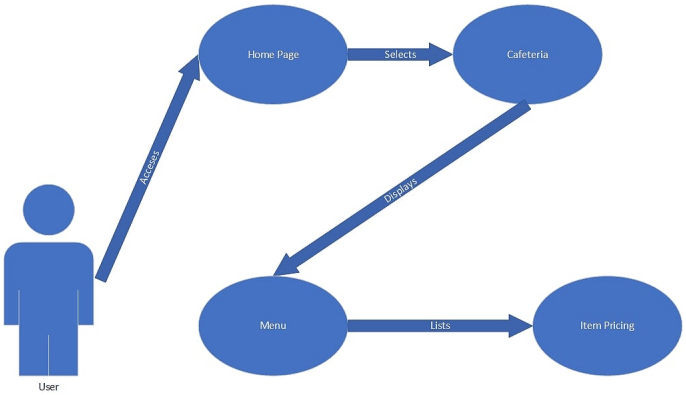

Figure 3.3: This use-case shows how the user can view an item's pricing through the cafeteria's menu feature.

This use-case shows how the user can view an item's pricing through the cafeteria's menu feature. The user accesses the cafeteria feature through the home page which displays the menu of the school for the current week. The items are displayed in a list along with their pricing.

3.4 Summary

In this chapter, we presented the system's requirements. Our proposed system includes user and system requirements. We detailed our process for gathering and analyzing the requirements and went over the specific requirements that were important to us and our users. Finally, we presented a few UML use-case diagrams in order to detail how some of the requirements will work and how the system will flow based on what users choose. The next chapter will cover our design of our proposed system.

Chapter 4

Design

Objectives:

- Define the Model-View Controller architecture.
- Present the design of our proposed system.
- Explain the database design for our proposed system.
- Present the graphical user interfaces of our proposed system.

4.1 Introduction

This chapter introduces the initial design of our proposed system. The chapter also defines the Model-View Controller architecture. The graphical user interfaces of the system will be based on this model. The initial design for the main page of our proposed system is showcased and explained. This inital design will serve as a guide for the implementation of our system. The initial design for every GUI in the system is presented and explained.

4.2 Model-View Controller architecture (MVC)

There are several components that must be planned for and completed when developing our proposed system. One of these components are the graphical user interfaces (GUIs). These interfaces willl allow our user to easily interact with the system. Deciding on the architecture that our GUIs will follow is important as it will affect the quality of our proposed system. For the design of our system, we will use the Model View Controller, or MVC, architecture. This method of design is mainly used to organize programs with different views of data. It can split a program into different views of data, with their own controller class, unless the compiler has its own controller, and can keep the views separate from one another. This is very useful, because it allows for the program to display or hide components to a user when necessary, as well as potentially informing a user when the data changes. It can also be combined with other architectures like client-server, where data can be stored on the server and the different views will be shown on the client side. The first part of this section will be about the model, which will show how the data will be used in the program via our database models. The second part will be about the view, to explain what the user will see when interacting with the program and the GUI design. The last part will talk about the Controller function which will explain how the program will work. Each of these elements of the architecture will be further explained within this section. [9][10].

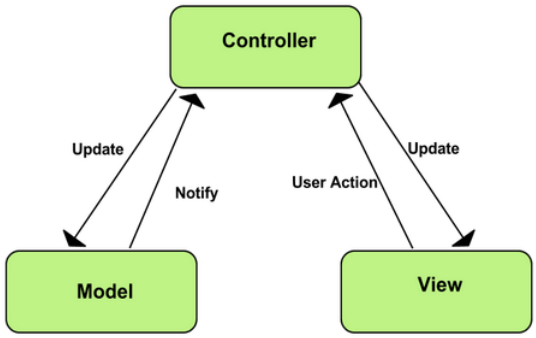

Figure 4.1: An illustration of the Model View Controller architecture.

4.2.1 Model

The model component of the MVC architecture defines the raw portion of an application and describes and defines the structure of the application and the information within. There may be many tasks in an application to be performed, so the model will define various portions of the internal schema for the application. It may define the data structures within the application, the different methods of actions the user can perform, or the general structure. The model interacts with the view by providing the structure that the user will use to interact with the application, and it interacts with the controller by allowing it to manipulate the data it stores. However, the model at times can also be considered independent from the view and controller, since data can exist in a vacuum without having a controller operating on it, and without views being used to look at it.[26][9]

4.2.2 View

The view component of the MVC architecture defines the portions that the users will see when using an application, and how the user will interact with the application. While the view does not actually control what the user does, as this is done by the controller, the view will instead provide a framework so that the model and controller can perform their tasks. There can be many views for the user, since as they navigate through the application, there will be many points where they may need to enter information or control the application, so multiple views must be provided for the user for them to use the application properly. The view interacts with the model by providing a gateway into the internal structure of the application, such as providing a framework for the user to enter information and once that information is entered, the model defines how that will be stored. The view interacts with the controller with this framework by allowing it to

manipulate the data as necessary, such as sending it somewhere or performing calculations on it. Views can be independent of the model and controller, since it may exist but will not operate on anything, and can sometimes be the controller, in cases such as that the view is used to look at the data but not truly operate on it. [27][9]

4.2.3 Controller

The controller component of the MVC architecture defines the way the commands from the user are used as inputs and converts that data for the model or the view to use. The controller is responsible for interpreting the actions that the user does, ranging from button clicks to what the user interacts with and it sends the information gathered to the appropriate source. The controller is the link that responds to the events that involves those actions taken by the user and notifies the user's actions to the model, which can change the state of the model. A new view then is generated based on the actions taken from the data model. The view then waits again for the next action that the user will complete in order for the same steps to be taken, which will repeat as long as there is interaction from the user.[28]

4.3 Our proposed system design

This section presents each element of the Model-View Controller architecture as it relates to our proposed system. A database will serve as the model for our system and will define the information within our application. The graphical user interfaces presented will serve as both the view and controller due to the design and code being tied together.

4.3.1 Database

In this section we present the database of our proposed system. The database has been designed in order to serve as the model for the system. As stated previously, the model defines the information within the application and its structure. We provide a wholistic view of our program, as well as the different types of relationships that our tables have with each other.

4.3.1.1 Logical Model

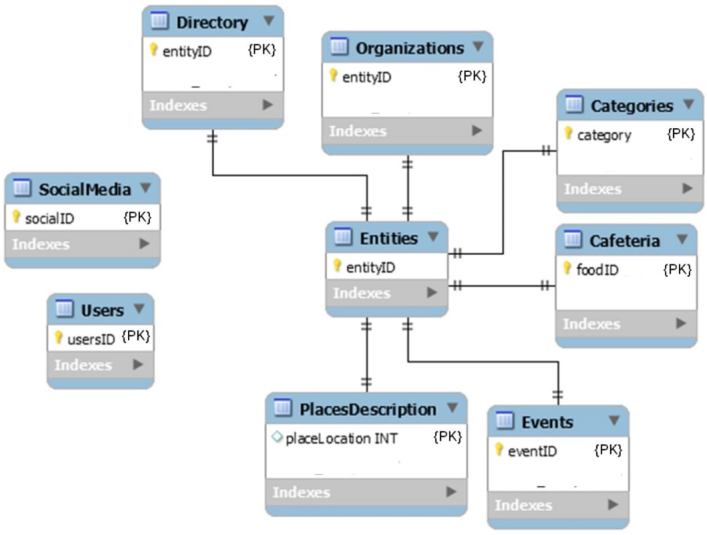

Figure 4.2: An illustration of the physical model for our system.

Figure 4.2 showcases the initial design of our database. This model is the logical model of our database. In this model the initial tables and their relationships are stated.

4.3.1.2 Physical Model

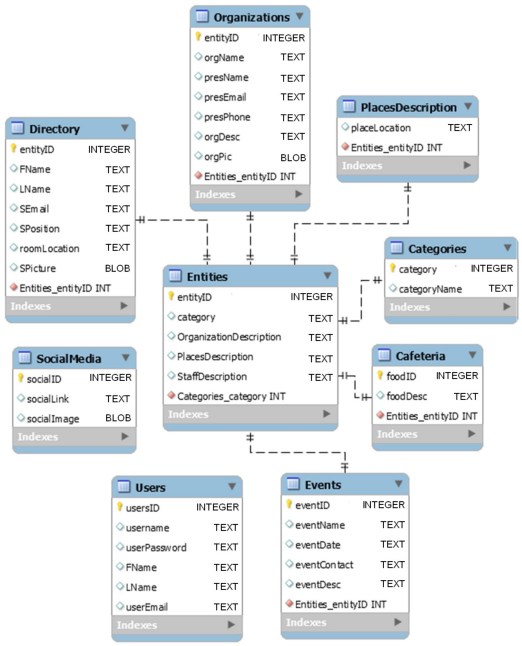

Figure 4.3: An illustration of the physical model for our system.

The finalized design of our database is showcased in figure 4.3. This is the physical model of our database, this model takes into consideration the datatypes offered by SQLite. In this model we can see how each of the tables connect and interact with each other and the different types of variables involved in each of the tables.

4.3.1.3 Database Description

The Users table will store information about the user's account. A unique account will be created in order to store information related to a user. UsersID stores the unique ID of the user and is of type INTEGER. Username will store the username the user chooses and is of type TEXT. UserPassword will store the password the user chooses and is of type TEXT. FName will store the user's first name and is of type TEXT. LName will store the user's last name and is of type TEXT. UserEmail will store the user's email address and is of type TEXT.

The SocialMedia table will store information about social media accounts. SocialID stores the unique ID of the social media account and is of type INTEGER. SocialLink will store a link to the social media page

and is of type TEXT. SocialImage includes a picture of the social media service's icon and is of type blob.

The entities table is used in the database to differentiate the categories of different features. EntityID stores the unique ID for the entities and is of type INTEGER. Category stores the type of category the entity will belong to, such as organization, place, or staff, and is of type INTEGER. OrganizationDescription stores the description about any organizations stored in the table and is of type TEXT. PlacesDescription stores the description about any places stored in the table and is of type TEXT. StaffDescription stores the description about any staff members stored in the table and is of type TEXT.

The events table will be used to store information about the events that may take place on campus. EventID stores the unique ID for each event and is of type INTEGER. EventName stores the name of the event and is of type TEXT. EventDate will store the date that an event will take place and is of type TEXT. EventContact stores the name of the contact person who runs the event and is of type TEXT. EventDesc includes the description about the event and is of type TEXT. This table is connected to the entities table with a one to one relationship, with the foreign key being entityID.

The cafeteria table is used to store information about the menu items that will be available in the cafeteria. FoodType stores the name of the food item available and is of type TEXT. FoodID is the unique ID given to each food item and is of type INTEGER. FoodDesc includes a description about each food item and is of type TEXT. FoodPic includes a picture to show what the food item looks like and is of type blob. This table is connected to the entities table with a one to one relationship, with the foreign key being entityID. The Categories table includes information about the categories that are used in the events table. Category stores the type of category the entity will belong to, such as organization, place, or staff, and is of type INTEGER. CategoryName stores the name of the category indicated by the INTEGER value of category and is of type TEXT. This table is connected to the entities table with a one to many relationship, with the foreign key being entityID.

The OrganizationDescription table will be used to store information about the organizations on campus. OrgName stores the name of the organization and is of type TEXT. PresName will store the organization's president's first name and is of type TEXT. PresEmail stores the email of the president of the organization and is of type TEXT. PresPhone stores the phone number of the president of the organization and is of type TEXT. OrgDescription includes a description about the organization and is of type TEXT. OrgPicture includes a picture of the organization's logo and is of type blob. This table is connected to the entities table with a one to one relationship, with the foreign key being entityID.

The StaffDescription table will store information about the staff. SFName will store the staff member's

first name and is of type TEXT. SLName will store the staff member's last name and is of type TEXT. RoomLocation stores the room number for the staff member and is of type TEXT. SEmail will store the staff member's email address and is of type TEXT. SPosition stores the position of the staff member and is of type TEXT. SPhone stores the phone number of the staff member and is of type TEXT. This table is connected to the entities table with a one to one relationship, with the foreign key being entityID.

The PlacesDescription table includes information about the location of different places. PlaceLocation stores the location of different place entities and is of type TEXT. This table is connected to the entities table with a one to one relationship, with the foreign key being entityID.

4.3.2 Graphical User Interfaces

In this section we present the graphical user interfaces for our proposed system. As stated previously, the view will define what the user is able to see while the controller will determine how each of those elements interacts with the system. The graphical user interfaces presented address the view and controller due to the code and design being tied together.

4.3.2.1 Home Page

An illustration of the main page of our proposed system is showcased in the figure 4.4. This will be the initial design for our proposed system. The home page is expected to offer twelve features. The features are layed out in a grid layout and are given a unique icon. The user is able to utilize a specific feature by selecting the icon. This will then lead the user to the interface for that feature. Each feature will provide its own interface for interacting with the user so the main page serves strictly to present the user with the features. Exiting any of the features will redirect the user to the main page.

Figure 4.4: An illustration of the main page of our proposed system.

4.3.2.2 Features

Figure 4.5: An illustration of the sign in and sign up page of our system.

As show in the figure 4.5 The user will be able to access the app in several ways with or without having a system account. When the user opens the app for the first time, the user will be greeted with several options. They will have the option to go ahead and just use the app as a guest without having to input any

personal information, and they will have access to basic functions in the app such as looking at the directory and seeing the cafeteria menu. However, if the user wants to do more advanced things such as looking at their grades, notifications, or creating event reminders in the app they will have to sign up and create an account. When the user first tries to sign-up, they will have to select what kind of user they are either a "Student" or a "Professor". Once they select their option, they will have to input their personal information such as Student ID, Student Email, Name, and password. Once they have finished with the account creating process, the user will have all the options and features unlocked.

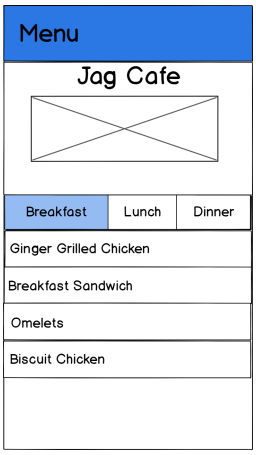

Figure 4.6: An illustration of the lunch menu feature of our system.

In figure 4.6 depicts the options that the user will have once they have selected the "Cafeteria" icon. On the top portion of the screen the user will have 3 options to choose from: Breakfast, Lunch, and Dinner. The default choice will be Breakfast, which the user can change the option depending on the time of day. The user will be able to see the items being serve at different times of the day. Each item will be given a name and listed.

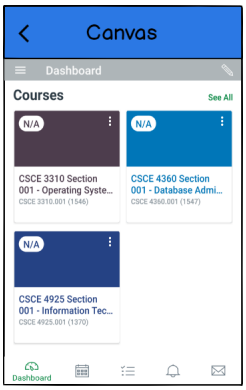

Figure 4.7: An illustration of the Canvas feature of our system.

In the following figure 4.7, the user will have the option of selecting the “Canvas” icon. The main purpose of this option will for the user to be able to access their course information such as grades, announcements, and other canvas related features. The user will be taken to a web-based page to canvas in which they can access all that information within the UNTD application.

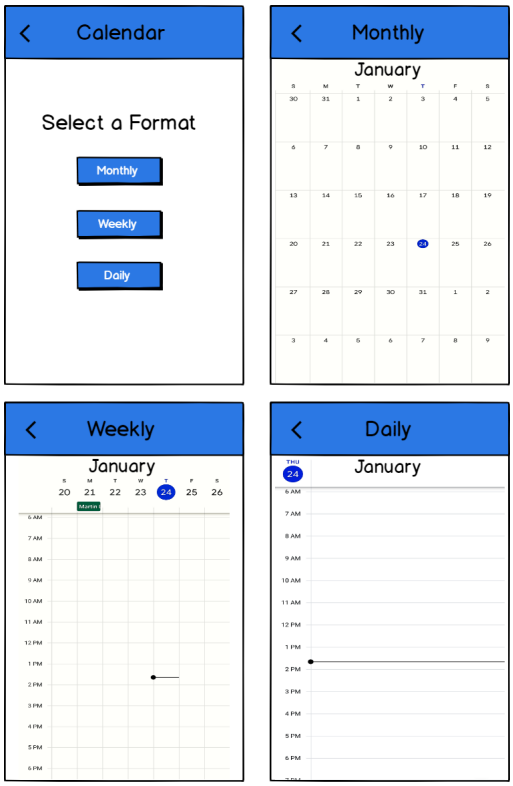

Figure 4.8: An illustration of the calendar feature of our system.

The figure 4.8 displays the calendar feature, in which when the user taps on the calendar icon they will have three viewing options on displaying the calendar. The options will be Monthly, Weekly, and daily options. Once the user selects the view they would prefer to see, the app will open that option and display it in the format the user chose. In that view the user will be able to see all the different events that are going to be happening in campus, if the user is in some organization, they will also have the ability to see events from that organization. The user will also be able to create notifications/reminders within the calendar view if they would like to be reminded of an event or need a reminder for a class.

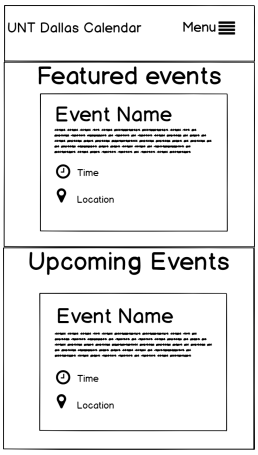

Figure 4.9: An illustration of the events feature of our system.

As shown in Figure 4.9, the events feature will display a list of events happening on campus. Each event listing will have the name and date of the event alongside the option to get more details. The list will be organized based on date, the events closest to the current date will be displayed on top. This page also displays detailed information about a given event. The location, information regarding the organization leading the event and a short description of the event are presented.

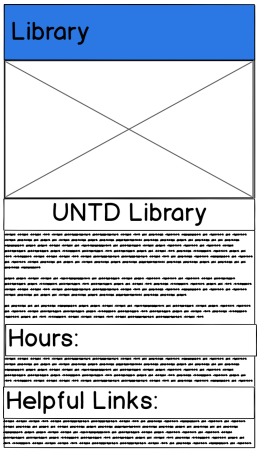

Figure 4.10: An illustration of the library feature in our system.

After selecting the library feature on the home page users will be lead to the page shown in Figure 4.10. This page showcases information related to the campus' library such as its address and operating hours. The page will also display helpful links related to the library.

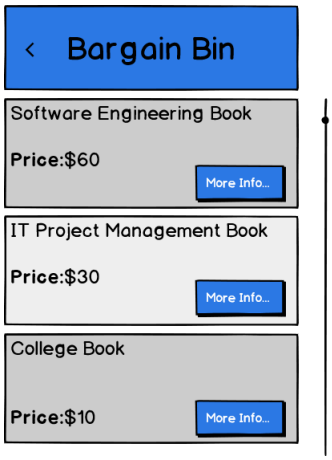

Figure 4.11: An illustration of the Bargain bin feature in our system.

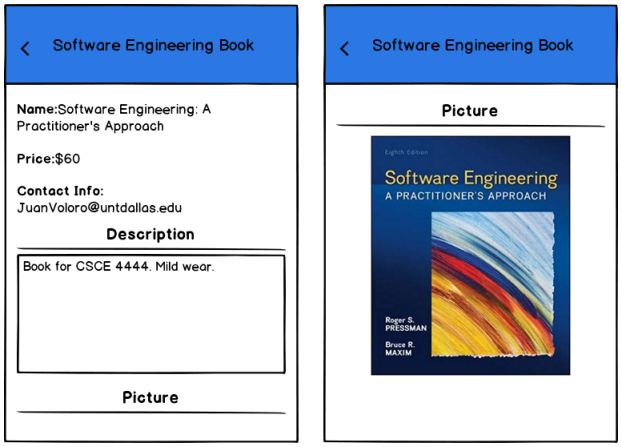

Figure 4.12: An illustration of the details of a selected item on the Bargain bin feature.

As shown in Figure 4.11 the user will be taken to a list of items when they select the Bargain bin feature. This feature will focus on the buying and selling of items between students. Each time a student posts an item for sale it will be added to this list. The item will be presented with a title and its corresponding price. If the user wants to know more about an item they can tap on the "More info" button which will take them to an interface similar to the one in Figure 4.12. This interface will present further details regarding the selected item. Information such as its title, price, contact information of the seller, a short description of the item and a picture of the item.

Figure 4.13: An illustration of the Social Media feature of our system.

Figure 4.13 showcases the interface that will appear when the user selects the Social Media feature. This feature provides a compilation of the campus' social media. Each social media platform is given an icon and a name. The user will be able to tap on the desired social media platform and then be taken to the appropriate site.

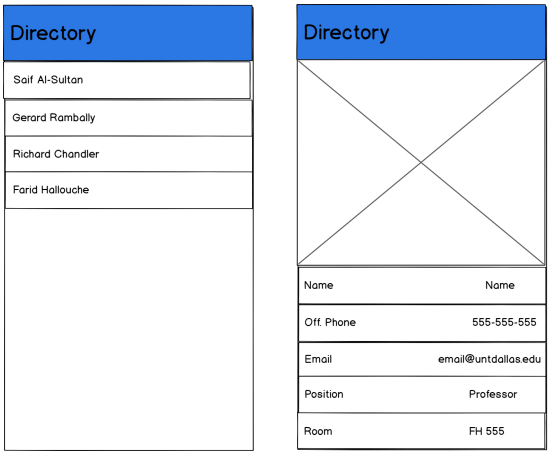

Figure 4.14: An illustration of the Directory feature of our system.

In figure 4.14, the user will have the ability to view the campus directory. There will be a scrollable list of the different faculty and other administrative workers at the campus. The main list will have their name for the users to view at a glance. Upon selecting a person another window will appear showcasing important information such as contact information, office number and position.

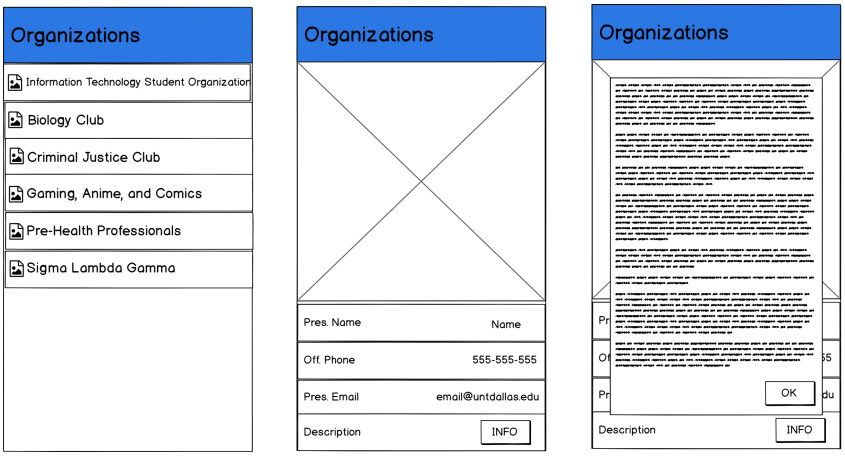

Figure 4.15: An illustration of the Organizations feature of our system.

In figure 4.15, the user will be able to view the list of organizations on campus. There will be a scrollable list which will display all the organizations on campus. Upon selecting an organization a new window will be create which will include the club President's contact info, a longer description, and any other info that will be relevant. This new window will have an image related to the purpose of the organization.

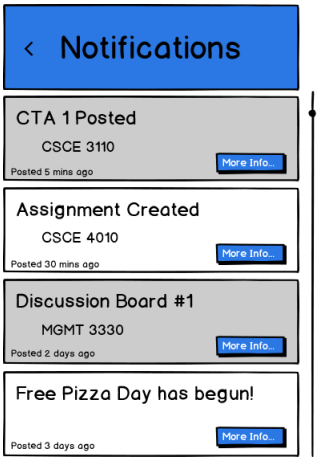

Figure 4.16: An illustration of the Notifications feature of our system.

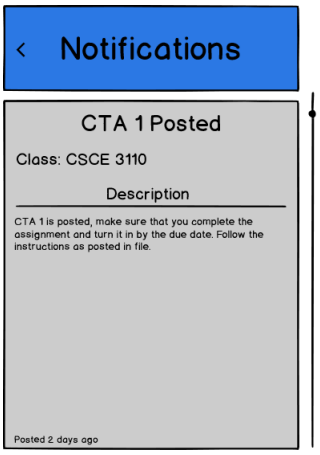

Figure 4.17: An illustration of the details of a selected notifications on the Notifications feature.

In figure 4.16 the image depicts how the Notification screen will look like for the users. The user will be able to see the different types of notifications that will be made available to them depending on the settings that they have it to be. For example, the user will receive different notifications from things that are posted on the app. The name of the notification will appear as well as a short description on where it came from. If the user wants to see more, they will need to click the more info button. Once they click that button, they will see the main title of the post and a description of the notification. The time that the event was posted will also be shown.

Figure 4.18: An illustration of the Maps feature of our system.

In figure 4.18 the user will be able to see the campus in different views. They will be able to choose between buildings and floors then depending on what they choose, the corresponding images will appear. The images will be that of the inside of the building with each section outlined. Room numbers and important

locations will be labeled similar to a blueprint.

4.3.3 Summary

This chapter presented the complete design of our proposed system. The Model View Controller architecture was introduced and explained alongside the role it played in our designs. We have also presented the design for our database with the logical and physical model. We demonstrated what our GUIs will look like from the point of view of the user. As we showed our pictures, we also described the functions and the details surrounding the GUIs. The next section will presented the implementation of the proposed system.

Chapter 5

Implementation, Testing and Evaluation

Objectives:

- Showcase the implementation of the proposed system.
- Provide the implemented graphical user interfaces for each feature.
- Provide the code behind each feature.
- Present testing scenarios to evaluate the system's performance.

5.1 Introduction

This chapter will introduce the implementation of the application. The graphical user interfaces for each feature within the application are presented alongside a brief explanation of each. Several testing scenarios are showcased which help to ensure the functionality of the system.

5.2 Implementation

This section will showcase the graphical user interfaces of the system. These graphical user interfaces represent the final version of the feature implemented.

Home Page:

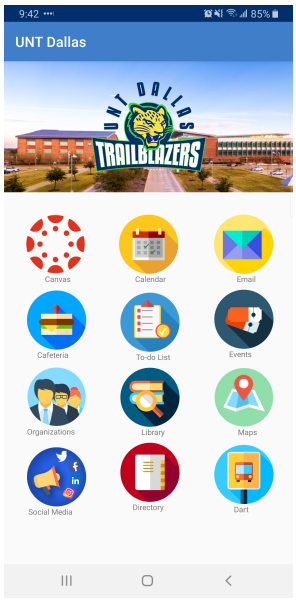

Figure 5.1: A picture of the Home Page of the system.

Figure 5.1 showcases the implementation of the Home Page of the system. This page allows users to select any of the features offered. The code behind this page's implementation can be found on Appendix A.1

Cafeteria:

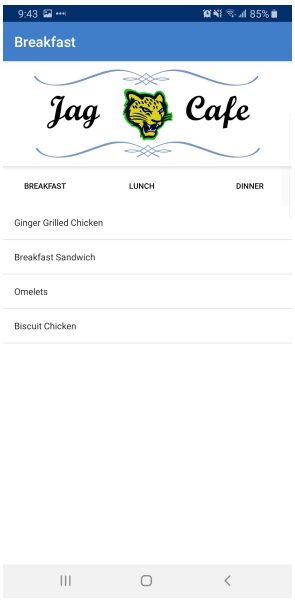

Figure 5.2: A picture of the Cafeteria feature of the system.

Figure 5.2 showcases the implementation of the Cafeteria feature of the system. This page allows users preview the items being sold at the campus' cafeteria. The code behind this page's implementation can be found on Appendix A.2

Organizations:

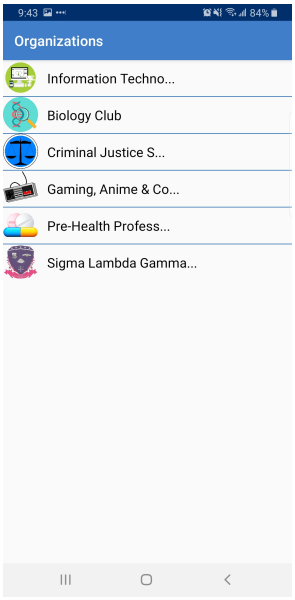

Figure 5.3: A picture of the Organizations feature of the system.

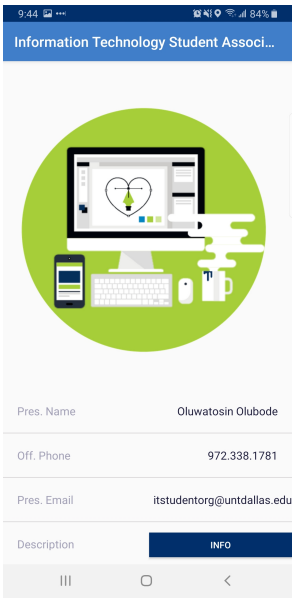

Figure 5.4: A picture showcasing detailed information within the Organizations feature of the system.

Figure 5.3 showcases the implementation of the Organization feature of the system. This page allows users to access information about organizations and clubs within the campus. Selecting an item within the list will display figure 5.4. The code behind this page's implementation can be found on Appendix A.3

Library:

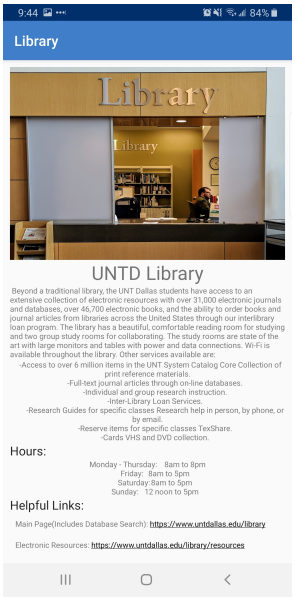

Figure 5.5: A picture of the Library feature of the system.

Figure 5.5 showcases the implementation of the Library feature of the system. This page allows users to access information about the library within the campus. Useful links and general information related to the library are presented within this feature. The code behind this page's implementation can be found on Appendix A.4

Maps:

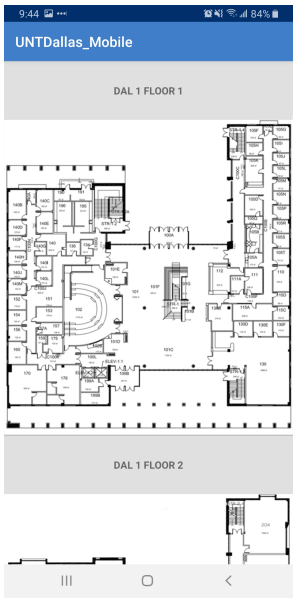

Figure 5.6: A picture of the Library feature of the system.

Figure 5.6 showcases the implementation of the Maps feature of the system. This page allows users to access detailed maps about the campus. The layout of buildings and their floors can be found within this feature. The code behind this page's implementation can be found on Appendix A.5

Social Media:

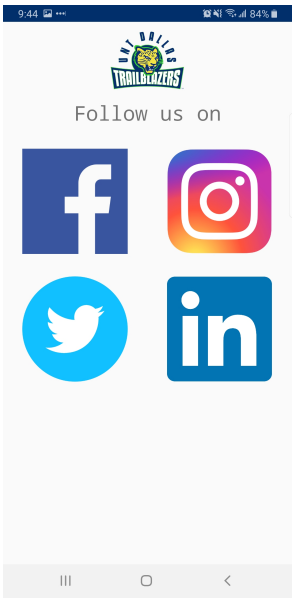

Figure 5.7: A picture of the Social Media feature of the system.

Figure 5.7 showcases the implementation of the Social Media feature of the system. This page allows users to access the social media accounts associated within the campus. The feature offers access to the school's Twitter, Facebook, Instagram and LinkedIn. The code behind this page's implementation can be found on Appendix A.6

Directory:

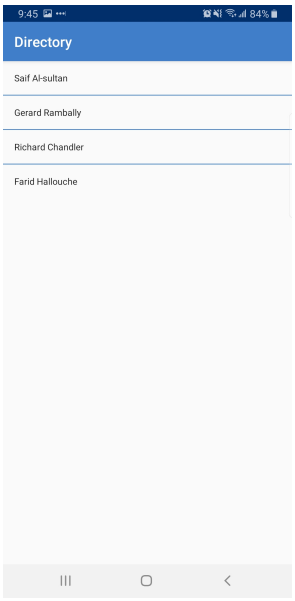

Figure 5.8: A picture of the Directory feature of the system.

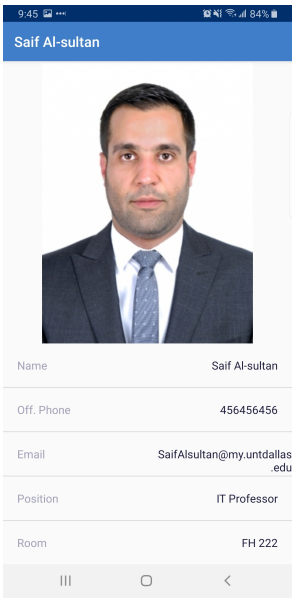

Figure 5.9: A picture showcasing detailed information within the Directory feature of the system.

Figure 5.8 showcases the implementation of the Directory feature of the system. This page allows users to access information about staff and professors within the campus. Selecting an item within the list will display figure 5.9. The code behind this page's implementation can be found on Appendix A.7

Dart:

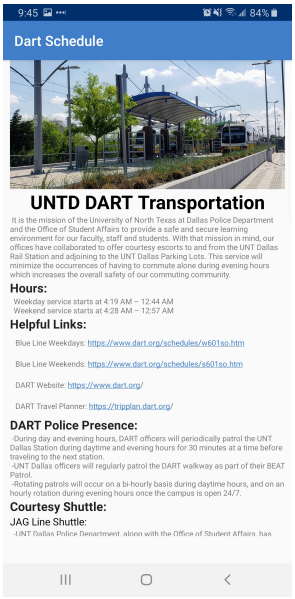

Figure 5.10: A picture of the Dart feature of the system.

Figure 5.10 showcases the implementation of the Dart feature of the system. This page allows users to access information about the Dart station close to the campus. The feature offers useful links and routine schedules. The code behind this page's implementation can be found on Appendix A.8

Email:

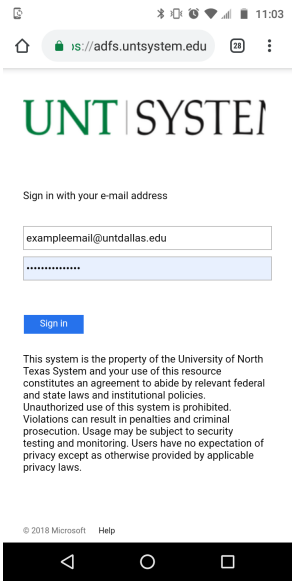

Figure 5.11: A picture of the Email feature of the system.

Figure 5.11 showcases the implementation of the Email feature of the system. This page allows users to access their school email. The feature only requires login once and provides access to up-to-date information. The code behind this page's implementation can be found on Appendix A.1 as part of the Home Page.

Calendar:

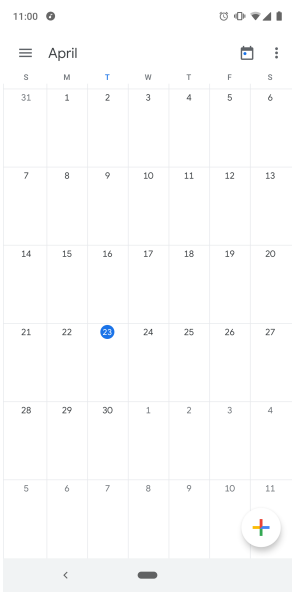

Figure 5.12: A picture of the Calendar feature of the system.

Figure 5.12 showcases the implementation of the Calendar feature of the system. This page allows users to access their personal calendar. The feature checks which calendar the user uses and updates accordingly. The code behind this page's implementation can be found on Appendix A.1 as part of the Home Page.

Events:

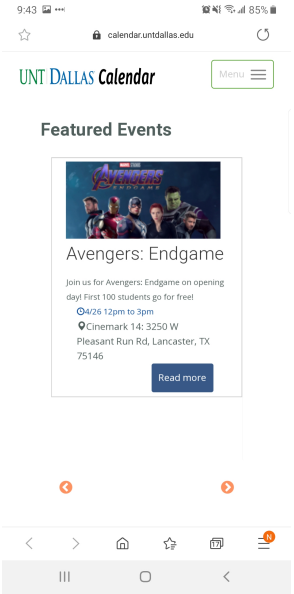

Figure 5.13: A picture of the Events feature of the system.

Figure 5.13 showcases the implementation of the Events feature of the system. This page allows users to access information about upcoming events within the campus. The feature provides a list of the events with the latest being displayed at the top. The code behind this page's implementation can be found on Appendix A.1 as part of the Home Page.

Canvas:

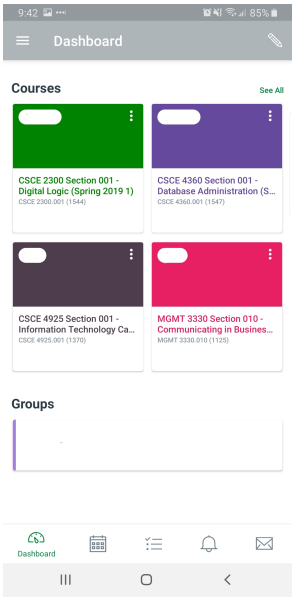

Figure 5.14: A picture of the Canvas feature of the system.

Figure 5.14 showcases the implementation of the Canvas feature of the system. This page allows users to access their personal Canvas account. The feature checks whether the app is installed in case it needs to be opened on the browser. The code behind this page's implementation can be found on Appendix A.1 as part of the Home Page.

5.3 Testing

Test Scenario 1:

In this scenario the user, Miguel Torres will open the application and will navigate the application in order to find campus club information. The following images depict the process the user took in order to access the club's information. Step 1: Open the application. The first step Miguel tapped on the application icon and was greeted by the applications home screen.

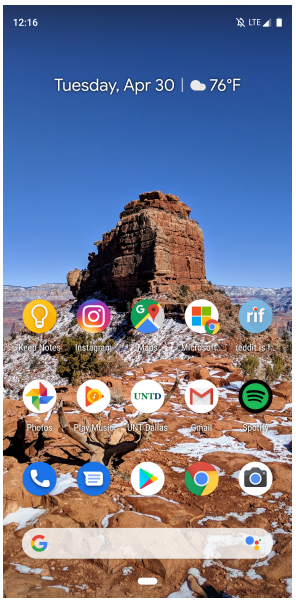

Figure 5.15: Depiction of the first step of the first test scenario.

Step 2: Once the application has been opened, the user will have several features to choose from. For this particular scenerio the user will select the "Organizations" option.

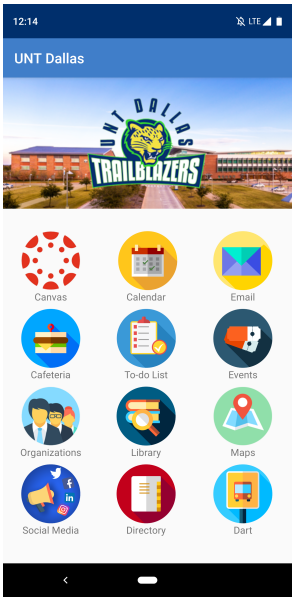

Figure 5.16: Depiction of the second step of the first test scenario.

Step 3: After tapping on the "Organizations" feature Miguel was greeted by the different clubs that are available in the campus. For the sake of this scenario Miguel tapped on the "Information Technology"

option.

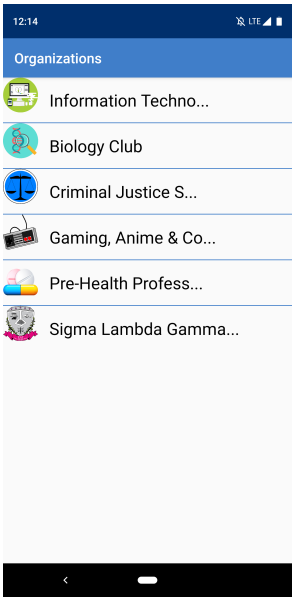

Figure 5.17: Depiction of the third step of the first test scenario.

Step 4: Once the app loaded the screen Miguel was greeted by information that was related to that club. Specifically speaking the contact information to reach out to that club and who is the president of the club.

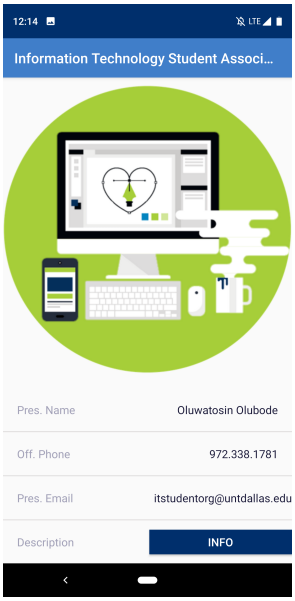

Figure 5.18: Depiction of the fourth step of the first test scenario.

Step 5: For the final step of the testing process, Miguel tapped on the “Info” button, this button brought

up a more detailed screen displaying the club's information, and what their mission is. The image below illustrates the clubs mission statement.

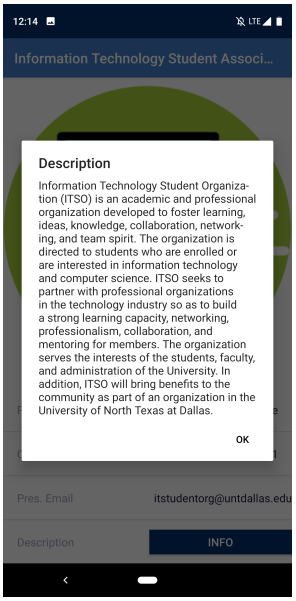

Figure 5.19: Depiction of the fifth step of the first test scenario.

Test Scenario 2:

For this testing scenario, Jared will use the app in order to find cafeteria food information. The following pictures describe the steps needed in order to access this information. Step 1: Jared taps on the application, taking him to the app's main screen.

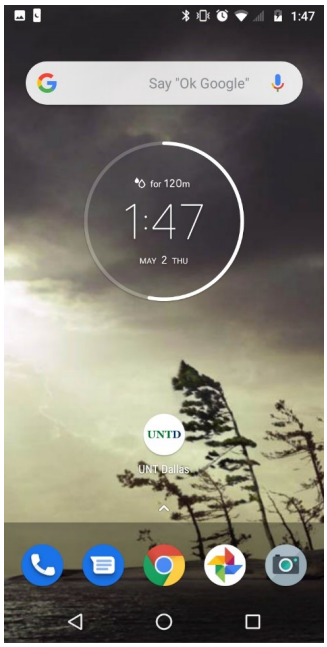

Figure 5.20: Depiction of the first step of the second test scenario.

Step 2: Jared taps on the cafeteria option, out of the many options provided.

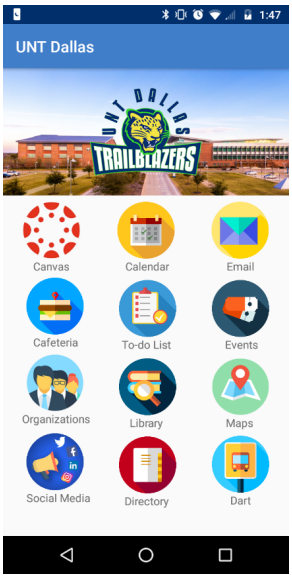

Figure 5.21: Depiction of the second step of the second test scenario.

Step 3: After tapping on this option, the cafeteria menu opens. By default, it begins on the breakfast tab for that day.

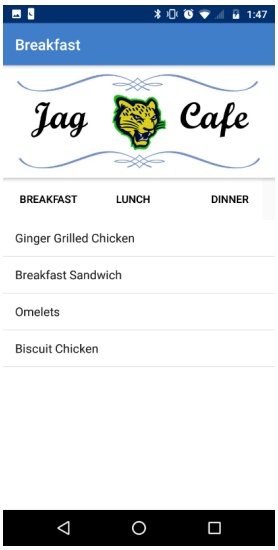

Figure 5.22: Depiction of the third step of the second test scenario.

Step 4: Jared taps on the Lunch tab, and the app displays the lunch options.

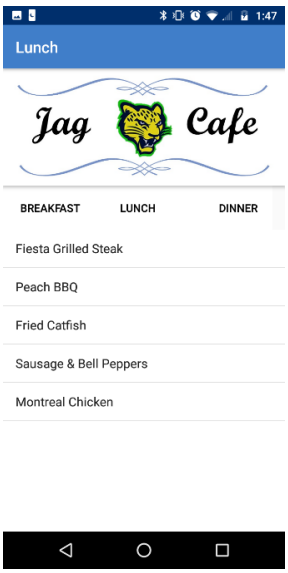

Figure 5.23: Depiction of the fourth step of the second test scenario.

Step 5: Jared taps on the Dinner tab, and dinner options are displayed.

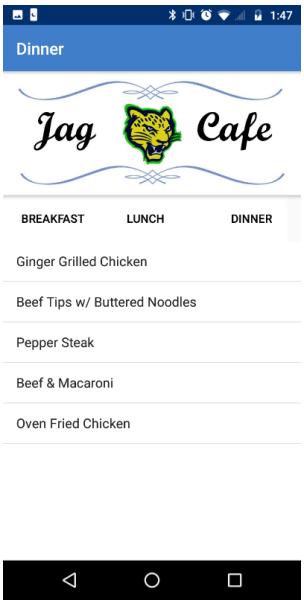

Figure 5.24: Depiction of the fifth step of the second test scenario.

Step 6: Finally, Jared taps again on the breakfast tab, to confirm that it is the same as the default option.

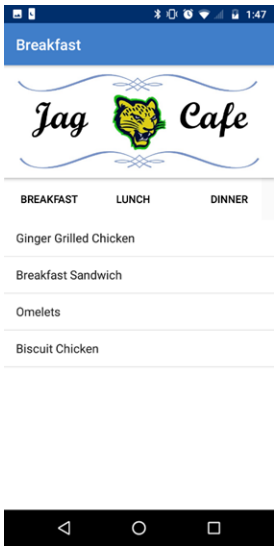

Figure 5.25: Depiction of the sixth step of the second test scenario.

5.4 Summary

This chapter showcased the implementation of the system. Each graphical user interface implemented within the application was presented and explained. The testing section demonstrated several GUIs that corre-

sponded with the activity being tested. The tests that were performed on the system were also summarized underneath the GUI in order to explain what is happening in the specific scenarios.

Chapter 6

Conclusion and Future Work

Objectives:

- Present the conclusion to the system's proposal.
- Describe future work related to the system.

6.1 Conclusion

This report presented the UNTDallas Mobile Application. This software system is a mobile application for the android platform which provides essential features for students within the University of North Texas at Dallas. The following list summarizes the previous chapters:

- Chapter 1 introduced the motivation behind developing the UNTDallas Mobile Application. This chapter outlined each section within the report along with a short description of each.
- Chapter 2 served as a literature review in which similar software systems were surveyed. This chapter also outlined the tools available implementation of the system and highlighted which ones were chosen.
- Chapter 3 defined the user and system requirements for the system. The chapter outlined the requirements engineering process and how it related to our proposed system.
- Chapter 4 showcased the design of the proposed system. A model of each graphical user interfaces was presented as it related to each planned feature. The complete design of the system's database was also present within this chapter.
- Chapter 5 presented the implementation of the system. The graphical user interfaces of each feature within the system were showcased and explained. This section also provided testing scenarios to guarantee the system's functionality.

6.2 Future Work

There are some features and further improvements the team would be interested in implementing in order to further help students. However due to a lack of time these features were not able to be implemented. The following is a list of said features:

- Database: The team originally planned to use MySQL as the database management system due to its many powerful features and their familiarity with the software. However there were several difficulties when attempting to use MySQL alongside the Android platform. The first obstacle was the fact that MySQL does not have native support for Android. Furthermore the work-arounds were not reliable and caused development problems. The second obstacle was the amount of time and investment it would have taken to ensure MySQL worked properly with the application. Instead the team decided

to use SQLite as the database management system of the application due to its support for Android. While this decision allowed the team to develop the application without delays it also caused a major problem. Unlike other systems SQLite is designed to be localized within a device. In the future, the team would like to use MySQL or another database so that information can be updated at any time and without requiring an application update.

- Bargain bin: The team originally wanted to give students an easy way to trade books and items with each other. This desire lead to the idea to have the bargain bin feature. This feature would make trading, buying and selling within the campus much easier. Students would no longer have to go to the bookstore or online to obtain items. However, since it took too long to develop a client to manage the buying and selling within the app we decided to leave it for future work.

- To-do list: The team wanted to give students an easy way to manage their tasks throughout the day through a to-do list feature. Users would be able to write down tasks and then mark them off once they were completed. However, since other applications provided similar functionality the team felt it should focus on other features and leave this one for future work.

- Notifications: The Notifications was originally planned to be a useful feature within our application. Students could be notified of additional assignments, new emails, messages and more. However, since we did not have experience with the notifications in Android and since other apps like Canvas provided their own notification capabilities we decided to leave notifications for future work.

- Sign in/up: The team wanted to provide a sign in feature so guests, students, and faculty/staff could have their own varying functions within the app. This would allow many options depending on the classification of the user. For example, students would have access to their assignments while professors would be allowed to post announcements. However, since there could be too many additional features based on the user, and we did not have access to the University's official database, we decided to leave it for future work.

Appendix A

Implementation Code

This is the appendix of the report. This section provides the kotlin code and XML code for the implementation of the system.

A.1 Home Page

This is the Kotlin code for the implementation of the Home Page.

```
1
2  import android.content.Intent
3  import android.content.pm.PackageManager
4  import android.content.pm.ResolveInfo
5  import android.net.Uri
6  import android.os.Bundle
7  import androidx.appcompat.app.AppCompatActivity
8  import android.widget.*
9
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         //Changes title for the activity
```

```
16          title = ("UNT Dallas")
17          //Checks to make sure that there's a native app to open the material, otherwise it will use
                the web browser.
18          val activities: List<ResolveInfo> = packageManager.queryIntentActivities(
19                 intent,
20                 PackageManager.MATCH_DEFAULT_ONLY
21          )
22          val isIntentSafe: Boolean = activities.isNotEmpty()
23
24          //Creates the buttons that will be used for the user to navigate to different pages.
25
26          //This button is for the canvas Feature
27          val canvasButton = findViewById<Button>(R.id.canvasButton)
28
29          //This button is for the Calendar Feature
30          val calendarButton = findViewById<Button>(R.id.calendarButton)
31
32          //This button is for the email Feature
33          val emailButton = findViewById<Button>(R.id.emailButton)
34
35          //This button is for the Cafeteria Feature
36          val cafeteriaButton = findViewById<Button>(R.id.cafeteriaButton)
37
38          //This button is for the to-do Feature
39          val todoButton = findViewById<Button>(R.id.todoButton)
40
41          //This button is for the Events Feature
42          val eventsButton = findViewById<Button>(R.id.eventButton)
43
44          //This button is for the Organizations Feature
45          val organizationButton = findViewById<Button>(R.id.organizationButton)
46
47          //This button is for the Library Feature
48          val libraryButton = findViewById<Button>(R.id.libraryButton)
```

```
49
50          //This button is for the Maps Feature
51          val mapsButton = findViewById<Button>(R.id.mapButton)
52
53          //This button is for the Social Media Feature
54          val socialButton = findViewById<Button>(R.id.socialButton)
55
56          //This button is for the Directory Feature
57          val directoryButton = findViewById<Button>(R.id.directoryButton)
58
59          //This button is for the Dart Feature
60          val dartButton = findViewById<Button>(R.id.dartButton)
61
62         //*************************************************************************************
63          //This creates the intents that will be triggered once a user taps on one of the buttons it
                will
64          //perform the corresponding action to that particular button.
65
66          //Pressing Canvas icon will open the canvas app.
67          val canvasPackage = "com.instructure.candroid"
68          val canvasPackage2 = "com.instructure.teacher"
69          val canvas: Intent = Uri.parse("https://untdallas.instructure.com/").let { webpage ->
70              Intent(Intent.ACTION_VIEW, webpage)
71          }
72
73          canvasButton.setOnClickListener(){
74              canvas(canvasPackage)
75              canvasTeacher(canvasPackage2)
76              if (isIntentSafe) {
77                  startActivity(canvas)
78              }
79          }
80
81          //creates objects to open the calendar for different cellphone manufacturers.
```

```
        val playCalendar = "com.google.android.calendar"
        val galaxyCalendar = "com.samsung.android.calendar"
        val genericCalendar = "com.android.calendar"

        //Pressing the calendar button allows you to open your calendar app and add events.
        calendarButton.setOnClickListener(){
            googleCalendar(playCalendar)
            samsungCalendar(galaxyCalendar)
            generalCalendar(genericCalendar)
        }

        //Pressing the email button allows you to see your school information.
        emailButton.setOnClickListener{
            val uri = Uri.parse("http://webmail.unt.edu/")
            val email = Intent(Intent.ACTION_VIEW, uri)
            startActivity(email)
        }

        //Pressing the cafeteria button allows you to see the cafeteria menu.
        cafeteriaButton.setOnClickListener(){
            startActivity(Intent(this@MainActivity, cafeteriaAttempt::class.java))
        }

        //Pressing to-do button opens the to-do list feature.
        todoButton.setOnClickListener(){
            startActivity(Intent(this@MainActivity, ToDoList::class.java))
        }

        //Pressing the events button displays the school events.
        eventsButton.setOnClickListener{
            val uri = Uri.parse("https://calendar.untdallas.edu/")
            val email = Intent(Intent.ACTION_VIEW, uri)
            startActivity(email)
        }
```

```
116
117         //Pressing the organizations button allows you to see the school organizations.
118         organizationButton.setOnClickListener(){
119             startActivity(Intent(this@MainActivity, OrganizationsActivity::class.java))
120         }
121
122         //Pressing the library button will open library information.
123         libraryButton.setOnClickListener(){
124             startActivity(Intent(this@MainActivity, Library::class.java))
125         }
126
127         //Pressing the maps button allows you to see the school maps.
128         mapsButton.setOnClickListener(){
129             startActivity(Intent(this@MainActivity, Maps::class.java))
130         }
131
132         //Pressing Social Media will open the social media page and links.
133         socialButton.setOnClickListener(){
134             startActivity(Intent(this@MainActivity, socialMedia::class.java))
135         }
136
137         //Pressing the directory button allows you to see the staff directory.
138         directoryButton.setOnClickListener(){
139             startActivity(Intent(this@MainActivity, DirectoryActivity::class.java))
140         }
141
142         //Pressing the dart button allows the user to see public transportation information.
143         dartButton.setOnClickListener(){
144             startActivity(Intent(this@MainActivity, Dart::class.java))
145         }
146
147     }
148
149     //*****Objects created to call them depending on what apps are installed on device*****
```

```
150
151     //Uses the regular Student Canvas application.
152     private fun canvas(packageName: String){
153         val pm = applicationContext.packageManager
154         val intent: Intent? = pm.getLaunchIntentForPackage(packageName)
155         intent?.addCategory(Intent.CATEGORY_LAUNCHER)
156         if(intent!=null){
157             applicationContext.startActivity(intent)
158         }
159     }
160
161     //Uses the Teacher Canvas application.
162     private fun canvasTeacher(packageName: String){
163         val pm = applicationContext.packageManager
164         val intent: Intent? = pm.getLaunchIntentForPackage(packageName)
165         intent?.addCategory(Intent.CATEGORY_LAUNCHER)
166         if(intent!=null){
167             applicationContext.startActivity(intent)
168         }
169     }
170
171     //Uses the Google calendar application on most android devices.
172     private fun googleCalendar(packageName: String){
173         val pm = applicationContext.packageManager
174         val intent: Intent? = pm.getLaunchIntentForPackage(packageName)
175         intent?.addCategory(Intent.CATEGORY_LAUNCHER)
176         if(intent!=null){
177             applicationContext.startActivity(intent)
178         }
179     }
180
181     //Uses the Samsung calendar if running app on Samsung device
182     private fun samsungCalendar(packageName: String){
183         val pm = applicationContext.packageManager
```

```
        val intent: Intent? = pm.getLaunchIntentForPackage(packageName)
        intent?.addCategory(Intent.CATEGORY_LAUNCHER)
        if(intent!=null){
            applicationContext.startActivity(intent)
        }
    }

    //Uses this calendar app instead of the other mentioned above.
    private fun generalCalendar(packageName: String){
        val pm = applicationContext.packageManager
        val intent: Intent? = pm.getLaunchIntentForPackage(packageName)
        intent?.addCategory(Intent.CATEGORY_LAUNCHER)
        if(intent!=null){
            applicationContext.startActivity(intent)
        }
    }


}
```

This is the XML code for the implementation of the Home Page.

```

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!-- This button is for the Maps Feature -->
```

```
13
14      <ScrollView
15          android:layout_width="match_parent"
16          android:layout_height="match_parent">
17
18          <LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content"
19              android:orientation="vertical">
20
21              <androidx.constraintlayout.widget.ConstraintLayout
22                  android:layout_width="match_parent"
23                  android:layout_height="match_parent">
24
25                  <ImageView
26                      android:layout_width="410dp"
27                      android:layout_height="161dp" app:srcCompat="@drawable/unt_header3"
28                      android:id="@+id/headerButton"
29                      tools:layout_editor_absoluteY="-7dp"
                            app:layout_constraintStart_toStartOf="parent"
30                      app:layout_constraintHorizontal_bias="0.5"
                            app:layout_constraintEnd_toEndOf="parent"/>
31
32                  <Button
33                      android:id="@+id/dartButton"
34                      android:layout_width="72dp"
35                      android:layout_height="72dp"
36                      android:background="@drawable/bus_stop"
37                      android:text=""
38                      app:layout_constraintTop_toBottomOf="@+id/textView"
39                      app:layout_constraintStart_toEndOf="@+id/guideline2"
40                      app:layout_constraintHorizontal_bias="0.5"
                            app:layout_constraintEnd_toEndOf="parent"
41                      android:layout_marginTop="8dp"/>
42
43                  <Button
```

```
44                  android:id="@+id/eventButton"
45                  android:layout_width="72dp"
46                  android:layout_height="72dp"
47                  android:background="@drawable/events"
48                  android:text=""
49                  app:layout_constraintStart_toEndOf="@+id/guideline2"
50                  app:layout_constraintHorizontal_bias="0.5"
51                  app:layout_constraintEnd_toEndOf="parent" android:layout_marginTop="8dp"
52                  app:layout_constraintTop_toBottomOf="@+id/textView10"/>
53
54              <TextView
55                  android:text="@string/maps"
56                  android:layout_width="31dp"
57                  android:layout_height="18dp"
58                  android:id="@+id/textView"
59                  app:layout_constraintTop_toBottomOf="@+id/mapButton"
60                  app:layout_constraintStart_toEndOf="@+id/guideline2"
61                  app:layout_constraintHorizontal_bias="0.566"
62                  app:layout_constraintEnd_toEndOf="parent"/>
63
64              <TextView
65                  android:text="@string/library"
66                  android:layout_width="wrap_content"
67                  android:layout_height="wrap_content"
68                  android:id="@+id/textView13"
69                  app:layout_constraintTop_toBottomOf="@+id/libraryButton"
70                  app:layout_constraintStart_toEndOf="@+id/guideline"
71                  app:layout_constraintHorizontal_bias="0.586"
72                  app:layout_constraintEnd_toStartOf="@+id/guideline2"/>
73
74              <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
75                  android:layout_height="wrap_content"
76                  android:id="@+id/guideline2"
77                  app:layout_constraintGuide_percent="0.66"
```

```
78                  android:orientation="vertical"/>
79
80              <Button
81                  android:id="@+id/mapButton"
82                  android:layout_width="72dp"
83                  android:layout_height="72dp"
84                  android:background="@drawable/maps"
85                  android:text=""
86                  app:layout_constraintTop_toBottomOf="@+id/textView12"
87                  app:layout_constraintStart_toEndOf="@+id/guideline2"
88                  app:layout_constraintHorizontal_bias="0.5"
89                  app:layout_constraintEnd_toEndOf="parent" android:layout_marginTop="8dp"/>
90
91              <TextView
92                  android:text="@string/organizations"
93                  android:layout_width="86dp"
94                  android:layout_height="20dp"
95                  android:id="@+id/textView5"
96                  app:layout_constraintTop_toBottomOf="@+id/organizationButton"
97                  app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintHorizontal_bias="0.727"
98                  app:layout_constraintEnd_toStartOf="@+id/guideline"/>
99
100             <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
101                 android:layout_height="wrap_content"
102                 android:id="@+id/guideline"
103                 app:layout_constraintGuide_percent="0.33"
104                 android:orientation="vertical"/>
105
106             <TextView
107                 android:text="@string/canvas"
108                 android:layout_width="wrap_content"
109                 android:layout_height="16dp"
110                 android:id="@+id/textView8"
```

```
111                 app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintHorizontal_bias="0.589"
112                 app:layout_constraintEnd_toStartOf="@+id/guideline"
113                 app:layout_constraintTop_toBottomOf="@+id/canvasButton"/>
114
115             <Button
116                 android:id="@+id/emailButton"
117                 android:layout_width="72dp"
118                 android:layout_height="72dp"
119                 android:background="@drawable/email"
120                 android:text=""
121                 app:layout_constraintStart_toEndOf="@+id/guideline2"
122                 android:layout_marginLeft="23dp" android:layout_marginStart="23dp"
123                 app:layout_constraintEnd_toEndOf="parent"
124                 android:layout_marginEnd="24dp" android:layout_marginRight="24dp"
125                 app:layout_constraintHorizontal_bias="0.523" android:layout_marginTop="8dp"
126                 app:layout_constraintTop_toBottomOf="@+id/headerButton"/>
127
128             <Button
129                 android:id="@+id/libraryButton"
130                 android:layout_width="72dp"
131                 android:layout_height="72dp"
132                 android:background="@drawable/library"
133                 android:text=""
134                 app:layout_constraintTop_toBottomOf="@+id/textView2"
135                 app:layout_constraintStart_toEndOf="@+id/guideline"
136                 app:layout_constraintHorizontal_bias="0.586"
137                 app:layout_constraintEnd_toStartOf="@+id/guideline2"
                        android:layout_marginTop="8dp"/>
138
139             <TextView
140                 android:text="To-do List"
141                 android:layout_width="wrap_content"
142                 android:layout_height="wrap_content"
```

```
143                 android:id="@+id/textView2"
144                 app:layout_constraintTop_toBottomOf="@+id/todoButton"
145                 app:layout_constraintStart_toEndOf="@+id/guideline"
                        app:layout_constraintHorizontal_bias="0.6"
146                 app:layout_constraintEnd_toStartOf="@+id/guideline2"/>
147
148             <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
149                 android:layout_height="wrap_content"
150                 android:id="@+id/guideline5"
151                 app:layout_constraintGuide_percent="0.33"
152                 android:orientation="horizontal"/>
153
154             <TextView
155                 android:text="Events"
156                 android:layout_width="44dp"
157                 android:layout_height="19dp"
158                 android:id="@+id/textView12"
159                 app:layout_constraintTop_toBottomOf="@+id/eventButton"
160                 app:layout_constraintStart_toEndOf="@+id/guideline2"
161                 app:layout_constraintHorizontal_bias="0.584"
                        app:layout_constraintEnd_toEndOf="parent"
162                 />
163
164             <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
165                 android:layout_height="wrap_content"
166                 android:id="@+id/guideline3"
167                 app:layout_constraintGuide_percent="1.0"
168                 android:orientation="vertical"/>
169
170             <Button
171                 android:id="@+id/directoryButton"
172                 android:layout_width="72dp"
173                 android:layout_height="72dp"
174                 android:background="@drawable/directory"
```

```
175                 android:text=""
176                 app:layout_constraintTop_toBottomOf="@+id/textView13"
177                 app:layout_constraintStart_toEndOf="@+id/guideline"
178                 app:layout_constraintHorizontal_bias="0.586"
179                 app:layout_constraintEnd_toStartOf="@+id/guideline2"
                        android:layout_marginTop="8dp"/>
180
181             <Button
182                 android:id="@+id/organizationButton"
183                 android:layout_width="72dp"
184                 android:layout_height="72dp"
185                 android:background="@drawable/organization"
186                 android:text=""
187                 app:layout_constraintTop_toBottomOf="@+id/textView15"
188                 app:layout_constraintStart_toStartOf="parent"
189                 app:layout_constraintHorizontal_bias="0.638"
190                 app:layout_constraintEnd_toStartOf="@+id/guideline"
                        android:layout_marginTop="8dp"/>
191
192             <Button
193                 android:id="@+id/canvasButton"
194                 android:layout_width="72dp"
195                 android:layout_height="72dp"
196                 android:background="@drawable/canvas"
197                 android:text=""
198                 app:layout_constraintStart_toStartOf="parent" android:layout_marginLeft="16dp"
199                 android:layout_marginStart="16dp"
                        app:layout_constraintEnd_toStartOf="@+id/guideline"
200                 android:layout_marginEnd="16dp" android:layout_marginRight="16dp"
201                 app:layout_constraintHorizontal_bias="0.781" android:layout_marginTop="8dp"
202                 app:layout_constraintTop_toBottomOf="@+id/headerButton"/>
203             <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
204                 android:layout_height="wrap_content"
205                 android:id="@+id/guideline6"
```

```
206                 app:layout_constraintGuide_percent="0.66"
207                 android:orientation="horizontal"/>
208
209             <androidx.constraintlayout.widget.Guideline android:layout_width="wrap_content"
210                 android:layout_height="wrap_content"
211                 android:id="@+id/guideline7"
212                 app:layout_constraintGuide_percent="1.0"
213                 android:orientation="horizontal"/>
214
215             <TextView
216                 android:text="Dart"
217                 android:layout_width="54dp"
218                 android:layout_height="20dp"
219                 android:id="@+id/textView4"
220                 app:layout_constraintTop_toBottomOf="@+id/dartButton"
221                 app:layout_constraintStart_toEndOf="@+id/guideline2"
222                 app:layout_constraintHorizontal_bias="0.567"
                        app:layout_constraintEnd_toEndOf="parent"
223                 />
224
225             <Button
226                 android:id="@+id/todoButton"
227                 android:layout_width="72dp"
228                 android:layout_height="72dp"
229                 android:background="@drawable/todo"
230                 android:text=""
231                 app:layout_constraintStart_toEndOf="@+id/guideline"
                        app:layout_constraintHorizontal_bias="0.586"
232                 app:layout_constraintEnd_toStartOf="@+id/guideline2"
                        android:layout_marginTop="8dp"
233                 app:layout_constraintTop_toBottomOf="@+id/textView3"/>
234
235             <Button
236                 android:id="@+id/socialButton"
```

```
237                 android:layout_width="72dp"
238                 android:layout_height="72dp"
239                 android:background="@drawable/social_media"
240                 android:text=""
241                 android:layout_marginTop="8dp"
242                 app:layout_constraintTop_toBottomOf="@+id/textView5"
243                 app:layout_constraintStart_toStartOf="parent"
244                 app:layout_constraintHorizontal_bias="0.638"
245                 app:layout_constraintEnd_toStartOf="@+id/guideline"/>
246             <TextView
247
248                 android:text="@string/email"
249                 android:layout_width="wrap_content"
250                 android:layout_height="wrap_content"
251                 android:id="@+id/textView10"
252                 app:layout_constraintTop_toBottomOf="@+id/emailButton"
253                 app:layout_constraintStart_toEndOf="@+id/guideline2"
254                 app:layout_constraintHorizontal_bias="0.466"
                        app:layout_constraintEnd_toEndOf="parent"
255                 />
256
257             <TextView
258                 android:text="@string/calendar"
259                 android:layout_width="wrap_content"
260                 android:layout_height="19dp"
261                 android:id="@+id/textView3"
262                 app:layout_constraintTop_toBottomOf="@+id/calendarButton"
263                 app:layout_constraintStart_toEndOf="@+id/guideline"
264                 app:layout_constraintHorizontal_bias="0.556"
265                 app:layout_constraintEnd_toStartOf="@+id/guideline2"/>
266
267             <TextView
268                 android:text="@string/directory"
269                 android:layout_width="57dp"
```

```
270                 android:layout_height="18dp"
271                 android:id="@+id/textView6"
272                 app:layout_constraintTop_toBottomOf="@+id/directoryButton"
273                 app:layout_constraintStart_toEndOf="@+id/guideline"
                        app:layout_constraintHorizontal_bias="0.606"
274                 app:layout_constraintEnd_toStartOf="@+id/guideline2"
275                 />
276
277             <TextView
278                 android:text="@string/social_media"
279                 android:layout_width="82dp"
280                 android:layout_height="19dp"
281                 android:id="@+id/textView7"
282                 app:layout_constraintTop_toBottomOf="@+id/socialButton"
283                 app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintHorizontal_bias="0.648"
284                 app:layout_constraintEnd_toStartOf="@+id/guideline"
285                 />
286
287             <Button
288                 android:id="@+id/cafeteriaButton"
289                 android:layout_width="72dp"
290                 android:layout_height="72dp"
291                 android:background="@drawable/cafeteria_menu"
292                 android:text=""
293                 app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintHorizontal_bias="0.638"
294                 app:layout_constraintEnd_toStartOf="@+id/guideline"
                        android:layout_marginTop="8dp"
295                 app:layout_constraintTop_toBottomOf="@+id/textView8"/>
296
297             <TextView
298                 android:text="@string/cafeteria"
299                 android:layout_width="59dp"
```

```
                    android:layout_height="17dp"
                    android:id="@+id/textView15"
                    app:layout_constraintTop_toBottomOf="@+id/cafeteriaButton"
                    app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintHorizontal_bias="0.616"
                    app:layout_constraintEnd_toStartOf="@+id/guideline"/>

                <Button
                    android:id="@+id/calendarButton"
                    android:layout_width="72dp"
                    android:layout_height="72dp"
                    android:background="@drawable/calendar"
                    android:text=""
                    app:layout_constraintStart_toStartOf="parent" android:layout_marginLeft="144dp"
                    android:layout_marginStart="144dp" android:layout_marginEnd="144dp"
                    android:layout_marginRight="144dp"
                    app:layout_constraintEnd_toEndOf="parent"
                        app:layout_constraintHorizontal_bias="0.509"
                    android:layout_marginTop="8dp"
                        app:layout_constraintTop_toBottomOf="@+id/headerButton"/>
            </androidx.constraintlayout.widget.ConstraintLayout>

        </LinearLayout>

    </ScrollView>

    <!-- This button is for the Notifications Feature -->


</androidx.constraintlayout.widget.ConstraintLayout>
```

A.2 Cafeteria

This is the Kotlin code for the implementation of the Cafeteria feature.

```
1  import androidx.appcompat.app.AppCompatActivity
2  import android.os.Bundle
3  import android.view.View
4  import android.widget.Button
5  import android.widget.ListView
6  import android.widget.ArrayAdapter
7
8  class cafeteriaAttempt : AppCompatActivity() {
9
10     override fun onCreate(savedInstanceState: Bundle?) {
11
12         super.onCreate(savedInstanceState)
13
14         setContentView(R.layout.activity_cafeteria_attempt)
15
16         //Button that will display breakfast when pressed
17         var breakfast = findViewById<Button>(R.id.leftB)
18
19         //Button that will display lunch when pressed
20         var lunch = findViewById<Button>(R.id.centerB)
21
22         //Button that will display dinner when pressed
23         var dinner = findViewById<Button>(R.id.rightB)
24
25         //This array will be used to store the names of all the breakfast items.
26         var breakfastOne = emptyArray<String>()
27
28         //This array will be used to store the names of all the lunch items.
29         var LunchOne = emptyArray<String>()
30
```

```
31          //This array will be used to store the names of all the dinner items.
32          var dinnerOne = emptyArray<String>()
33
34          //This array will store the Cafeteria object that get inputted into the database.
35          //These Cafeteria objects will be used later on to create the listview
36          var allCafe = emptyArray<Cafeteria>()
37
38          //Creates an object from the DBHandler_v2 class.
39          //This object is used to access the database. Use its methods to access the database.
40          var db = DBHandler_v2(this);
41
42          //Creates an object from cafeteria and stores it in the database.
43          //These object are also stored in the array allCafe.
44          //Creating a Cafeteria object requires three values: ID (int), FoodType(String),
                FoodDescription(String).
45
46          //Cafeteria objects related to breakfast are inserted here
47          var test = db.addCafeteria(Cafeteria(0,"breakfast","Ginger Grilled Chicken"))
48          allCafe = allCafe + test
49          test = db.addCafeteria(Cafeteria(1,"breakfast","Breakfast Sandwich"))
50          allCafe = allCafe + test
51          test =db.addCafeteria(Cafeteria(2,"breakfast","Omelets"))
52          allCafe = allCafe + test
53          test =db.addCafeteria(Cafeteria(3,"breakfast","Biscuit Chicken"))
54          allCafe = allCafe + test
55
56          //Cafeteria objects related to lunch are inserted here
57
58          test =db.addCafeteria(Cafeteria(4,"lunch","Fiesta Grilled Steak"))
59          allCafe = allCafe + test
60          test =db.addCafeteria(Cafeteria(5,"lunch","Peach BBQ"))
61          allCafe = allCafe + test
62          test =db.addCafeteria(Cafeteria(6,"lunch","Fried Catfish"))
63          allCafe = allCafe + test
```

```
64          test =db.addCafeteria(Cafeteria(7,"lunch","Sausage & Bell Peppers"))
65          allCafe = allCafe + test
66          test = db.addCafeteria(Cafeteria(8,"lunch","Montreal Chicken"))
67          allCafe = allCafe + test
68
69          //Cafeteria objects related to dinner are inserted here
70
71          test = db.addCafeteria(Cafeteria(9,"dinner","Ginger Grilled Chicken"))
72          allCafe = allCafe + test
73          test =db.addCafeteria(Cafeteria(10,"dinner","Beef Tips w/ Buttered Noodles"))
74          allCafe = allCafe + test
75          test =db.addCafeteria(Cafeteria(11,"dinner","Pepper Steak"))
76          allCafe = allCafe + test
77          test = db.addCafeteria(Cafeteria(12,"dinner","Beef & Macaroni"))
78          allCafe = allCafe + test
79          test =db.addCafeteria(Cafeteria(13,"dinner","Oven Fried Chicken"))
80          allCafe = allCafe + test
81
82          //This loops through all objects in the allCafe array.
83          //This loop inserts values into an array depending on the foodType of the object.
84          //Remember that we must display the name of the item which is variable foodDescription
85          for(cafe in allCafe){
86              if(cafe.foodType.equals("breakfast")){
87                  breakfastOne = breakfastOne + cafe.foodDescription
88              }
89              else if(cafe.foodType.equals("lunch")){
90                  LunchOne = LunchOne + cafe.foodDescription
91              }
92              else if(cafe.foodType.equals("dinner")){
93                  dinnerOne = dinnerOne + cafe.foodDescription
94              }
95          }
96
97          //Creates an adapters that can be inserted into a listview.
```

```
98
99          //Adapter that can display the Breakfast items.
100         val adapter = ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,breakfastOne);
101
102         //Adapter that can display the Lunch items.
103         val adapterOne = ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,LunchOne);
104
105         //Adapter that can display the Dinner items.
106         val adapterTwo = ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,dinnerOne);
107
108         //This is the only listview within the whole activity.
109         var simpleList = findViewById<View>(R.id.simpleListView) as ListView
110
111         //Sets the initial adapter of the listview to be the one for Breakfast.
112         simpleList.setAdapter(adapter)
113
114         //Sets the initial title of the activity to breakfast. Will be changed later.
115         setTitle("Breakfast")
116
117         //When you click on the breakfast button the listview gets the corresponding adapter.
118         //Clicking the lunch button sets the adapter of the listview to the lunch adapter. The
                listview stays the same.
119         breakfast.setOnClickListener(){
120             simpleList.setAdapter(adapter)
121             setTitle("Breakfast")
122         }
123
124         lunch.setOnClickListener(){
125             simpleList.setAdapter(adapterOne)
126             setTitle("Lunch")
127         }
128
129         dinner.setOnClickListener(){
130             simpleList.setAdapter(adapterTwo)
```

```
            setTitle("Dinner")
        }

    }

}
```

This is the XML code for the implementation of the Cafeteria feature.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".cafeteriaAttempt">

    <com.google.android.material.appbar.AppBarLayout

        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:id="@+id/appbarid"
        android:background="@color/colorPrimary"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="0dp">

        <ImageView
            android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:padding="10dp"
            android:src="@drawable/jagcafe"
            android:id="@+id/jagcafeimageid"
            android:contentDescription="image"
            android:background="@color/white"/>
    </com.google.android.material.appbar.AppBarLayout>

    <androidx.appcompat.widget.Toolbar
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:id="@+id/app_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/simpleListView"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintVertical_bias="0.05"
        android:background="@color/white">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Breakfast"
            android:layout_gravity="start"
            android:id="@+id/leftB"
            android:layout_weight=".30"
            android:background="@color/white"
            tools:layout_editor_absoluteX="16dp"
            android:textColor="@color/colorAccent"
```

```
60            android:visibility="visible" style="@style/Widget.AppCompat.Button.Borderless"
61            tools:layout_editor_absoluteY="5dp"/>
62
63        <Button
64            android:layout_width="wrap_content"
65            android:layout_height="wrap_content"
66            android:text="Lunch"
67            android:layout_gravity="center"
68            android:id="@+id/centerB"
69            android:layout_weight=".30"
70            android:background="@color/white"
71            tools:layout_editor_absoluteX="162dp"
72            android:textColor="@color/colorAccent"
73            android:visibility="visible" style="@style/Widget.AppCompat.Button.Borderless"
74            tools:layout_editor_absoluteY="5dp"/>
75
76        <Button
77            android:layout_width="wrap_content"
78            android:layout_height="wrap_content"
79            android:text="Dinner"
80            android:layout_gravity="end"
81            android:id="@+id/rightB"
82            android:layout_weight=".30"
83            android:background="@color/white"
84            tools:layout_editor_absoluteX="16dp"
85            android:textColor="@color/colorAccent"
86            android:visibility="visible" style="@style/Widget.AppCompat.Button.Borderless"
87            tools:layout_editor_absoluteY="5dp"/>
88
89    </androidx.appcompat.widget.Toolbar>
90
91    <ListView
92        xmlns:android="http://schemas.android.com/apk/res/android"
93        xmlns:tools="http://schemas.android.com/tools"
```

```
        android:id="@+id/simpleListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintVertical_bias="0.949"
        android:background="@color/white">
    </ListView>

</LinearLayout>
```

A.3 Organizations

This is the Kotlin code for the implementation of the Organizations feature.

```
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.view.ViewParent
import android.widget.Adapter
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.ListView
import kotlinx.android.synthetic.main.activity_organizations.*

class OrganizationsActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
```

```
16
17          super.onCreate(savedInstanceState)
18          setContentView(R.layout.activity_organizations)
19          setTitle("Organizations")
20
21          //This variable stores the database instance. Use its functions to access the database.
22          var db = DBHandler_v2(this);
23
24          //Adds an organization to the database. This is the ITSO organization
25          db.addOrg(Organizations(0,"Information Technology Student Association",
26                  "Oluwatosin Olubode","itstudentorg@untdallas.edu","972.338.1781",
27                  "Information Technology Student Organization (ITSO) is an academic " +
28                          "and professional organization developed to foster learning, ideas,
                                knowledge," +
29                          " collaboration, networking, and team spirit. The organization is directed
                                to students" +
30                          " who are enrolled or are interested in information technology and computer
                                science." +
31                          " ITSO seeks to partner with professional organizations in the technology
                                industry so as" +
32                          " to build a strong learning capacity, networking, professionalism,
                                collaboration, and" +
33                          " mentoring for members. The organization serves the interests of the
                                students, faculty," +
34                          " and administration of the University. In addition, ITSO will bring
                                benefits to the community" +
35                          " as part of an organization in the University of North Texas at
                                Dallas.","it"))
36
37          //Adds an organization to the database. This is the Biology club organization
38          db.addOrg(Organizations(1,"Biology Club","Marcela Torres","MarcelaTorres@my.unt.edu",
39                  "972.338.1781",
40                  "The Biology Club is the gathering of Biology majors and any students interested in
                        Biology to" +
```

```
                    " discuss the field as a whole. The group is also interested in studying the
                        different content areas" +
                    " of Biology that are offered in the Bachelors of Science in Biology degree
                        plan, so that all students" +
                    " who are either Biology majors or taking Biology classes will have the
                        tools needed to succeed. The" +
                    " objectives (mission) of The University of North Texas at Dallas Biology
                        Club is to spark an interest in" +
                    " biology and the sciences to students of many different levels. Community
                        service, teamwork, speakers," +
                    " new scientific discoveries, and communication will allow students to
                        enjoy and learn more about the " +
                    "aspects of biology that are unattainable within the classroom setting. By
                        attending Biology Club we hope " +
                    "that the students of UNT Dallas will strengthen their interest in the
                        sciences.","bio"))

        //Adds an organization to the database. This is the Criminal Justice organization
        db.addOrg(Organizations(2,"Criminal Justice Student Association (CJSA)","Alma Alejandro",
                "almaalejandro@my.unt.edu","972.338.1781",
                "The Criminal Justice Students Association is an organization designed to encourage
                    students interested in" +
                    " criminal justice, as well as other interested factions of the University
                        of North Texas at Dallas to share" +
                    " interests and ideas thereby supplementing and enriching our overall
                        learning environment. The Association will" +
                    " assist in the expansion of programs of an educational and cultural nature
                        which will serve the interests of" +
                    " the students, faculty, and administration of the University of North Texas
                        at Dallas. In addition, the outside" +
                    " community will benefit from the existence of The Association at the
                        University of North Texas at Dallas.","chem"))
```

```
61          //Adds an organization to the database. This is the Gaming organization
62          db.addOrg(Organizations(3,"Gaming, Anime & Comics Club","Alejandro
                Valle","alejandrovalle@my.untdallas.edu",
63                 "972.338.1781",
64                 "The Gaming, Anime, and Comics Club purpose is to provide a social gathering spot
                        where members can share a common interest" +
65                         " with other members and introduce new ones. It is an organization for
                                members to make new friends and gain" +
66                         " new fellowship for their time here at UNT Dallas and beyond.\n" +
67                         "Mission Statement:\n" +
68                         "a) To further promote interest in comics, anime, manga, movies and video
                                games.\n" +
69                         "\n" +
70                         "b) To provide and create greater fellowship among students.\n" +
71                         "\n" +
72                         "c) To represent student needs and wants regarding comics, anime, manga,
                                movies and video games.\n" +
73                         "\n" +
74                         "d) To provide a forum for the presentation of innovative ideas to the
                                benefit of the University community.","pot"))
75
76          //Adds an organization to the database. This is the Pre-health Professions organization
77          db.addOrg(Organizations(4,"Pre-Health Professions Club","Laquze
                Morris","Laquzemorris@my.untdallas.edu",
78                 "972.338.1781",
79                 "The purpose of UNT Dallas Pre-Health Professions Society is to prepare pre-health
                        students for entry into their" +
80                         " respective graduate programs following their career at UNTD. Our goal is
                                to enhance not only our student members" +
81                         " likelihood to succeed in their health career, but create a community of
                                like-minded students that inspire and challenge" +
82                         " each other to achieve ambitious goals. To achieve this goal, we will use
                                several methods. We will give our members opportunities" +
83                         " to attend informative lecture-style meetings held by local health
```

```
                    professionals, current professional students, and science" +
84                  " professors focused on augmenting student pre-professional development. We
                        also will provide academic resources such as study" +
85                  " groups and preparatory materials for student taking graduate-level
                        entrance tests such as the MCAT, DAT, GRE, etc. In addition," +
86                  " we will organize community service events and shadowing opportunities with
                        the goal of enhancing students overall graduate" +
87                  " application attractiveness. ","art"))
88
89          //Adds an organization to the database. This is the Sigma Lambda Gamma National Sorority
                organization
90          db.addOrg(Organizations(5,"Sigma Lambda Gamma National Sorority, Inc. ","Jovanna
                Moreno","slguntd.president@gmail.com",
91              "972.338.1781",
92              "Recognizing our responsibility to the progression of a positive global community,
                    we stress the importance of morals, ethics," +
93                  " and education in our daily lives so that we serve the needs of our
                        neighbors through a mutual respect and understanding of" +
94                  " our varying cultures.","sig"))
95
96          //This is the variable that will hold the main listview
97          var simpleList: ListView
98
99          //This is the array to hold the Organization objects
100         var orgListing = arrayOf("")
101
102         //Counts the number of objects stored in the orgListing
103         var count = 0
104
105         //This list of Organizations gets filled with organization objects from the database.
106         var orgs: List<Organizations> = db.allOrganizations;
107
108         //This array gets all the objects from the orgs List.
109         //Arrays are easier to use to utilize orgsArray to access the objects
```

```
110         var orgsArray = orgs.toTypedArray()
111
112         //This loop deletes all the organizations within the database
113         //The objects are stored in the array so it is okay.
114         //Once the app is created again the organizations are re-inserted.
115         //This loops helps keep the organizations up to date.
116         for ( Org in orgs) {
117             db.deleteOrganization(Org)
118             count = count + 1
119         }
120
121         //This adapter is an object from another class that was created.
122         //This adapter will determine how each organization square is displayed in the list.
123         val adapter = customAdapter(this, 0, orgs)
124
125         //This is the listView of the main page.
126         simpleList = findViewById<View>(R.id.simpleListView) as ListView
127
128         //The adapter we previously created gets plugged-in to the listView
129         //The adapter can be changed to any other that is offered by android
130         //The customAdapter class can also be changed to display the list any other way
131         simpleList.setAdapter(adapter)
132
133         //This is the Listener that determines what happens when you click on an item in the
                listView
134         //The text within the function is grey due to using some redundant code, it is okay.
135         simpleList.setOnItemClickListener(AdapterView.OnItemClickListener { list, v, pos, id ->
136
137
138             //This variable finds the activity that will be called when you click on an item
139             val intentOne = Intent(applicationContext,organizationInformation::class.java)
140
141             //This is an individual object from the orgsArray.
142             //The object will be determined based on which item is clicked.
```

```
143             var obj = orgsArray[pos]
144
145             //An array is created that will store all the info from the selected object
146             var infoArray:Array<String> =
                    arrayOf(obj.entityID.toString(),obj.orgName,obj.orgDesc,obj.presName
147
148                        ,obj.presEmail,obj.presPhone,obj.orgPic)
149
150             //The array we created gets passed to the organizationInformation class
151             //We will access this array in the other class so we can have the same information
                    between classes
152             intentOne.putExtra("info",infoArray)
153
154             //We start the organizationInformation activity
155             startActivity(intentOne)
156
157         })
158
159     }
160
161 }
```

This is the XML code for the implementation of the Organizations feature.

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3
4 <androidx.constraintlayout.widget.ConstraintLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
5
6     xmlns:app="http://schemas.android.com/apk/res-auto"
7     xmlns:tools="http://schemas.android.com/tools"
8     android:layout_width="match_parent"
9     android:layout_height="match_parent"
```

```
    tools:context=".OrganizationsActivity">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <ListView
            android:id="@+id/simpleListView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:divider="@color/colorPrimary"
            android:dividerHeight="1dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            />

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

This is additional Kotlin code needed for the implementation of the Organizations feature.

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.appcompat.app.AlertDialog
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import android.R.string.cancel
```

```
9
10 import android.app.Dialog
11 import android.content.DialogInterface
12 import androidx.core.content.ContextCompat
13 import java.security.AccessController.getContext
14
15 class organizationInformation : AppCompatActivity() {
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18
19         super.onCreate(savedInstanceState)
20         setContentView(R.layout.activity_organization_information)
21
22         //This array will store all the Strings related to an object.
23         /*
24         Indexes for the moreInfo array. Remember, THEY ARE ALL STRINGS.
25         moreInfo[0] stores the entityID of the selected object as a String.
26         moreInfo[1] stores the name of the organization.
27         moreInfo[2] stores the description of the organization.
28         moreInfo[3] stores the name of the president of the organization.
29         moreInfo[4] stores the email of the organization.
30         moreInfo[5] stores the phone of the organization.
31         moreInfo[6] stores the name of the picture of the organization.
32
33          */
34         var moreInfo = intent.getStringArrayExtra("info")
35
36         //Stores the place where the president Name will be placed
37         var presName = findViewById<TextView>(R.id.presNameFill)
38
39         //Stores the place where the email will be placed
40         var email = findViewById<TextView>(R.id.presEmalFill)
41
42         //Stores the place where the phone will be placed
```

```
43          var phone = findViewById<TextView>(R.id.presPhoneFill)
44
45          //Stores the place where the picture of the organization will be placed
46          var pic = findViewById<ImageView>(R.id.orgImage)
47
48          //Sets the title to the name of the organization
49          setTitle(moreInfo[1])
50          presName.setText(moreInfo[3])
51          email.setText(moreInfo[4])
52          phone.setText(moreInfo[5])
53
54          //These statements manually assign a picture to pic depending on the picture name value.
55          //If you add a new entry to Organization do not forget to add a statement here to display
                images.
56          if(moreInfo[6].equals("it")){
57              pic.setImageResource(R.drawable.it)
58          }
59          else if(moreInfo[6].equals("bio")){
60              pic.setImageResource(R.drawable.bio)
61          }
62          else if(moreInfo[6].equals("chem")){
63              pic.setImageResource(R.drawable.chem)
64          }
65          else if(moreInfo[6].equals("pot")){
66              pic.setImageResource(R.drawable.pot)
67          }
68          else if(moreInfo[6].equals("art")){
69              pic.setImageResource(R.drawable.art)
70          }
71          else if(moreInfo[6].equals("sig")){
72              pic.setImageResource(R.drawable.sig)
73          }
74          else{
75              pic.setImageResource(R.drawable.ic_launcher_background)
```

```
        }

        //This button will display information related to an organzation once clicked.
        var infoButton = findViewById<Button>(R.id.descInfo)

        //Determines what will happen when the user click on the "Info" button.
        infoButton.setOnClickListener(){

            //An Alert box will be created
            val ad = AlertDialog.Builder(this).create()

            //The title of the alert box will be Description
            ad.setTitle("Description")

            //All the information within moreInfo[2] will be displayed on the alert box
            ad.setMessage(moreInfo[2])

            //Allows the user to manually close the alert box
            ad.setCancelable(true)

            //Allows user to close alert box and sets the button within to display "Ok"
            ad.setButton(Dialog.BUTTON_POSITIVE,"OK") { dialog, which -> finish() }

            //Shows the alert box
            ad.show()

        }

    }

}
```

This is additional XML code needed for the implementation of the Organizations feature.

```
1
2  <?xml version="1.0" encoding="utf-8"?>
3
4  <RelativeLayout
5      xmlns:android="http://schemas.android.com/apk/res/android"
6      xmlns:tools="http://schemas.android.com/tools"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent">
9
10     <FrameLayout
11         android:layout_width="match_parent"
12         android:layout_height="434dp"
13         android:layout_above="@+id/linear"
14         android:layout_marginBottom="10dp">
15
16         <ImageView
17             android:id="@+id/orgImage"
18             android:layout_width="401dp"
19             android:layout_height="match_parent"
20             android:layout_marginBottom="0dp"
21             android:scaleType="centerInside"
22             android:src="@drawable/ic_launcher_background"
23             tools:ignore="ContentDescription" />
24
25         <LinearLayout
26             android:layout_width="match_parent"
27             android:layout_height="match_parent"
28             android:gravity="bottom"
29             android:orientation="vertical">
30
31             <LinearLayout
32                 android:layout_width="match_parent"
33                 android:layout_height="wrap_content"
34                 android:baselineAligned="false"
```

```
35              android:orientation="horizontal">
36
37              <LinearLayout
38                  android:layout_width="match_parent"
39                  android:layout_height="match_parent"
40                  android:layout_marginBottom="20dp"
41                  android:layout_weight="1"
42                  android:orientation="vertical"></LinearLayout>
43          </LinearLayout>
44
45      </LinearLayout>
46
47  </FrameLayout>
48
49  <LinearLayout
50      android:id="@+id/linear"
51      android:layout_width="match_parent"
52      android:layout_height="250dp"
53      android:layout_alignParentBottom="true"
54      android:layout_marginBottom="0dp"
55      android:orientation="vertical">
56
57      <LinearLayout
58          android:layout_width="match_parent"
59          android:layout_height="wrap_content"
60          android:orientation="horizontal"
61          >
62
63          <TextView
64              android:id="@+id/presName"
65              android:layout_width="match_parent"
66              android:layout_height="wrap_content"
67              android:layout_weight="1"
68              android:background="#0000"
```

```
69              android:gravity="left"
70              android:maxLength="50"
71              android:padding="20dp"
72              android:text="Pres. Name"
73              android:textColor="#a2a1b8"
74              android:textSize="16dp"
75              />
76
77          <TextView
78              android:id="@+id/presNameFill"
79              android:layout_width="match_parent"
80              android:layout_height="wrap_content"
81              android:layout_weight="1"
82              android:background="#0000"
83              android:gravity="right"
84              android:maxLength="50"
85              android:padding="20dp"
86              android:text=""
87              android:textColor="#181737"
88              android:textColorHint="#181737"
89              android:textSize="16dp"
90              />
91
92      </LinearLayout>
93
94      <View
95          android:layout_width="match_parent"
96          android:layout_height="1dp"
97          android:background="#dad8d8" />
98
99      <LinearLayout
100         android:layout_width="match_parent"
101         android:layout_height="wrap_content"
102         android:orientation="horizontal"
```

```
103             >
104
105             <TextView
106                 android:id="@+id/presPhone"
107                 android:layout_width="match_parent"
108                 android:layout_height="wrap_content"
109                 android:layout_weight="1"
110                 android:background="#0000"
111                 android:gravity="left"
112                 android:maxLength="50"
113                 android:padding="20dp"
114                 android:text="Off. Phone"
115                 android:textColor="#a2a1b8"
116                 android:textSize="16dp"
117                 />
118
119             <TextView
120                 android:id="@+id/presPhoneFill"
121                 android:layout_width="match_parent"
122                 android:layout_height="wrap_content"
123                 android:layout_weight="1"
124                 android:background="#0000"
125                 android:gravity="right"
126                 android:maxLength="50"
127                 android:padding="20dp"
128                 android:text=""
129                 android:textColor="#181737"
130                 android:textColorHint="#1b193b"
131                 android:textSize="16dp"
132                 />
133
134         </LinearLayout>
135
136         <View
```

```
137             android:layout_width="match_parent"
138             android:layout_height="1dp"
139             android:background="#dad8d8" />
140
141         <LinearLayout
142             android:layout_width="match_parent"
143             android:layout_height="wrap_content"
144             android:orientation="horizontal"
145             >
146
147             <TextView
148                 android:id="@+id/presEmail"
149                 android:layout_width="match_parent"
150                 android:layout_height="wrap_content"
151                 android:layout_weight="1"
152                 android:background="#0000"
153                 android:gravity="left"
154                 android:maxLength="50"
155                 android:padding="20dp"
156                 android:text="Pres. Email"
157                 android:textColor="#a2a1b8"
158                 android:textSize="16dp"
159                 />
160
161             <!--com.rey.material.widget.Switch!-->
162
163             <TextView
164                 android:id="@+id/presEmalFill"
165                 android:layout_width="match_parent"
166                 android:layout_height="wrap_content"
167                 android:layout_weight="1"
168                 android:background="#0000"
169                 android:gravity="right"
170                 android:maxLength="50"
```

```
            android:padding="0dp"
            android:text=""
            android:textColor="#181737"
            android:textColorHint="#1b193b"
            android:textSize="16dp"
            />

    </LinearLayout>

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="#dad8d8" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        >

        <TextView
            android:id="@+id/orgDesc"
            android:layout_width="155dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#0000"
            android:gravity="left"
            android:maxLength="50"
            android:padding="20dp"
            android:text="Description"
            android:textColor="#a2a1b8"
            android:textSize="16dp" />

        <Button
```

```
                android:id="@+id/descInfo"
                android:layout_width="150dp"
                android:layout_height="35dp"
                android:layout_weight="1"
                android:background="@color/colorPrimaryDark"
                android:gravity="center"
                android:paddingHorizontal="20dp"
                android:text="Info"
                android:textColor="@color/white"
                android:textSize="16sp" />

        </LinearLayout>

    </LinearLayout>

</RelativeLayout>
```

A.4 Library

This is the Kotlin code for the implementation of the Library feature.

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class Library : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_library)
        setTitle("Library")
    }
}
```

This is the XML code for the implementation of the Library feature.

```
1
2  <?xml version="1.0" encoding="utf-8"?>
3  <androidx.constraintlayout.widget.ConstraintLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".Library">
9
10     <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
11         android:layout_width="fill_parent"
12         android:layout_height="wrap_content"
13         android:fillViewport="false"
14         android:orientation="vertical"
15         android:padding="10dp"
16         app:layout_constraintBottom_toBottomOf="parent"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent">
20
21         <LinearLayout
22             android:layout_width="match_parent"
23             android:layout_height="wrap_content"
24             android:orientation="vertical">
25
26             <ImageView
27                 android:id="@+id/imageView"
28                 android:layout_width="wrap_content"
29                 android:layout_height="200dp"
30                 android:scaleType="centerCrop"
31                 android:src="@drawable/library_front" />
32
```

```
33          <TextView
34              android:id="@+id/textView"
35              android:layout_width="match_parent"
36              android:layout_height="wrap_content"
37              android:gravity="center"
38              android:text="@string/title"
39              android:textAppearance="@style/TextAppearance.AppCompat.Display1" />
40
41          <TextView
42              android:id="@+id/textView2"
43              android:layout_width="match_parent"
44              android:layout_height="wrap_content"
45              android:text="@string/description"
46              android:textAppearance="?android:attr/textAppearanceSmall" />
47
48          <TextView
49              android:id="@+id/overflow"
50              android:layout_width="wrap_content"
51              android:layout_height="wrap_content"
52              android:layout_centerVertical="true"
53              android:layout_gravity="left"
54              android:gravity="center"
55              android:text="@string/desOverflow"
56              android:textAppearance="?android:attr/textAppearanceSmall" />
57
58
59          <TextView
60              android:id="@+id/textView3"
61              android:layout_width="wrap_content"
62              android:layout_height="wrap_content"
63              android:text="Hours:"
64              android:textAppearance="?android:attr/textAppearanceLarge" />
65
66          <TextView
```

```
67              android:id="@+id/textView4"
68              android:layout_width="match_parent"
69              android:layout_height="wrap_content"
70              android:gravity="center"
71              android:text="@string/hours"
72              android:textAppearance="?android:attr/textAppearanceSmall" />
73
74          <TextView
75              android:id="@+id/textView5"
76              android:layout_width="wrap_content"
77              android:layout_height="wrap_content"
78              android:text="Helpful Links:"
79              android:textAppearance="?android:attr/textAppearanceLarge" />
80
81          <TextView
82              android:id="@+id/link1"
83              android:layout_width="wrap_content"
84              android:layout_height="wrap_content"
85              android:autoLink="web"
86              android:padding="20px"
87              android:text="@string/link1"
88              android:textAppearance="?android:attr/textAppearanceSmall" />
89
90          <TextView
91              android:id="@+id/link2"
92              android:layout_width="wrap_content"
93              android:layout_height="wrap_content"
94              android:autoLink="web"
95              android:padding="20px"
96              android:text="@string/link2"
97              android:textAppearance="?android:attr/textAppearanceSmall" />
98
99          <TextView
100             android:id="@+id/link3"
```

```
101             android:layout_width="wrap_content"
102             android:layout_height="wrap_content"
103             android:autoLink="web"
104             android:padding="20px"
105             android:text="@string/link3"
106             android:textAppearance="?android:attr/textAppearanceSmall" />
107
108         <TextView
109             android:id="@+id/link4"
110             android:layout_width="wrap_content"
111             android:layout_height="wrap_content"
112             android:autoLink="web"
113             android:padding="20px"
114             android:text="@string/link4"
115             android:textAppearance="?android:attr/textAppearanceSmall" />
116
117         <TextView
118             android:id="@+id/link5"
119             android:layout_width="wrap_content"
120             android:layout_height="wrap_content"
121             android:autoLink="web"
122             android:padding="20px"
123             android:text="@string/link5"
124             android:textAppearance="?android:attr/textAppearanceSmall" />
125
126         <TextView
127             android:id="@+id/link6"
128             android:layout_width="wrap_content"
129             android:layout_height="wrap_content"
130             android:autoLink="web"
131             android:padding="20px"
132             android:text="@string/link6"
133             android:textAppearance="?android:attr/textAppearanceSmall" />
134
```

```
            <TextView
                android:id="@+id/link7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:autoLink="web"
                android:padding="20px"
                android:text="@string/link7"
                android:textAppearance="?android:attr/textAppearanceSmall" />
        </LinearLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.5 Maps

This is the Kotlin code for the implementation of the Maps feature.

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class Maps : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        setTitle("Maps")
    }
}
```

This is the XML code for the implementation of the Maps feature.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
 xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".Maps">

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <androidx.constraintlayout.widget.Guideline
                android:id="@+id/guideline4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintGuide_begin="205dp"
                app:layout_constraintHorizontal_bias="0.5"
                app:layout_constraintStart_toStartOf="parent" />

            <TextView
                android:id="@+id/textView16"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
37                  android:layout_marginTop="32dp"
38                  android:text="Dal 1 Floor 1"
39                  android:textAllCaps="true"
40                  android:textSize="18sp"
41                  android:textStyle="bold"
42                  app:layout_constraintEnd_toEndOf="parent"
43                  app:layout_constraintStart_toStartOf="parent"
44                  app:layout_constraintTop_toTopOf="parent" />
45
46              <com.github.chrisbanes.photoview.PhotoView
47                  android:id="@+id/Dal1F1"
48                  android:layout_width="wrap_content"
49                  android:layout_height="450dp"
50                  android:layout_marginTop="32dp"
51                  android:scaleType="centerCrop"
52                  android:src="@drawable/dallas_1_floor1"
53                  app:layout_constraintEnd_toEndOf="parent"
54                  app:layout_constraintHorizontal_bias="0.0"
55                  app:layout_constraintStart_toStartOf="parent"
56                  app:layout_constraintTop_toBottomOf="@+id/textView16" />
57
58              <TextView
59                  android:id="@+id/textView17"
60                  android:layout_width="wrap_content"
61                  android:layout_height="wrap_content"
62                  android:layout_marginTop="32dp"
63                  android:text="Dal 1 Floor 2"
64                  android:textAllCaps="true"
65                  android:textSize="18sp"
66                  android:textStyle="bold"
67                  app:layout_constraintEnd_toEndOf="parent"
68                  app:layout_constraintHorizontal_bias="0.5"
69                  app:layout_constraintStart_toStartOf="parent"
70                  app:layout_constraintTop_toBottomOf="@+id/Dal1F1" />
```

```
71
72              <com.github.chrisbanes.photoview.PhotoView
73                  android:id="@+id/Dal1F2"
74                  android:layout_width="wrap_content"
75                  android:layout_height="450dp"
76                  android:layout_marginTop="32dp"
77                  android:scaleType="centerCrop"
78                  android:src="@drawable/dallas_1_floor2"
79                  app:layout_constraintEnd_toEndOf="parent"
80                  app:layout_constraintStart_toStartOf="parent"
81                  app:layout_constraintTop_toBottomOf="@+id/textView17" />
82
83              <TextView
84                  android:id="@+id/textView9"
85                  android:layout_width="wrap_content"
86                  android:layout_height="wrap_content"
87                  android:layout_marginTop="32dp"
88                  android:text="Dal 1 Floor 3"
89                  android:textAllCaps="true"
90                  android:textSize="18sp"
91                  android:textStyle="bold"
92                  app:layout_constraintEnd_toEndOf="parent"
93                  app:layout_constraintHorizontal_bias="0.5"
94                  app:layout_constraintStart_toStartOf="parent"
95                  app:layout_constraintTop_toBottomOf="@+id/Dal1F2" />
96
97              <com.github.chrisbanes.photoview.PhotoView
98                  android:id="@+id/Dal1F3"
99                  android:layout_width="wrap_content"
100                 android:layout_height="450dp"
101                 android:layout_marginTop="32dp"
102                 android:scaleType="centerCrop"
103                 android:src="@drawable/dallas_1_floor3"
104                 app:layout_constraintEnd_toEndOf="parent"
```

```
105                 app:layout_constraintHorizontal_bias="0.0"
106                 app:layout_constraintStart_toStartOf="parent"
107                 app:layout_constraintTop_toBottomOf="@+id/textView9" />
108
109             <TextView
110                 android:id="@+id/textView18"
111                 android:layout_width="wrap_content"
112                 android:layout_height="27dp"
113                 android:layout_marginStart="170dp"
114                 android:layout_marginLeft="170dp"
115                 android:layout_marginTop="32dp"
116                 android:layout_marginEnd="170dp"
117                 android:layout_marginRight="170dp"
118                 android:text="Dal 2 Floor 1"
119                 android:textAllCaps="true"
120                 android:textSize="18sp"
121                 android:textStyle="bold"
122                 app:layout_constraintEnd_toEndOf="parent"
123                 app:layout_constraintStart_toStartOf="parent"
124                 app:layout_constraintTop_toBottomOf="@+id/Dal1F3" />
125
126             <com.github.chrisbanes.photoview.PhotoView
127                 android:id="@+id/Dal2F1"
128                 android:layout_width="wrap_content"
129                 android:layout_height="450dp"
130                 android:layout_marginTop="32dp"
131                 android:scaleType="centerCrop"
132                 android:src="@drawable/dallas_2_floor1"
133                 app:layout_constraintEnd_toEndOf="parent"
134                 app:layout_constraintStart_toStartOf="parent"
135                 app:layout_constraintTop_toBottomOf="@+id/textView18" />
136
137             <TextView
138                 android:id="@+id/textView19"
```

```
            android:layout_width="wrap_content"
            android:layout_height="27dp"
            android:layout_marginStart="170dp"
            android:layout_marginLeft="170dp"
            android:layout_marginTop="32dp"
            android:layout_marginEnd="170dp"
            android:layout_marginRight="170dp"
            android:text="Dal 2 Floor 2"
            android:textAllCaps="true"
            android:textSize="18sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/Dal2F1" />

        <com.github.chrisbanes.photoview.PhotoView
            android:id="@+id/Dal2F2"
            android:layout_width="wrap_content"
            android:layout_height="450dp"
            android:layout_marginTop="32dp"
            android:scaleType="centerCrop"
            android:src="@drawable/dallas_2_floor2"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView19" />

        <TextView
            android:id="@+id/textView20"
            android:layout_width="wrap_content"
            android:layout_height="27dp"
            android:layout_marginStart="170dp"
            android:layout_marginLeft="170dp"
            android:layout_marginTop="32dp"
```

```
                    android:layout_marginEnd="170dp"
                    android:layout_marginRight="170dp"
                    android:text="Dal 2 Floor 3"
                    android:textAllCaps="true"
                    android:textSize="18sp"
                    android:textStyle="bold"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintHorizontal_bias="0.5"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/Dal2F2" />

                <com.github.chrisbanes.photoview.PhotoView
                    android:id="@+id/Dal2F3"
                    android:layout_width="wrap_content"
                    android:layout_height="450dp"
                    android:layout_marginTop="32dp"
                    android:scaleType="centerCrop"
                    android:src="@drawable/dallas_2_floor3"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintHorizontal_bias="0.0"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/textView20" />

            </androidx.constraintlayout.widget.ConstraintLayout>
        </LinearLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.6 Social Media

This is the Kotlin code for the implementation of the Social Media feature.

```kotlin
import android.content.Intent
import android.content.pm.PackageManager
import android.content.pm.ResolveInfo
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_social_media.*

class socialMedia : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_social_media)

        title = "Social Media"

        //This variable will get the package name of the Instagram application in case it is
            installed.
        val activities: List<ResolveInfo> = packageManager.queryIntentActivities(
               intent,
               PackageManager.MATCH_DEFAULT_ONLY
        )

        //This is a null check and makes sure the application is installed.
        val isIntentSafe: Boolean = activities.isNotEmpty()

        //This variable will store the URL address of UNTD's Instagram account.
        val instagram: Intent = Uri.parse("https://www.instagram.com/untdallas/?hl=en").let {
            webpage ->

            Intent(Intent.ACTION_VIEW, webpage)
        }
```

```
33
34          //When this button is clicked Instagram will be launched if the application is available
35          instaButton.setOnClickListener {
36              if (isIntentSafe) {
37                  startActivity(instagram)
38              }
39          }
40
41          //This variable will store the URL address of UNTD's Facebook account.
42          val facebook: Intent = Uri.parse("https://www.facebook.com/UNTDallas/").let { webpage ->
43              Intent(Intent.ACTION_VIEW, webpage)
44          }
45
46          //When this button is clicked Facebook will be launched if the application is available
47          facebookApp.setOnClickListener{
48              if (isIntentSafe) {
49                  startActivity(facebook)
50              }
51
52          }
53
54          //This variable will store the URL address of UNTD's LinkedIn account.
55          val linkedin: Intent =
                Uri.parse("https://www.linkedin.com/school/university-of-north-texas-at-dallas/").let
                { webpage ->
56              Intent(Intent.ACTION_VIEW, webpage)
57          }
58
59          //When this button is clicked LinkedIn will be launched if the application is available
60          linkedInButton.setOnClickListener{
61              if (isIntentSafe) {
62                  startActivity(linkedin)
63              }
64
```

```
        }

        //This variable will store the URL address of UNTD's Twitter account.
        val twitter: Intent =
            Uri.parse("https://twitter.com/UNTDallas?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor").let
            { webpage ->
            Intent(Intent.ACTION_VIEW, webpage)
        }

        //When this button is clicked Twitter will be launched if the application is available
        twitterButton.setOnClickListener{
            if (isIntentSafe) {
                startActivity(twitter)
            }
        }

    }

}
```

This is the XML code for the implementation of the Social Media feature.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ToDoList">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_to_do_list" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

A.7 Directory

This is the Kotlin code for the implementation of the Directory feature.

```
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.AdapterView
import android.widget.ArrayAdapter
```

```
import android.widget.ListView

class DirectoryActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_directory)

        setTitle("Directory")

        //Creates an object from the DBHandler_v2 class.
        //This object is used to access the database. Use its methods to access the database.
        var db = DBHandler_v2(this);

        //Creates and adds Directory objects to the database.
        //In order to add an new entry simply add an entry. Don't forget about its image too.
        db.addDirec(Directory(0,"Saif","Al-sultan","SaifAlsultan@my.untdallas.edu",
                "456456456", "IT Professor", "FH 222","Alsultan"))

        db.addDirec(Directory(1,"Gerard","Rambally","GerardRambally@my.untdallas.edu",
                "456123456", "IT Professor", "FH 232","Rambally"))

        db.addDirec(Directory(2,"Richard","Chandler","RichardChandler@my.untdallas.edu",
                "456789900", "Math Professor", "FH 236","Chandler"))

        db.addDirec(Directory(3,"Farid","Hallouche","FaridHallouche@my.untdallas.edu",
                "456576394", "IT Professor", "FH 240","Hallouche"))


        //Gets all Directory objects stored within the database.
        var dirList = db.allDirectory

        //Turns dirList into an array while keeping the objects.
        var dirArray = dirList.toTypedArray()
```

```
40
41          //This array will be used to store the names of each entry.
42          var names = emptyArray<String>()
43
44          //Loops through each stored object and gets their first and last name.
45          //The full names are stored within array "names".
46          for(dir in dirArray){
47              var fullName = dir.fName + " " + dir.lName
48              names = names + fullName
49          }
50
51          for ( Org in dirList) {
52              db.deleteDirectory(Org)
53          }
54
55          //Creates the adapter for the listview of the activity. Uses the array "names" in its
                constructor.
56          var dirAdapter = ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,names)
57
58          //Listview for the entire activity.
59          var dirListView = findViewById<ListView>(R.id.directoryListView)
60
61          //Sets the adapter of the listview of the activity.
62          dirListView.adapter = dirAdapter
63
64          //Sets a listener used whenever an item in the listview is clicked.
65          //The "pos" value is used to determine which item is being clicked and used to access
                objects in the array.
66          dirListView.setOnItemClickListener(AdapterView.OnItemClickListener { list, v, pos, id ->
67
68              //This Intent stores a new activity that will be launched when an item is clicked.
69              val intentOne = Intent(applicationContext,directoryInformation::class.java)
70
71              //This variable stores a Directory object. This object will allow access to its
```

```
                variables.
72              //The Directory object comes from the array storing all Directory objects.
73              var obj = dirArray[pos]
74
75              //This array of strings stores all information related to the object selected.
76              var infoArray:Array<String> =
                    arrayOf(obj.entityID.toString(),obj.fName,obj.lName,obj.sEmail
77
78                      ,obj.sPhone,obj.sPosition,obj.sPicture)
79
80              //This statement uses the intent previously created in order to pass on an array.
81              //The array that is passed on will be used to display the information of the selected
                    item.
82              intentOne.putExtra("info",infoArray)
83
84              //The directoryInformation activity is launched.
85              startActivity(intentOne)
86
87          })
88
89      }
90  }
```

This is the XML code for the implementation of the Directory feature.

```
1
2   <?xml version="1.0" encoding="utf-8"?>
3
4   <androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
5
6       xmlns:app="http://schemas.android.com/apk/res-auto"
7       xmlns:tools="http://schemas.android.com/tools"
8       android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".DirectoryActivity">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <ListView
            android:id="@+id/directoryListView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:divider="@color/colorPrimary"
            android:dividerHeight="1dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            />

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

This is additional Kotlin code for the implementation of the Directory feature.

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ImageView
import android.widget.TextView
import org.w3c.dom.Text

class directoryInformation : AppCompatActivity() {
```

```
8
9       override fun onCreate(savedInstanceState: Bundle?) {
10          super.onCreate(savedInstanceState)
11          setContentView(R.layout.activity_directory_information)
12
13          //This array stores an array of strings previous created by DirectoryActivity.
14          //This array stores all information related to a specific item. Use its index to get
                information.
15          /*
16          For array "moreInfo":
17          moreInfo[0] = ID
18          moreInfo[1] = First name of item
19          moreInfo[2] = Last name of item
20          moreInfo[3] = Email of item
21          moreInfo[4] = Phone of item
22          moreInfo[5] = Position of item
23          moreInfo[6] = Picture related to item
24           */
25          var moreInfo = intent.getStringArrayExtra("info")
26
27          //This variables combines First and Last Name into a new variable.
28          var fullName = moreInfo[1] + " " + moreInfo[2]
29
30          setTitle(fullName)
31
32          //This TextView will display the Full Name of the item selected
33          val nameBox = findViewById<TextView>(R.id.FullNameFill)
34
35          nameBox.setText(fullName)
36
37          //This TextView will display the phonebox of the item selected
38          val phoneBox = findViewById<TextView>(R.id.OffPhoneFill)
39
40          phoneBox.setText(moreInfo[4])
```

```
41
42          //This TextView will display the email of the item selected
43          val emailBox = findViewById<TextView>(R.id.EmalFill)
44
45          emailBox.setText(moreInfo[3])
46
47          //This TextView will display the position of the item selected
48          val positionBox = findViewById<TextView>(R.id.positionFieldFill)
49
50          positionBox.setText(moreInfo[5])
51
52          //This TextView will display the room number of the item selected
53          val roomBox = findViewById<TextView>(R.id.roomFieldFill)
54
55          roomBox.setText(moreInfo[6])
56
57          //This ImageView will display the image related to the item selected
58          val picBox = findViewById<ImageView>(R.id.dirImage)
59
60          //These statements manually assign a picture to picBox depending on the ID number.
61          //If you add a new entry to directory do not forget to add a statement here to display
                images
62          if(moreInfo[0].equals("0")){
63              picBox.setImageResource(R.drawable.alsultan)
64          }
65          else if(moreInfo[0].equals("1")){
66              picBox.setImageResource(R.drawable.rambally)
67          }
68          else if(moreInfo[0].equals("2")){
69              picBox.setImageResource(R.drawable.chandler)
70          }
71          else if(moreInfo[0].equals("3")){
72              picBox.setImageResource(R.drawable.hall)
73          }
```

```
    }
}
```

This is additional XML code for the implementation of the Directory feature.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/linear"
        >

        <ImageView
            android:id="@+id/dirImage"
            android:layout_width="match_parent"
            android:layout_height="400dp"
            android:layout_marginBottom="0dp"
            android:scaleType="fitCenter"
            android:src="@drawable/ic_launcher_background"
            tools:ignore="ContentDescription" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="295dp"
            android:gravity="bottom"
            android:orientation="vertical">
```

```
30
31              <LinearLayout
32                  android:layout_width="match_parent"
33                  android:layout_height="wrap_content"
34                  android:orientation="horizontal"
35                  android:baselineAligned="false">
36
37                  <LinearLayout
38                      android:layout_width="match_parent"
39                      android:layout_height="match_parent"
40                      android:layout_marginBottom="20dp"
41                      android:layout_weight="1"
42                      android:orientation="vertical"></LinearLayout>
43              </LinearLayout>
44
45          </LinearLayout>
46
47      </FrameLayout>
48
49      <LinearLayout
50          android:id="@+id/linear"
51          android:layout_width="match_parent"
52          android:layout_height="wrap_content"
53          android:orientation="vertical"
54          android:layout_alignParentBottom="true">
55
56          <LinearLayout
57              android:layout_width="match_parent"
58              android:layout_height="wrap_content"
59              android:orientation="horizontal"
60              >
61
62              <TextView
63                  android:layout_weight="1"
```

```
64              android:id="@+id/FullName"
65              android:textSize="16dp"
66              android:layout_width="match_parent"
67              android:layout_height="wrap_content"
68              android:text="Name"
69              android:textColor="#a2a1b8"
70              android:maxLength="50"
71              android:background="#0000"
72              android:padding="20dp"
73              android:gravity="left"
74              />
75
76          <TextView
77              android:layout_weight="1"
78              android:id="@+id/FullNameFill"
79              android:textSize="16dp"
80              android:layout_width="match_parent"
81              android:layout_height="wrap_content"
82              android:text=""
83              android:textColorHint="#181737"
84              android:textColor="#181737"
85              android:maxLength="50"
86              android:background="#0000"
87              android:padding="20dp"
88              android:gravity="right"
89              />
90
91      </LinearLayout>
92
93      <View
94          android:layout_width="match_parent"
95          android:layout_height="1dp"
96          android:background="#dad8d8"/>
97
```

```
98          <LinearLayout
99              android:layout_width="match_parent"
100             android:layout_height="wrap_content"
101             android:orientation="horizontal"
102             >
103
104             <TextView
105                 android:layout_weight="1"
106                 android:id="@+id/Off.Phone"
107                 android:textSize="16dp"
108                 android:layout_width="match_parent"
109                 android:layout_height="wrap_content"
110                 android:text="Off. Phone"
111                 android:textColor="#a2a1b8"
112                 android:maxLength="50"
113                 android:background="#0000"
114                 android:padding="20dp"
115                 android:gravity="left"
116                 />
117
118             <TextView
119                 android:layout_weight="1"
120                 android:id="@+id/OffPhoneFill"
121                 android:textSize="16dp"
122                 android:layout_width="match_parent"
123                 android:layout_height="wrap_content"
124                 android:textColorHint="#1b193b"
125                 android:text=""
126                 android:textColor="#181737"
127                 android:maxLength="50"
128                 android:background="#0000"
129                 android:padding="20dp"
130                 android:gravity="right"
131                 />
```

```
132
133         </LinearLayout>
134
135         <View
136             android:layout_width="match_parent"
137             android:layout_height="1dp"
138             android:background="#dad8d8"/>
139
140         <LinearLayout
141             android:layout_width="match_parent"
142             android:layout_height="wrap_content"
143             android:orientation="horizontal"
144             >
145
146             <TextView
147                 android:layout_weight="1"
148                 android:id="@+id/presEmail"
149                 android:textSize="16dp"
150                 android:layout_width="match_parent"
151                 android:layout_height="wrap_content"
152                 android:text="Email"
153                 android:textColor="#a2a1b8"
154                 android:maxLength="50"
155                 android:background="#0000"
156                 android:padding="20dp"
157                 android:gravity="left"
158                 />
159
160             <!--com.rey.material.widget.Switch!-->
161
162             <TextView
163                 android:layout_weight="1"
164                 android:id="@+id/EmalFill"
165                 android:textSize="16dp"
```

```
166             android:layout_width="match_parent"
167             android:layout_height="wrap_content"
168             android:textColorHint="#1b193b"
169             android:text=""
170             android:textColor="#181737"
171             android:maxLength="50"
172             android:background="#0000"
173             android:padding="0dp"
174             android:gravity="right"
175             />
176
177     </LinearLayout>
178
179     <View
180         android:layout_width="match_parent"
181         android:layout_height="1dp"
182         android:background="#dad8d8"/>
183
184     <LinearLayout
185         android:layout_width="match_parent"
186         android:layout_height="wrap_content"
187         android:orientation="horizontal"
188         >
189
190         <TextView
191             android:layout_weight="1"
192             android:id="@+id/positionField"
193             android:textSize="16dp"
194             android:layout_width="match_parent"
195             android:layout_height="wrap_content"
196             android:text="Position"
197             android:textColor="#a2a1b8"
198             android:maxLength="50"
199             android:background="#0000"
```

```
            android:padding="20dp"
            android:gravity="left"

            />

        <TextView
            android:layout_weight="1"
            android:id="@+id/positionFieldFill"
            android:textSize="16dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColorHint="#1b193b"
            android:text=""
            android:textColor="#181737"
            android:maxLength="50"
            android:background="#0000"
            android:padding="20dp"
            android:gravity="right"
            />

    </LinearLayout>

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="#dad8d8"/>



    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        >
```

```
234
235             <TextView
236                 android:layout_weight="1"
237                 android:id="@+id/roomField"
238                 android:textSize="16dp"
239                 android:layout_width="match_parent"
240                 android:layout_height="wrap_content"
241                 android:text="Room"
242                 android:textColor="#a2a1b8"
243                 android:maxLength="50"
244                 android:background="#0000"
245                 android:padding="20dp"
246                 android:gravity="left"
247                 />
248
249             <TextView
250                 android:layout_weight="1"
251                 android:id="@+id/roomFieldFill"
252                 android:textSize="16dp"
253                 android:layout_width="match_parent"
254                 android:layout_height="wrap_content"
255                 android:textColorHint="#1b193b"
256                 android:text=""
257                 android:textColor="#181737"
258                 android:maxLength="50"
259                 android:background="#0000"
260                 android:padding="20dp"
261                 android:gravity="right"
262                 />
263
264         </LinearLayout>
265
266     </LinearLayout>
267
```

```
</RelativeLayout>
```

A.8 Dart

This is the Kotlin code for the implementation of the Dart feature.

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class Dart : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dart)
        setTitle("Dart Schedule")
    }
}
```

This is the XML code for the implementation of the Dart feature.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Dart">

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
```

```
14          android:padding="10dp"
15          android:fillViewport="false"
16          app:layout_constraintTop_toTopOf="parent"
17          app:layout_constraintBottom_toBottomOf="parent"
18          android:layout_marginBottom="59dp"
19          app:layout_constraintStart_toStartOf="parent"
20          app:layout_constraintEnd_toEndOf="parent">
21
22          <LinearLayout
23              android:layout_width="match_parent"
24              android:layout_height="wrap_content"
25              android:orientation="vertical">
26
27              <ImageView
28                  android:id="@+id/imageView"
29                  android:layout_width="match_parent"
30                  android:layout_height="156dp"
31                  android:scaleType="centerCrop"
32                  android:src="@drawable/dart_station" />
33
34              <TextView
35                  android:id="@+id/textView"
36                  android:layout_width="match_parent"
37                  android:layout_height="wrap_content"
38                  android:gravity="center"
39                  android:text="@string/titleDart"
40                  android:textAppearance="@style/TextAppearance.AppCompat.Display1"
41                  android:textColor="@android:color/black"
42                  android:textStyle="bold"/>
43
44              <TextView
45                  android:id="@+id/textView2"
46                  android:layout_width="match_parent"
47                  android:layout_height="wrap_content"
```

```
48              android:text="@string/descriptionDart"
49              android:textAppearance="?android:attr/textAppearanceSmall"
50              android:textColorLink="@color/colorPrimary"/>
51
52          <TextView
53              android:id="@+id/textView3"
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:text="Hours:"
57              android:textAppearance="?android:attr/textAppearanceLarge"
58              android:textStyle="bold"/>
59          <TextView
60              android:id="@+id/textView4"
61              android:layout_width="match_parent"
62              android:layout_height="wrap_content"
63              android:gravity="left"
64              android:text="@string/hoursDart"
65              android:textAppearance="?android:attr/textAppearanceSmall"/>
66          <TextView
67              android:id="@+id/textView5"
68              android:layout_width="wrap_content"
69              android:layout_height="wrap_content"
70              android:text="Helpful Links:"
71              android:textAppearance="?android:attr/textAppearanceLarge"
72              android:textStyle="bold"/>
73
74          <TextView
75              android:id="@+id/link1"
76              android:layout_width="wrap_content"
77              android:layout_height="wrap_content"
78              android:autoLink="web"
79              android:padding="20px"
80              android:text="@string/link20"
81              android:textAppearance="?android:attr/textAppearanceSmall"
```

```
82              android:textColorLink="@color/colorPrimary"/>
83          <TextView
84              android:id="@+id/link2"
85              android:layout_width="wrap_content"
86              android:layout_height="wrap_content"
87              android:autoLink="web"
88              android:padding="20px"
89              android:text="@string/link21"
90              android:textAppearance="?android:attr/textAppearanceSmall"
91              android:textColorLink="@color/colorPrimary"/>
92          <TextView
93              android:id="@+id/link3"
94              android:layout_width="wrap_content"
95              android:layout_height="wrap_content"
96              android:autoLink="web"
97              android:padding="20px"
98              android:text="@string/link22"
99              android:textAppearance="?android:attr/textAppearanceSmall"
100             android:textColorLink="@color/colorPrimary"/>
101         <TextView
102             android:id="@+id/link4"
103             android:layout_width="wrap_content"
104             android:layout_height="wrap_content"
105             android:autoLink="web"
106             android:padding="20px"
107             android:text="@string/link23"
108             android:textAppearance="?android:attr/textAppearanceSmall"
109             android:textColorLink="@color/colorPrimary"/>
110
111         <TextView
112             android:id="@+id/textView6"
113             android:layout_width="wrap_content"
114             android:layout_height="wrap_content"
115             android:text="DART Police Presence:"
```

```
116                 android:textAppearance="?android:attr/textAppearanceLarge"
117                 android:textStyle="bold"/>
118             <TextView
119                 android:id="@+id/DARTPolice"
120                 android:text="@string/DARTPolice"
121                 android:layout_width="wrap_content"
122                 android:layout_height="wrap_content"
123                 android:gravity="left"
124                 android:layout_gravity="left"
125                 android:layout_centerVertical="true"
126                 android:textAppearance="?android:attr/textAppearanceSmall"/>
127
128             <TextView
129                 android:id="@+id/textView7"
130                 android:layout_width="wrap_content"
131                 android:layout_height="wrap_content"
132                 android:text="Courtesy Shuttle:"
133                 android:textAppearance="?android:attr/textAppearanceLarge"
134                 android:textStyle="bold"/>
135
136             <TextView
137                 android:id="@+id/textView8"
138                 android:layout_width="wrap_content"
139                 android:layout_height="wrap_content"
140                 android:text="JAG Line Shuttle:"
141                 android:textAppearance="?android:attr/textAppearanceMedium"
142                 android:textColor="@android:color/black"/>
143             <TextView
144                 android:id="@+id/CourtesyShuttle"
145                 android:text="@string/CourtesyShuttle"
146                 android:layout_width="wrap_content"
147                 android:layout_height="wrap_content"
148                 android:gravity="left"
149                 android:layout_gravity="left"
```

```
150             android:layout_centerVertical="true"
151             android:textAppearance="?android:attr/textAppearanceSmall"/>
152         <TextView
153             android:id="@+id/JAGLine"
154             android:text="@string/JAGShuttleStops"
155             android:layout_width="wrap_content"
156             android:layout_height="wrap_content"
157             android:gravity="left"
158             android:layout_gravity="left"
159             android:layout_centerVertical="true"
160             android:textAppearance="?android:attr/textAppearanceSmall"/>
161
162         <TextView
163             android:id="@+id/textView9"
164             android:layout_width="wrap_content"
165             android:layout_height="wrap_content"
166             android:text="CUB Line Shuttle:"
167             android:textAppearance="?android:attr/textAppearanceMedium"
168             android:textColor="@android:color/black"/>
169         <TextView
170             android:id="@+id/CUBLine"
171             android:text="@string/CUBShuttleStops"
172             android:layout_width="wrap_content"
173             android:layout_height="wrap_content"
174             android:gravity="left"
175             android:layout_gravity="left"
176             android:layout_centerVertical="true"
177             android:textAppearance="?android:attr/textAppearanceSmall"/>
178
179         <TextView
180             android:id="@+id/textView10"
181             android:layout_width="wrap_content"
182             android:layout_height="wrap_content"
183             android:text="DART Passes:"
```

```
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:textStyle="bold"/>
            <TextView
                android:id="@+id/DARTPasses"
                android:text="@string/DARTPasses"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="left"
                android:layout_gravity="left"
                android:layout_centerVertical="true"
                android:textAppearance="?android:attr/textAppearanceSmall"/>

            <TextView
                android:id="@+id/link5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:autoLink="web"
                android:padding="20px"
                android:text="@string/link24"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textColorLink="@color/colorPrimary"/>

        </LinearLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.9 Additional Code

This section presents additional code needed for the implementation of the system.

```

import android.annotation.SuppressLint;
```

```
3
4         import android.content.ContentValues;
5         import android.content.Context;
6         import android.database.Cursor;
7         import android.database.sqlite.SQLiteDatabase;
8         import android.database.sqlite.SQLiteOpenHelper;
9         import java.util.ArrayList;
10        import java.util.List;
11
12 public class DBHandler_v2 extends SQLiteOpenHelper{
13
14     private static final int DATABASE_VERSION = 1;
15     private static final String DATABASE_NAME = "Capstone";
16
17     // Organizations Table Columns names
18     private static final String TABLE_ORGANIZATIONS = "Organizations";
19     private static final String ENTITY_ID = "entityID";
20     private static final String ORG_NAME = "orgName";
21     private static final String PRES_NAME = "presName";
22     private static final String PRES_EMAIL = "presEmail";
23     private static final String PRES_PHONE = "presPhone";
24     private static final String ORG_DESC = "orgDesc";
25     private static final String ORG_PIC = "orgPic";
26
27     // Directory Table Columns names
28     private static final String TABLE_DIRECTORY = "Directory";
29     private static final String ENTITY_ID_DIRECTORY = "entityID";
30     private static final String SF_NAME = "FName";
31     private static final String SL_NAME = "LName";
32     private static final String S_EMAIL = "SEmail";
33     private static final String S_PHONE = "SPhone";
34     private static final String S_POSITION = "SPosition";
35     private static final String S_ROOM = "roomLocation";
36     private static final String S_PIC = "SPicture";
```

```
37
38      //Cafeteria Table Columns Names
39      private static final String TABLE_CAFETERIA = "Cafeteria";
40      private static final String FOOD_TYPE = "foodType";
41      private static final String FOOD_ID = "foodID";
42      private static final String FOOD_DESCRIPTION = "foodDescription";
43      public DBHandler_v2(Context context) {
44
45          super(context, DATABASE_NAME, null, DATABASE_VERSION);
46
47      }
48
49      @Override
50      public void onCreate(SQLiteDatabase db) {
51
52          //Creates the Organizations Table
53          String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_ORGANIZATIONS + "("
54                  + ENTITY_ID + " INTEGER PRIMARY KEY," + ORG_NAME + " TEXT,"
55                  + PRES_NAME + " TEXT," + PRES_EMAIL + " TEXT,"
56                  + PRES_PHONE + " TEXT," + ORG_DESC + " TEXT,"
57                  + ORG_PIC + " TEXT" + ")";
58          db.execSQL(CREATE_CONTACTS_TABLE);
59
60          //Creates the Directory Table
61          String CREATE_DIRECTORY_TABLE = "CREATE TABLE " + TABLE_DIRECTORY + "("
62                  + ENTITY_ID_DIRECTORY + " INTEGER PRIMARY KEY," + SF_NAME + " TEXT,"
63                  + SL_NAME + " TEXT," + S_EMAIL + " TEXT,"
64                  + S_PHONE + " TEXT," + S_POSITION + " TEXT,"
65                  + S_PIC + " TEXT,"
66                  + S_ROOM + " TEXT" + ")";
67
68          db.execSQL(CREATE_DIRECTORY_TABLE);
69          String CREATE_CAFETERIA_TABLE = "CREATE TABLE " + TABLE_CAFETERIA + "("
70                  + FOOD_ID + " INTEGER PRIMARY KEY," + FOOD_TYPE + " TEXT,"
```

```
71                 + FOOD_DESCRIPTION + " TEXT)";
72
73         db.execSQL(CREATE_CAFETERIA_TABLE);
74     }
75
76     @Override
77     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
78
79 // Drop older tables if existed
80         db.execSQL("DROP TABLE IF EXISTS " + TABLE_ORGANIZATIONS);
81         db.execSQL("DROP TABLE IF EXISTS " + TABLE_DIRECTORY);
82         db.execSQL("DROP TABLE IF EXISTS " + TABLE_CAFETERIA);
83 // Creating tables again
84         onCreate(db);
85
86     }
87
88     ////////////////////////////////////////////////////////////////////////////////
89     //The methods related to the Organization Table start here.
90     // Adding new organization
91     public void addOrg(Organizations org) {
92
93         SQLiteDatabase db = this.getWritableDatabase();
94         ContentValues values = new ContentValues();
95         //Inserting columns and their values to a values variable
96         values.put(ENTITY_ID, org.getEntityID()); //!!!!!!!!!!!!!
97         values.put(ORG_NAME, org.getOrgName());
98         values.put(PRES_NAME, org.getPresName());
99         values.put(PRES_EMAIL, org.getPresEmail());
100        values.put(PRES_PHONE, org.getPresPhone());
101        values.put(ORG_DESC, org.getOrgDesc());
102        values.put(ORG_PIC, org.getOrgPic());
103        // Inserting Row
104        db.insert(TABLE_ORGANIZATIONS, null, values);
```

```
105         db.close(); // Closing database connection
106 
107     }
108 
109     // Getting one Organization
110     public Organizations getOrganization(int id) {
111 
112         SQLiteDatabase db = this.getReadableDatabase();
113         @SuppressLint("Recycle") Cursor cursor = db.query(TABLE_ORGANIZATIONS, new String[] {
                ENTITY_ID,
114                     ORG_NAME, PRES_NAME, PRES_EMAIL,PRES_PHONE,ORG_DESC,ORG_PIC }, ENTITY_ID +
                        "=?",
115                 new String[] { String.valueOf(id) }, null, null, null, null);
116 
117         if (cursor != null)
118 
119             cursor.moveToFirst();
120         // return Organization
121         assert cursor != null;
122         return new Organizations(Integer.parseInt(cursor.getString(0)),
123                 cursor.getString(1), cursor.getString(2),cursor.getString(3),
124                 cursor.getString(4),cursor.getString(5),cursor.getString(6));
125 
126     }
127 
128     // Getting All Organizations
129     public List<Organizations> getAllOrganizations() {
130 
131         List<Organizations> orgList = new ArrayList<Organizations>();
132 // Select All Query
133         String selectQuery = "SELECT * FROM " + TABLE_ORGANIZATIONS;
134         SQLiteDatabase db = this.getWritableDatabase();
135         @SuppressLint("Recycle") Cursor cursor = db.rawQuery(selectQuery, null);
136 // looping through all rows and adding to list
```

```
137         if (cursor.moveToFirst()) {
138
139             do {
140
141                 Organizations org = new Organizations();
142                 org.setEntityID(Integer.parseInt(cursor.getString(0)));
143                 org.setOrgName(cursor.getString(1));
144                 org.setPresName(cursor.getString(2));
145                 org.setPresEmail(cursor.getString(3));
146                 org.setPresPhone(cursor.getString(4));
147                 org.setOrgDesc(cursor.getString(5));
148                 org.setOrgPic(cursor.getString(6));
149 // Adding contact to list
150                 orgList.add(org);
151
152             } while (cursor.moveToNext());
153
154         }
155
156 // return contact list
157         return orgList;
158
159     }
160
161     // Getting Organizations Count
162     public int getOrganizationsCount() {
163
164         String countQuery = "SELECT * FROM " + TABLE_ORGANIZATIONS;
165         SQLiteDatabase db = this.getReadableDatabase();
166         Cursor cursor = db.rawQuery(countQuery, null);
167         cursor.close();
168 // return count
169         return cursor.getCount();
170
```

```
171     }
172
173     // Updating an organization
174     public int updateOrganization(Organizations org) {
175
176         SQLiteDatabase db = this.getWritableDatabase();
177         ContentValues values = new ContentValues();
178         values.put(ORG_NAME, org.getOrgName());
179         values.put(PRES_NAME, org.getPresName());
180         values.put(PRES_EMAIL, org.getPresEmail());
181         values.put(PRES_PHONE, org.getPresPhone());
182         values.put(ORG_DESC, org.getOrgDesc());
183         values.put(ORG_PIC, org.getOrgPic());
184 // updating row
185         return db.update(TABLE_ORGANIZATIONS, values, ENTITY_ID + " = ?",
186                 new String[]{String.valueOf(org.getEntityID())});
187
188     }
189
190     // Deleting an organization
191     public void deleteOrganization(Organizations org) {
192
193         SQLiteDatabase db = this.getWritableDatabase();
194         db.delete(TABLE_ORGANIZATIONS, ENTITY_ID + " = ?",
195                 new String[] { String.valueOf(org.getEntityID()) });
196         db.close();
197
198     }
199
200     ////////////////////////////////////////////////////////////////////////////
201     //The methods related to the Directory Table start here.
202     public void addDirec(Directory dir) {
203
204         SQLiteDatabase db = this.getWritableDatabase();
```

```
205         ContentValues values = new ContentValues();
206         //Inserting columns and their values to a values variable
207         values.put(ENTITY_ID_DIRECTORY, dir.getEntityID()); //!!!!!!!!!!!!!
208         values.put(SF_NAME, dir.getFName());
209         values.put(SL_NAME, dir.getLName());
210         values.put(S_EMAIL, dir.getSEmail());
211         values.put(S_PHONE, dir.getSPhone());
212         values.put(S_POSITION, dir.getSPosition());
213         values.put(S_ROOM, dir.getSRoom());
214         values.put(S_PIC, dir.getSPicture());
215         // Inserting Row
216         db.insert(TABLE_DIRECTORY, null, values);
217         db.close(); // Closing database connection
218
219     }
220
221     // Getting one Directory
222     public Directory getDirectory(int id) {
223
224         SQLiteDatabase db = this.getReadableDatabase();
225         @SuppressLint("Recycle") Cursor cursor = db.query(TABLE_DIRECTORY, new String[] {
                ENTITY_ID_DIRECTORY,
226                         SF_NAME, SL_NAME, S_EMAIL,S_PHONE,S_POSITION,S_ROOM,S_PIC },
                            ENTITY_ID_DIRECTORY + "=?",
227                 new String[] { String.valueOf(id) }, null, null, null, null);
228
229         if (cursor != null)
230             cursor.moveToFirst();
231         // return Organization
232         assert cursor != null;
233         return new Directory(Integer.parseInt(cursor.getString(0)),
234                 cursor.getString(1), cursor.getString(2),cursor.getString(3),
235                 cursor.getString(4),cursor.getString(5),cursor.getString(6),
236                 cursor.getString(7));
```

```
237
238     }
239
240     // Getting All Organizations
241     public List<Directory> getAllDirectory() {
242
243         List<Directory> dirList = new ArrayList<Directory>();
244 // Select All Query
245         String selectQuery = "SELECT * FROM " + TABLE_DIRECTORY;
246         SQLiteDatabase db = this.getWritableDatabase();
247         @SuppressLint("Recycle") Cursor cursor = db.rawQuery(selectQuery, null);
248 // looping through all rows and adding to list
249         if (cursor.moveToFirst()) {
250
251             do {
252
253                 Directory dir = new Directory();
254                 dir.setEntityID(Integer.parseInt(cursor.getString(0)));
255                 dir.setFName(cursor.getString(1));
256                 dir.setLName(cursor.getString(2));
257                 dir.setSEmail(cursor.getString(3));
258                 dir.setSPhone(cursor.getString(4));
259                 dir.setSPosition(cursor.getString(5));
260                 dir.setSRoom(cursor.getString(6));
261                 dir.setSPicture(cursor.getString(7));
262 // Adding contact to list
263                 dirList.add(dir);
264
265             } while (cursor.moveToNext());
266
267         }
268
269 // return contact list
270         return dirList;
```

```
271     }
272
273     // Getting Organizations Count
274
275     public int getDirectoryCount() {
276
277         String countQuery = "SELECT * FROM " + TABLE_DIRECTORY;
278         SQLiteDatabase db = this.getReadableDatabase();
279         Cursor cursor = db.rawQuery(countQuery, null);
280         cursor.close();
281 // return count
282         return cursor.getCount();
283
284     }
285
286     // Updating an organization
287     public int updateDirectory(Directory dir) {
288
289         SQLiteDatabase db = this.getWritableDatabase();
290         ContentValues values = new ContentValues();
291         values.put(SF_NAME, dir.getFName());
292         values.put(SL_NAME, dir.getLName());
293         values.put(S_EMAIL, dir.getSEmail());
294         values.put(S_PHONE, dir.getSPhone());
295         values.put(S_POSITION, dir.getSPosition());
296         values.put(S_ROOM, dir.getSRoom());
297         values.put(S_PIC, dir.getSPicture());
298 // updating row
299         return db.update(TABLE_DIRECTORY, values, ENTITY_ID_DIRECTORY + " = ?",
300                 new String[]{String.valueOf(dir.getEntityID())});
301
302     }
303
304     // Deleting an organization
```

```
305     public void deleteDirectory(Directory dir) {
306
307         SQLiteDatabase db = this.getWritableDatabase();
308         db.delete(TABLE_DIRECTORY, ENTITY_ID_DIRECTORY + " = ?",
309                 new String[] { String.valueOf(dir.getEntityID()) });
310         db.close();
311
312     }
313
314 ////////////////////////////////////////////////////////////////////////////////////////////////////
315
316     //Methods for the Cafeteria Table start here
317
318     public Cafeteria addCafeteria(Cafeteria cafe) {
319
320         Cafeteria returnCafe = cafe;
321         SQLiteDatabase db = this.getWritableDatabase();
322         ContentValues values = new ContentValues();
323
324         //Inserting columns and their values to a values variable
325         values.put(FOOD_ID, cafe.getFoodID()); //!!!!!!!!!!!!!
326         values.put(FOOD_TYPE, cafe.getFoodType());
327         values.put(FOOD_DESCRIPTION, cafe.getFoodDescription());
328         // Inserting Row
329         db.insert(TABLE_CAFETERIA, null, values);
330         db.close(); // Closing database connection
331         return returnCafe;
332
333     }
334
335     public Cafeteria getCafeteria(int id) {
336
337         SQLiteDatabase db = this.getReadableDatabase();
338         @SuppressLint("Recycle") Cursor cursor = db.query(TABLE_CAFETERIA, new String[] { FOOD_ID,
```

```
                    FOOD_TYPE,FOOD_DESCRIPTION }, FOOD_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);

        if (cursor != null)
            cursor.moveToFirst();
        // return Organization
        assert cursor != null;
        return new Cafeteria(Integer.parseInt(cursor.getString(0)),
                cursor.getString(1), cursor.getString(2));

    }

    public int getCafeteriaCount() {

        String countQuery = "SELECT * FROM " + TABLE_CAFETERIA;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();
// return count
        return cursor.getCount();
    }

    // Updating an organization
    public int updateCafeteria(Cafeteria cafe) {

        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(FOOD_TYPE, cafe.getFoodType());
        values.put(FOOD_DESCRIPTION, cafe.getFoodDescription());
// updating row
        return db.update(TABLE_CAFETERIA, values, FOOD_ID + " = ?",
                new String[]{String.valueOf(cafe.getFoodID())});

    }
```

```
373
374     // Deleting an organization
375     public void deleteCafeteria(Cafeteria cafe) {
376
377         SQLiteDatabase db = this.getWritableDatabase();
378         String statement = "DELETE FROM "+TABLE_CAFETERIA;
379         db.execSQL(statement);
380         db.close();
381
382     }
383
384 }
```

```
1
2  import android.annotation.SuppressLint;
3  import android.app.Activity;
4  import android.content.Context;
5  import androidx.annotation.NonNull;
6  import android.view.LayoutInflater;
7  import android.view.View;
8  import android.view.ViewGroup;
9  import android.widget.ArrayAdapter;
10 import android.widget.ImageView;
11 import android.widget.TextView;
12 import java.util.List;
13
14 /*This class is used to create a custom adapter which allows you to modify
15 however a listview will display your elements. This customAdapter will only
16 work with the listview used on the Organizations activity.
17  */
18
19 public class customAdapter extends ArrayAdapter<Organizations> {
20
```

```
21      private Context context;
22      private List<Organizations> orgList;
23      //The constructor receives a list of objects when created.
24      //Will only work for the Organizations activity.
25      public customAdapter(Context context, int resource, List<Organizations> objects) {
26
27          super(context, resource, objects);
28          this.context = context;
29          this.orgList = objects;
30
31      }
32
33      @NonNull
34      public View getView(int position, View convertView, @NonNull ViewGroup parent) {
35
36          //Gets the organization that will be displayed based on the position within the array.
37          Organizations orgOne = orgList.get(position);
38
39          //Get the inflater and inflate the XML layout for each item
40
41          LayoutInflater inflater = (LayoutInflater)
                context.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
42
43          @SuppressLint({"ViewHolder", "InflateParams"}) View view =
                inflater.inflate(R.layout.custom_layout, null);
44
45          //Each item within the listview will have an image.
46          ImageView image = (ImageView) view.findViewById(R.id.icon);
47
48          //Each item within the listview will have a textview to display its title.
49          TextView name = (TextView)view.findViewById(R.id.textView);
50
51          //Manually determines what image will be displayed
52          //In order to add a new item make sure to add another if statement
```

```
53          if(orgOne.getEntityID() == 0){
54              image.setImageResource(R.drawable.it);
55          }
56          else if(orgOne.getEntityID()==1){
57              image.setImageResource(R.drawable.bio);
58          }
59          else if(orgOne.getEntityID()==2){
60              image.setImageResource(R.drawable.chem);
61          }
62          else if(orgOne.getEntityID()==3){
63              image.setImageResource(R.drawable.pot);
64          }
65          else if(orgOne.getEntityID()==4){
66              image.setImageResource(R.drawable.art);
67          }
68          else if(orgOne.getEntityID()==5){
69              image.setImageResource(R.drawable.sig);
70          }
71          else{
72              image.setImageResource(R.drawable.ic_launcher_background);
73          }
74
75          //Limits the numbers of character that can be displayed in an item's title.
76          if(orgOne.getOrgName().length() > 20){
77
78              String newName = orgOne.getOrgName().substring(0,18);
79              newName = newName + "...";
80              name.setText(newName);
81
82          }
83
84          else{
85              name.setText(orgOne.getOrgName());
86          }
```

```
        return view;
    }

}
```

```

/*
This class will work with DBHandler_v2 in order
to implement the database. DBHandler will create an
object from this class. This class will also be used by
the corresponding activity fo the feature.
Overall, a standard class with setters and getters for
each variable.
*/

public class Cafeteria {

    private int foodID;
    private String foodType;
    private String foodDescription;

    public Cafeteria()
    {

    }

    public Cafeteria(int foodID, String foodType, String foodDescription)
    {
        this.foodID = foodID;
        this.foodType = foodType;
        this.foodDescription = foodDescription;
    }

```

```
29      public int getFoodID() {
30          return foodID;
31      }
32
33      public void setFoodID(int foodID) {
34          this.foodID = foodID;
35      }
36
37      public String getFoodType() {
38          return foodType;
39      }
40
41      public void setFoodType(String foodType) {
42          this.foodType = foodType;
43      }
44
45      public String getFoodDescription() {
46          return foodDescription;
47      }
48
49      public void setFoodDescription(String foodDescription) {
50          this.foodDescription = foodDescription;
51      }
52  }
```

```
1
2   /*
3   This class will work with DBHandler_v2 in order
4   to implement the database. DBHandler will create an
5   object from this class. This class will also be used by
6   the corresponding activity fo the feature.
7   Overall, a standard class with setters and getters for
8   each variable.
```

```
*/

public class Directory {

    private int entityID;
    private String FName;
    private String LName;
    private String SEmail;
    private String SPhone;
    private String SPosition;
    private String SRoom;
    private String SPicture;

    public Directory()
    {

    }

    public Directory(int entityID,String FName,String LName,String SEmail,String SPhone,String
        SPosition,String SRoom,String SPicture)

    {
        this.entityID =entityID;
        this.FName = FName;
        this.LName = LName;
        this.SEmail = SEmail;
        this.SPhone = SPhone;
        this.SPosition = SPosition;
        this.SRoom = SRoom;
        this.SPicture = SPicture;

    }

    public int getEntityID() {
```

```
42          return entityID;
43      }
44
45      public void setEntityID(int entityID) {
46          this.entityID = entityID;
47      }
48
49      public String getFName() {
50          return FName;
51      }
52
53      public void setFName(String FName) {
54          this.FName = FName;
55      }
56
57      public String getLName() {
58          return LName;
59      }
60
61      public void setLName(String LName) {
62          this.LName = LName;
63      }
64
65      public String getSEmail() {
66          return SEmail;
67      }
68
69      public void setSEmail(String SEmail) {
70          this.SEmail = SEmail;
71      }
72
73      public String getSPhone() {
74          return SPhone;
75      }
```

```
76
77      public void setSPhone(String SPhone) {
78          this.SPhone = SPhone;
79      }
80
81      public String getSPosition() {
82          return SPosition;
83      }
84
85      public void setSPosition(String SPosition) {
86          this.SPosition = SPosition;
87      }
88
89      public String getSRoom() {
90          return SRoom;
91      }
92
93      public void setSRoom(String SRoom) {
94          this.SRoom = SRoom;
95      }
96
97      public String getSPicture() {
98          return SPicture;
99      }
100
101     public void setSPicture(String SPicture) {
102         this.SPicture = SPicture;
103     }
104
105 }
```

```
1
2  /*
```

```
3  This class will work with DBHandler_v2 in order
4  to implement the database. DBHandler will create an
5  object from this class. This class will also be used by
6  the corresponding activity fo the feature.
7  Overall, a standard class with setters and getters for
8  each variable.
9  */
10
11 public class Organizations {
12     private int entityID;
13     private String orgName;
14     private String presName;
15     private String presEmail;
16     private String presPhone;
17     private String orgDesc;
18     private String orgPic;
19
20     public Organizations()
21     {
22
23     }
24
25     public Organizations(int entityID,String orgName,String presName,String presEmail,String
           presPhone,String orgDesc,String orgPic)
26     {
27         this.entityID = entityID;
28         this.orgName = orgName;
29         this.presName = presName;
30         this.presEmail = presEmail;
31         this.presPhone = presPhone;
32         this.orgDesc = orgDesc;
33         this.orgPic = orgPic;
34     }
35
```

```
36      public void setEntityID(int entityID) {
37          this.entityID = entityID;
38      }
39
40      public int getEntityID() {
41          return entityID;
42      }
43
44      public void setOrgName(String orgName) {
45          this.orgName = orgName;
46      }
47
48      public String getOrgName() {
49          return orgName;
50      }
51
52      public String getPresName() {
53          return presName;
54      }
55
56      public void setPresName(String presName) {
57          this.presName = presName;
58      }
59
60      public String getPresEmail() {
61          return presEmail;
62      }
63
64      public void setPresEmail(String presEmail) {
65          this.presEmail = presEmail;
66      }
67
68      public String getPresPhone() {
69          return presPhone;
```

```
    }

    public void setPresPhone(String presPhone) {
        this.presPhone = presPhone;
    }

    public String getOrgDesc() {
        return orgDesc;
    }

    public void setOrgDesc(String orgDesc) {
        this.orgDesc = orgDesc;
    }

    public String getOrgPic() {
        return orgPic;
    }

    public void setOrgPic(String orgPic) {
        this.orgPic = orgPic;
    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <ImageView
```

```
10          android:id="@+id/icon"
11          android:layout_width="50dp"
12          android:layout_height="50dp"
13          android:src="@drawable/ic_launcher_background"
14          />
15
16      <TextView
17          android:id="@+id/textView"
18          android:layout_width="fill_parent"
19          android:layout_height="wrap_content"
20          android:layout_gravity="center"
21          android:textColor="@color/black"
22          android:textSize="24sp"
23          android:padding="16sp"
24          />
25
26  </LinearLayout>
27
28      <!--android:padding="@dimen/activity_horizontal_margin"-->
```

Bibliography

[1] Texas Tech System, "Texas tech university application."

[2] University of Texas at Austin, "University of texas at austin application."

[3] Straxis Technology, "University of utah application."

[4] University of Oklahoma, "Ou application."

[5] Texas State University, "Texas state mobile application."

[6] Louisiana State University, "Lsu mobile application."

[7] DCCCD, "Dcccd application."

[8] The University of Texas at Arlington, "Uta mobile application."

[9] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.

[10] B. B. Frank Tsui, Orland Karam, *Essential of Software Engineering*. Jones and Barlett Learning, 2006.

[11] T. Inc, "C++ programming language," 2018.

[12] H. D. Paul Deitel, *Java How to Program Early Objects*. Deitel, 2016.

[13] B. Hein, "Swift is already one of the world's most popular programming languages," 2017.

[14] M. Vinther, "Why you should totally switch to kotlin," 2017.

[15] R. Belov, "Kotlin 1.1 released with javascript support, coroutines and more," 2017.

[16] E. Britannica, "Sql computer language," 2018.

[17] C. B. Thomas Connolly, *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson, 2015.

[18] T. Network, "Microsoft sql server," 2017.

[19] O. Corporation, "Multimodel database with oracle database 12c release 2," 2017.

[20] T. A. S. Foundation, "Projects by name," 2017.

[21] T. A. S. Foundation, "Apache ignite is...," 2015.

[22] O. Corporation, "The main features of mysql," 2018.

[23] J. Mack, "Five advantages and disadvantages of mysql," 2014.

[24] D. R. Hipp, "What is sqlite?," 2019.

[25] uml diagrams.org, "Classification of uml 2.5 diagrams," 2009.

[26] CodingHorror, "Understanding model-view-controller."

[27] CodeAcademy, "Mvc: Model, view, controller app organization explained."

[28] C. Project, "Simple example of mvc (model view controller) design pattern for abstraction," 2008.