Capstone Project Report

|  |  |
| --- | --- |
| Nathaniel Hector  Department of Computer Science  HCTNAT003@myuct.ac.za | Saiuri Moodley  Department of Computer Science  MDLSAI008@myuct.ac.za |
| Zuleigha Patel  Department of Computer Science  PTLZUL001@myuct.ac.za | |

## Abstract

This report presents the results of the University of Cape Town (UCT) Food Places Review mobile application project. The project focuses on enhancing the existing UCTMobile mobile application with an additional feature that displays cafeteria information. This report presents a requirements analysis, design, and architecture overview and describes the user interface. Primary features provided by the app are highlighted and include displaying food vendor profiles and user reviews, capturing stock availability and the busyness level in real-time, and improving user engagement through a rewards system. This report concludes by addressing the importance of the chosen testing protocol in delivering a refined mobile application that is both functional and usable.

## Introduction

This project focuses on enhancing the existing UCTMobile mobile application with a feature that displays cafeteria information and the busyness level entered by food vendors. User reviews rating the cleanliness, stock availability, order turnaround time, and customer service of food vendors, and optional replies from the vendor are also displayed. Submitting a review allows users to earn points, which convert into vouchers for use at a food vendor of their choosing. Lastly, integrating Firebase Authentication and a quick response (QR) code scanner prevents exploitation of the rewards system.

The approach used to solve the problem combined elements from the Waterfall model and Scrum. During the requirements-capturing process, use case narratives and diagrams were produced. During design, package, class, and state machine diagrams were drawn. Implementation began with the evolutionary and horizontal prototype delivered during the design process. Use of the Scrum framework during the implementation stage allowed for flexibility in incorporating continuous and changing feedback from the client. Implementing the Model View Presenter (MVP) architectural pattern occurred during this stage.

## Requirements Captured

* Functional Requirements

**Sign Up**

Use Case Name: User Sign Up   
Actor: Student/Staff   
Description: The user wants to sign up to access restricted features   
Precondition: User is on the sign-up page   
Postcondition: User is registered and can access restricted features of the application   
Flow:

1.User clicks account button at top left corner and selects drop-down options to sign up as either student or staff and enters required details.

2. Application validates the details using certain criteria for student or staff [checks if email has @myuct.ac.za for student and @uct.ac.za for staff]

3. If the email and other credentials are invalid, the student or staff will be shown an error message stating that their sign-up was invalid

4. If their email credentials are valid, they will be signed up successfully

4. User is registered and gets confirmation through the application.

**Log In**

Use Case Name: User Log in   
Actor: Student/Staff   
Description: The user wants to log in to access restricted features   
Precondition: User is on the log in page and has a registered account   
Postcondition: User is logged in to the application and can access exclusive features   
Flow:

1. User clicks account button at top left corner, and selects drop-down options to login as either student or staff and enters required details

2. Application validates the details using certain criteria and compares login information against the email and other credentials stored in the cloud database

3. If the email and other credentials are invalid, the student or staff will be shown an error message stating that their log-in was invalid

4. If their email credentials are valid, they will get confirmation through the application. And be logged in successfully.

5. The user can now access restricted features only available to logged in student or staff members [such as the ability to leave a review or gain voucher points]   
 

**Log Out**

Use Case Name: User Log Out   
Actor: Student/Staff   
Description: The user wants to log out of the application   
Precondition: User is on their account page   
Postcondition: User is successfully logged out of the application and can no longer access exclusive features   
Flow:

1.User goes to their account page and selects the option ‘Log Out’ button

2. User is logged out and taken to the home page and the account page no longer displays their information

**Delete Account**

Use Case Name: User Deletes Account   
Actor: Student/Staff   
Description: The user wants to delete their account   
Precondition: User is on the log in page and has a registered account   
Postcondition: User is logged in to the application   
Flow:

1.User presses ‘Delete Account’ Button

2. Application sends pop-up to confirm account deletion

3. User confirms, and account is deleted. They no longer have access to exclusive features and their data is removed from the database.

**QR Code Scanning**

Use Case Name: QR Code Scanning for reviews   
Actor: Student/Staff   
Description: The user wants to scan the vendor’s QR code to make a review on them   
Precondition: The user is at the cashier desk of the vendor and has the application open   
Postcondition: User is taken to the ‘leave a review’ screen of the application   
Flow:

1.User opens the QR code scanner screen in the application

2. User scans the QR code available at the vendor

3. User is directed to the screen to make a review of the vendor

**Making a Review**

Use Case Name: Making a review   
Actor: Student/Staff   
Description: The student or staff wants to make a review on the vendor so that they can receive voucher points   
Precondition: The student or staff has bought items from the vendor and scanned their QR code and is on the ’make a review’ screen   
Postcondition: The student or staff’s review is posted on the vendor review page and the user receives more voucher points   
Flow:

1.Student or staff logs into the application using their email and other credentials

2. If the student or staff is not logged in and tries clicking the ‘Make a review’ button they will be prompted to log in

3. Once the student or staff is logged in, and in the ‘Make a review’ screen, they will be prompted to select star ratings for four categories and leave an additional optional comment as well

4. The review is posted on the review page once the ‘confirm’ button is pressed

5. The student or staff makes a review along with their ratings and clicks the submit button

6. Their review is now visible to other users

**Redeeming a Voucher**

Use Case Name: Redeeming a Voucher

Actor: Student or Staff

Description: The Student or Staff wants to redeem a voucher using their accumulated points.

Preconditions: The student or staff is logged into the application and has received more voucher points.

Postconditions: Depending on the number of voucher points, the option a voucher is generated, or a message is displayed telling the actor they do not have enough points to redeem a voucher.

Main Flow:

1. Actor selects the “profile” button.

2. On the profile screen, the actor presses the “redeem voucher” button and taken to their voucher screen which has an option for them to redeem a voucher [if applicable] and shows their voucher history along with its respective statuses and expiry dates.

3. Application checks the actor's accumulated points.

4. If the actor has enough points:

    1. System generates a unique voucher code, and voucher indicated as available for a specific vendor is added under their voucher history.

    2. Actor is presented with the clickable button ‘Redeem Voucher’ and a message which says they are eligible for a voucher

5. If the actor does not have enough points:

    1. The ‘Redeem Voucher’ button is not clickable, and a message is displayed on the voucher screen indicating they are not eligible for a voucher

    2. Actor is shown an error message stating they need x number of additional points to redeem the voucher.

**Selecting a Vendor**

Use Case Name: Selecting a vendor   
Actor: Student/Staff   
Description: User wants to view the vendor menu and information   
Precondition: User is on the vendor listing screen   
Postcondition: User has selected a vendor and can see their information   
Flow:

1.User scrolls through the list of vendors

2. User chooses a vendor

3. Application shows vendor information screen

**Vendor Replying to Reviews**

Use Case Name: Vendor replying to comments on reviews   
Actor: Vendor   
Description: A specific vendor wants to respond to a review made by one of the users   
Precondition: Vendor is logged into the application and on their review page   
Postcondition: Vendor’s reply can be seen on the review page   
Flow:

1.Vendor selects the ‘comment on reviews’ button in their account page

2. Vendor clicks reply button under a review

3. Vendor types in their reply to the comment and clicks the ‘submit’ button

4. The vendor reply can be seen on the review page below the review

**Vendor Editing their Menu**

Use Case Name: Vendor editing their menu   
Actor: Vendor   
Description: Vendor wants to update their menu items   
Precondition: Vendor is logged into the application and has clicked the ‘edit menu’ button   
Postcondition: The menu items are updated   
Flow:

1.Vendor clicks on the ‘edit menu’ option and is on the menu page

2. Vendor adds or removes menu items or toggles their stock availability

3. The menu can be seen as updated when it is refreshed.

**Voucher Marking**

Use Case Name: Voucher Marking   
Actor: Vendor   
Description: The vendor wants to flag certain vouchers as used   
Precondition: A user has redeemed a voucher at the vendor   
Postcondition: The voucher is flagged as used   
Flow:

1.User attempts to redeem voucher at vendor

2. The Vendor then logs in and goes to their ‘Redeem Vouchers’ page

3. The vendor is prompted to enter in a voucher code and types in the voucher code of the user

4. The vendor marks that voucher as redeemed if it is a valid code

5. The voucher status is updated to redeemed/used on the application

**Vendor Updating Menu**

Use Case Name: Vendor updating menu   
Actor: Vendor   
Description: The vendor wants to see their menu updates in real-time   
Precondition: Vendor has updated their menu   
Postcondition: The menu changes can be seen by the users   
Flow:

1.Vendor adds/removes items on the menu

2. The menu changes are updated for all users

3. Users can see the change to the menu.

* Non-functional Requirements

**Data Security**

Use Case Name: Data Security   
Actor: User   
Description: The user wants their login information to be stored in a secure database so that their privacy is protected.

Precondition: User has created an account with login details    
Postcondition: User login details and password are stored in a secure database

Flow:

1. User gives login information

2. The login information is stored securely by the application. Their login information is stored securely in Firebase – a cloud database which ensures data encryption and security.

**Vendor Restricted Access**

Use Case Name: Vendor Restricted Access   
Actor: Vendor   
Description: The vendor wants access to certain parts of the application inaccessible by the user for them to update their menu items and vendor information.

Precondition: Vendor has logged into the application using their vendor login information provided by the admin   
Postcondition: Vendor has access to exclusive parts of the application, and unauthorized changes to these parts are prohibited from users [such as updating menu items, busyness, vendor information and replying to comments]. The vendor, however, does not have access to certain functions in the application available to users [such as making a review or scanning QR code to redeem their vouchers]

Flow:

1. Vendor attempts to access one of these exclusive features of the application

2. The application allows the vendor access to specific parts of the application corresponding to their function

**Database Handling Large Volumes of Data**

Use Case Name: Database handling large volumes of data   
Actor: Vendor   
Description: The vendor wants the cloud database to manage large volumes of data, to ensure that the application is working with low latency.

Precondition: Large volumes of data updates are made [data can include menu or reviews or new user login credentials]   
Postcondition: The application still works as intended and remains consistent

Flow:

1. Multiple users make reviews on the application, or the vendor updates their menu [adds or removes items, updates stock availability of items] or unregistered users sign up for the application

2. The application can process and store these large volumes of data without any adverse effects on application performance.

**Program Documentation**

Use Case Name: Program Documentation   
Actor: Developer   
Description: The developer wants the programs for the application to have good documentation and refactoring of code so that updates and bug fixes can be made easier.

Precondition: Developer has the application code open   
Postcondition: The developer finds the code easy to understand and modify

Flow:

1. The developer adds documentation to the code while reviewing it

2. The developer has a good understanding of the code due to the documentation

**Mitigation for Security**

Use Case Name: Mitigation for security   
Actor: User   
Description: The user wants the application to have measures against security risks to ensure their data is secure.

Precondition: User is using the application   
Postcondition: User’s data is protected against security risks. It is protected by employing the Firebase cloud database which has built-in data authentication, and the data is protected from unauthorized users.

Flow:

1. User is interacting with the application

2. The application has measures against any potential data security risks

**Application Handling Large Volume of Users.**

Use Case Name: Application handling large volume of users   
Actor: User   
Description: The user wants the application to be able to manage a large volume of users effectively.

Precondition: Multiple users are interacting with the application   
Postcondition: The application still works as intended and sustains responsiveness

Flow:

1. Multiple users are inside the application and interacting with its features

2. The application manages its resources effectively to manage increasing numbers of users

**Backup of Application Data**

Use Case Name: Backup of application data   
Actor: Vendor   
Description: The vendor wants the application data to backed up [such as login credentials, reviews, voucher information, vendor menu information] and stored in a cloud database to reduce the risk of data loss.

Precondition: Data is being updated and stored on the application by the vendor [such as the vendor adding or deleting menu items, updating stock availability, adding replies to their reviews, redeeming vouchers for their users  (the application would need to update the status of these vouchers as redeemed)]   
Postcondition: The data is backed up in a database and kept safe.

Flow:

1. The vendor updates information on the application

2. The updated information is stored in a backup database [cloud Firebase database]

3. A backup of the most recent data is stored in the cloud database

* Usability Requirements

**Intuitive Application Design**

Use Case Name: Intuitive Application Design   
Actor: User   
Description: The user wants the application to be user-friendly and aesthetically pleasing.

Precondition: User accesses the application   
Postcondition: User interacts with the application and finds the design satisfactory

Flow:

1. User goes into the application

2. User experiences a visually appealing interface

3. User navigates through the various application screens with ease

**Intuitive Voucher Reward System**

Use Case Name: Intuitive voucher reward system   
Actor: User   
Description: The user wants the voucher reward system to be straightforward and easy to understand.

Precondition: User is interacting with the voucher screen   
Postcondition: User can comprehend how the voucher reward system works with ease and understands how to redeem their vouchers [1 point for reviews with comments of <70 words, 2 points for >70 points, vouchers are vendor-specific]

Flow:

1. User is inside the voucher screen

2. User sees all their relevant voucher history, voucher points and a place for them to redeem available vouchers on the voucher screen

3. User feels comfortable using the voucher reward system

**Accessible User Manual**

Use Case Name: Accessible User Manual   
Actor: User   
Description: The user wants to understand how to use the application’s key features.

Precondition: User is interacting with the user manual   
Postcondition: User now understands how to use the key features of application

Flow:

1. User is inside the user manual

2. User reads the user manual and understands how to use the key features of the application

**Accurate Vendor Information**

Use Case Name: Accurate Vendor Information   
Actor: User   
Description: The user wants to ensure that the vendor information and menus are up-to-date and accurate.

Precondition: User is browsing through the vendor screens   
Postcondition: User makes informed food choices based on the accurate vendor information.

Flow:

1. User selects a specific vendor from the vendor selections page

2. User views the specific vendor’s dietary information and menu in their vendor home page

3. User trusts the vendor information and menu being displayed

**Intuitive Menu Updating**

Use Case Name: Intuitive Menu Updating   
Actor: Vendor   
Description: The vendor wants an intuitive and easy-to-understand navigation for updating their menu items.

Precondition: The vendor is logged into the application and accessing the menu editing screen   
Postcondition: The vendor can update their menu items with ease

Flow:

1. Vendor clicks the ‘Edit Menu’ button inside their profile screen

2. Vendor finds intuitive buttons for them to add, remove or modify the stock availability of menu items [these are given by icons at the bottom of the screen making it easier for the vendor to understand what they are for]

3. Vendor successfully updates their menu items without any hassles

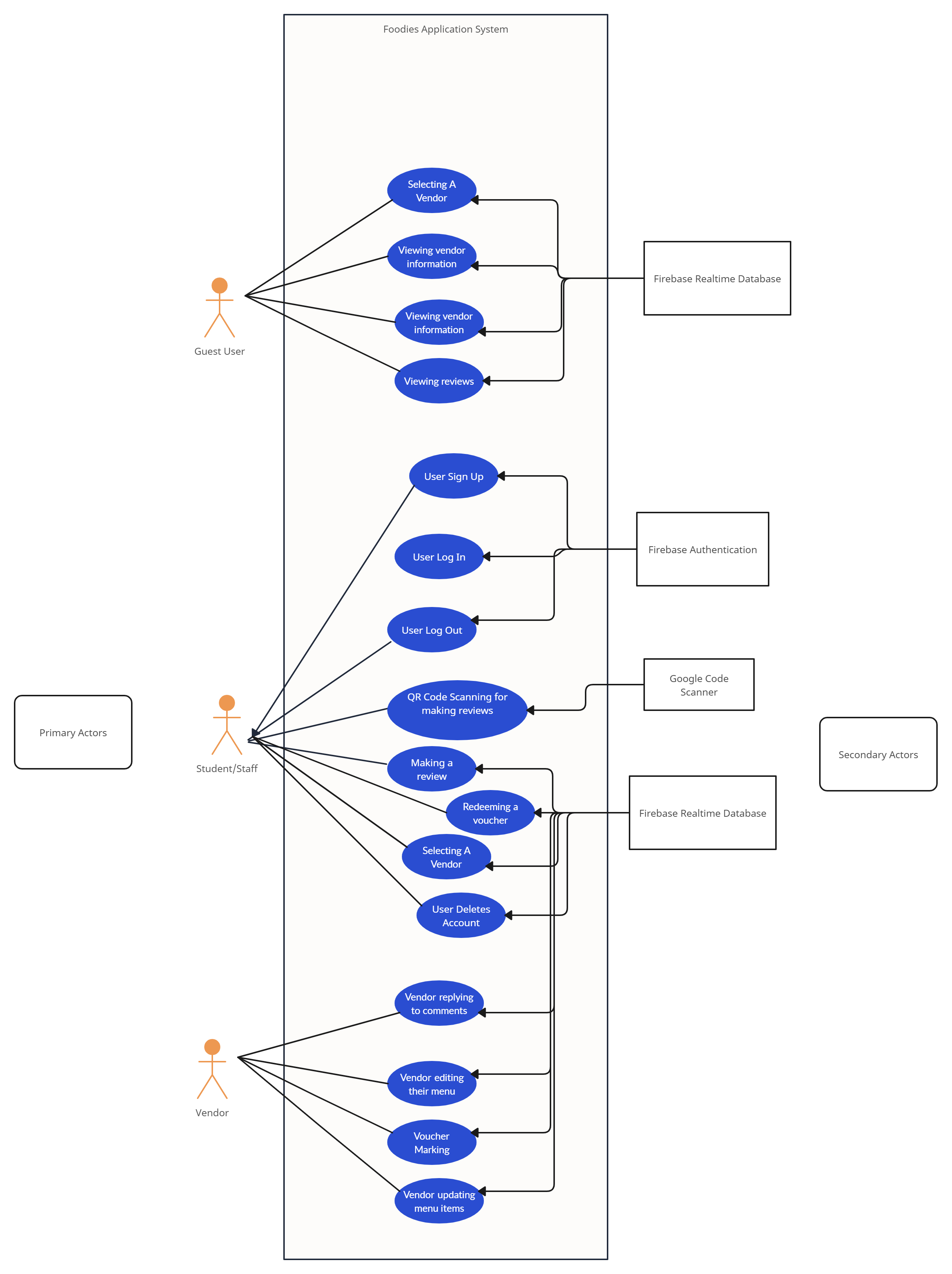
****

Figure 1: Use Case Diagram

## Design OverviewA diagram of a company Description automatically generated

Figure 2: Visual Representation of System Architecture: Package Diagram

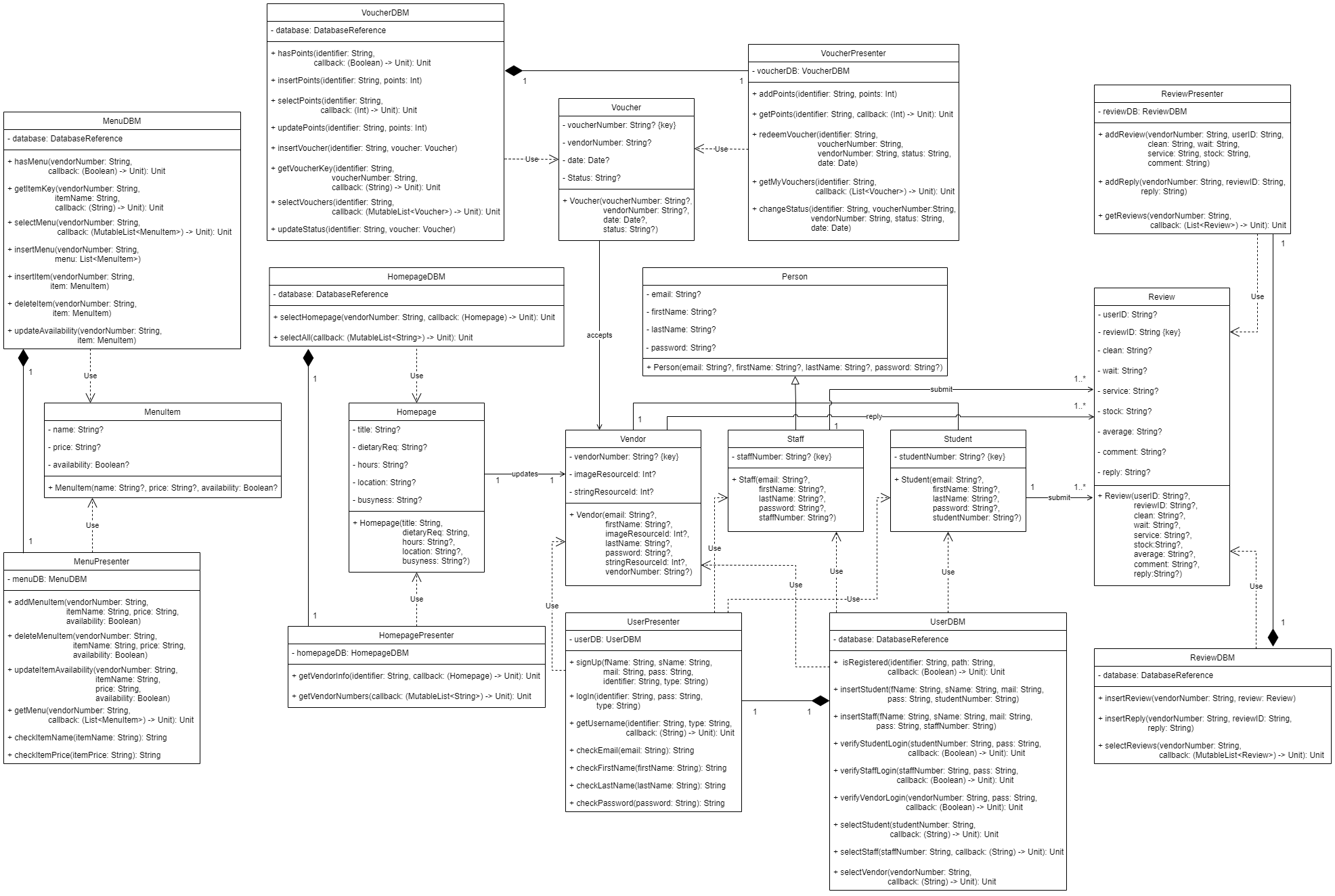


Figure 3: Visual Representation of System Architecture: Design Class Diagram

* Data Model

The Firebase Realtime Database is a NoSQL database that synchronizes data across all clients in real-time (Firebase, 2023). Vendors capture live updates such as the availability of preprepared menu items and their busyness. Using this database for this application was essential because students and staff members expect to receive these updates in real-time.

* Data Storage and Retrieval Algorithms Used

The Firebase Realtime Database is a cloud-hosted NoSQL database that stores data as JavaScript Object Notation (JSON) (Firebase, 2023).

Data is written or overwritten using setValue(), which automatically maps and nests an object’s attributes to child locations (Firebase, 2023). Using Kotlin objects for this application was essential in ensuring code readability and maintainability.

Data is appended using push(), which writes an object to an existing reference and generates a unique key (Firebase, 2023). This method prevents write conflicts, which can arise when several clients attempt to write to one location simultaneously.

Data is read using a Firebase Realtime Database Query class instance and a SingleValueEventListener (Firebase, 2023). The Query class includes functionality to sort data by key, value, or child value and filter sorted data by a range of keys or values or limit the number of items returned in the results list.

## Implementation

* Database Design

The homepages path contains publicly available vendor information stored as Homepage objects. A vendor number uniquely identifies each Homepage object. Homepage data includes busyness, dietary requirements, operating hours, location, and business name. This organization allows retrieving and updating homepage data to be quick and efficient (Firebase, 2023).

The menus path contains vendor menus stored as a list of MenuItem objects. A vendor number uniquely identifies each list, and an auto-generated key uniquely identifies each MenuItem object. MenuItem data includes the product's name, price, and availability. This organization allows retrieving and displaying menu data to be simple and up-to-date (Firebase, 2023). There is also an additional benefit for vendors who can easily view and manage their menus.

The points path contains user points rewards stored as a String. A student or staff number uniquely identifies each String. This organization allows adding, subtracting, and retrieving points data to be quick and efficient (Firebase, 2023).

The reviews path contains user reviews of vendors stored as a list of Review objects. A vendor number uniquely identifies each list, and an auto-generated key uniquely identifies each Review object. Review data includes the date, ratings, a user comment, an optional vendor reply, and a user and review identifier. This organization allows accessing review data for display purposes to be simple and provides a review history (Firebase, 2023). There is also an additional benefit for vendors who can easily view and reply to user reviews.

The users path contains user profiles stored as Student, Staff, or Vendor objects. A student, staff, or vendor number uniquely identifies each profile. User data includes first name, last name, email address, and a user identifier. This organization allows inserting users into the database and retrieving user data from the database to be quick and efficient (Firebase, 2023).

The vouchers path contains user voucher rewards stored as a list of Voucher objects. A student or staff number uniquely identifies each list, and an auto-generated key uniquely identifies each Voucher object. Voucher data includes the date, a status, voucher number and the vendor for which the voucher is redeemable. This organization allows accessing voucher data for display purposes to be simple and provides a voucher history (Firebase, 2023).

* Significant Methods

Table 1: Significant Methods and Their Functionality

|  |  |
| --- | --- |
| Methods | Functionality |
| 1. selectHomepage() | Retrieve a homepage from the database for a given vendor number. |
| 1. updateBusyness() | Update a vendor's busyness in the database for a given vendor number. |
| 1. selectMenu() | Retrieve a menu from the database for a given vendor number. |
| 1. insertItem() | Append a menu item to an existing menu in the database for a given vendor number. |
| 1. deleteItem | Remove a menu item from an existing menu in the database for a given vendor number and product name. |
| 1. updateAvailability() | Update the stock availability of a menu item in the database for a given vendor number and product name. |
| 1. insertReview() | Append a user review of a vendor to an existing list of reviews in the database for a given vendor number. |
| 1. insertReply() | Update a user's review with a vendor's reply in the database for a given vendor number and review identifier. |
| 1. selectReviews() | Retrieve a list of a vendor's reviews from the database for a given vendor number. |
| 1. addNewUser() | Create a user account with an identifier using Firebase Authentication. |
| 1. deleteAccount() | Remove a user account from the database for a given user identifier. |
| 1. insertStudent() | Insert a student profile into the database with a given student number. |
| 1. insertStaff() | Insert a staff member profile into the database with a given staff number. |
| 1. verifyLogin() | Verify a login credentials using Firebase Authentication. |
| 1. selectStudent() | Retrieve a student profile from the database for a given student number. |
| 1. selectStaff() | Retrieve a staff member profile from the database for a given staff number. |
| 1. selectVendor() | Retrieve a private vendor profile from the database for a given vendor number. |
| 1. Constructors | No-argument default constructors are required for data deserialization from a DataSnapshot. |

A diagram of a computer

Description automatically generated

Figure 4: State Machine Diagram

* Libraries Used

The Firebase Realtime Database is a cloud-hosted NoSQL database that stores data as JavaScript Object Notation (JSON) (Firebase, 2023). All clients share a single Realtime Database instance and data synchronization is ensured when clients automatically load changes in the data.

The Google Code Scanner is a local, offline, and permissionless barcode scanning Application Programming Interface (API) (ML Kit, 2023). It decodes structured data such as text when a QR code is detected.

## Program Validation and Verification

The application was evaluated by going through various test cases that correspond with the main functions of the application. The testing techniques used were white box testing (where the internal structure of the part being evaluated is known to the tester). The goal of testing is to identify all potential problems before making the application accessible to users. We chose this testing plan because it effectively stress tests the backend and frontend. Our understanding of the application allowed us to ensure that we address all the required functionalities to ensure the application runs smoothly. The test plan worked 100% successfully.

Table 2: Summary Testing Plan.

|  |  |
| --- | --- |
| Process | Technique |
| 1. Class Testing: test methods and state behaviour of classes | **White-Box Tests:** Testing code to ensure all parts of the application such as making sure that all requirements for review rating (1-5 stars) are tested. |
| 1. Integration Testing: evaluate the interaction of sets of classes | **Behavioural Testing:** Test how the application behaves when the new features are integrated, such as ensuring that when users make a review, it is submitted successfully and correctly updates the vendor’s overall average rating |
| 1. Validation Testing: test whether customer requirements are satisfied | **Use-case based black box:**  This involves testing the application based on user stories. For example, a scenario where the user logs in as a student or staff member, selects a vendor, and views their vendor information, and leaves a review.  **Acceptance tests:**  Acceptance tests are used to ensure user is satisfied with the application. This involves testing the use cases and trying out the main features of the application and checking if we are satisfied with the application. We also tested it with the real users, and they tried out the main features and gave relevant feedback. |
| 1. System Testing: evaluate the behaviour of the system as part of a larger environment | **Recovery Testing:** We tested to ensure that the application can detect errors in login and sign-up credentials by testing if it can detect if the wrong information is entered when the user logs in or signs up.  **Security testing:**  Ensuring that prohibited access to features specific to users or vendors are abided by. For example, only vendors have access to exclusive features that are unauthorised from users [such as the editing menu screen]. Another example would be ensuring that vendors cannot change the reviews made by users, and users cannot reply to comments made on the review page.  **Stress testing:**  We tested the application with large volumes of data such as for the menu items for each of the vendors being stored in the cloud database and ensuring that the application still works as intended with this large volume of data. |

Table 3: Summary of Test Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Descriptions | Test Cases | | |
| Normal Data Set | Boundary Data Set | Invalid Data Set |
| Tests as a Student: | | | |
| Sign Up | Passed | N/A | Passed |
| Log In | Passed | N/A | Passed |
| View Vendor Information | Passed | N/A | N/A |
| View Updated Vendor Information | Passed | N/A | N/A |
| Verify a Review | Passed | N/A | Passed |
| Submit a Review | Passed | N/A | Passed |
| Redeem a Voucher | Passed | Passed | Passed |
| Log Out | Passed | N/A | N/A |
| Delete Account | Passed | N/A | N/A |
| Tests as a Staff Member: | | | |
| Sign Up | Passed | N/A | Passed |
| Log In | Passed | N/A | Passed |
| View Vendor Information | Passed | N/A | N/A |
| View Updated Vendor Information | Passed | N/A | N/A |
| Verify a Review | Passed | N/A | Passed |
| Submit a Review | Passed | N/A | Passed |
| Redeem a Voucher | Passed | Passed | Passed |
| Log Out | Passed | N/A | N/A |
| Delete Account | Passed | N/A | N/A |
| Tests as a Vendor: | | | |
| Sign Up | N/A | N/A | N/A |
| Log In | Passed | N/A | Passed |
| Add Menu Item | Passed | N/A | Passed |
| Update Stock Availability | Passed | N/A | N/A |
| Remove Menu Item | Passed | N/A | Passed |
| Update Busyness | Passed | N/A | N/A |
| Reply to a Review | Passed | N/A | Passed |
| Edit a Reply | Passed | N/A | Passed |
| Verify a Voucher | Passed | N/A | Passed |
| Log Out | Passed | N/A | N/A |
| Tests as a Guest: | | | |
| View Vendor Information | Passed | N/A | N/A |
| View Updated Vendor Information | Passed | N/A | N/A |
| Verify a Review | Passed | N/A | Passed |

## Conclusion

We met 100% of the functionality for the application.

We have included the required main functions such as the user being able to view vendor information, redeem and view their vouchers, leave reviews, sign-up and log in and scan QR codes. The functions available to the vendor are the ability to edit their menu items, vendor information, busyness, and stock availability. Vendors can also sign in, redeem vouchers for their users and respond to reviews made by users for them.

An additional feature we’ve included is firebase authentication. There were various additional features we were keen on adding but unable to due to time constraints such as being able to search for vendors or change your username as a user. We therefore have all the functional and non-functional requirements set out in this document present in the application.

We have ensured that the UI design meets usability requirements and the principles of material design by adding intuitive navigation and processes for vendors to edit their menu items or users to leave a review for a vendor. We ensured data security and error checking by having error messages (through toast messages in the application) displayed such as when the incorrect credentials are entered for logging in or signing up or entering an incorrect voucher code. However, overall, our necessary aims were sufficiently met, and all our test cases gave successful outcomes.

# References

* Firebase, 2023. *Firebase Realtime Database.* [Online]   
  Available at: https://firebase.google.com/docs/database  
  [Accessed 14 August 2023].
* Firebase, 2023. *Read and Write Data on Android.* [Online]   
  Available at: https://firebase.google.com/docs/database/android/read-and-write  
  [Accessed 14 August 2023].
* Firebase, 2023. *Structure Your Database.* [Online]   
  Available at: https://firebase.google.com/docs/database/android/structure-data  
  [Accessed 24 August 2023].
* Firebase, 2023. *Work with Lists of Data on Android.* [Online]   
  Available at: https://firebase.google.com/docs/database/android/lists-of-data  
  [Accessed 14 August 2023].
* ML Kit, 2023. *Google code scanner (Android only).* [Online]   
  Available at: https://developers.google.com/ml-kit/vision/barcode-scanning/code-scanner  
  [Accessed 24 August 2023].

# Appendix A — Database Schema

{

  "homepages": [

    {

      "1": {

        "busyness": "Low",

        "dietaryReq": "Vegetarian",

        "hours": "08:30 - 16:30",

        "location": "Cissy Gool Plaza",

        "title": "AfriQuezeen"

      }

    }

  ],

  "menus": [

    {

      "1": [

        {

          "availability": true,

          "name": "Full Chicken Curry",

          "price": "80"

        }

      ]

    }

  ],

  "points": {

    "HCTNAT003": 10

  },

  "reviews": [

    {

      "1":[

        {

          "-NegGoKcvdpLU6vouwJO": {

            "average": "2.5",

            "clean": "1",

            "comment": "Fantastic food and friendly staff.",

            "date": "18/09/2023",

            "reply": "Thank you for your support!",

            "reviewID": "-NegGoKcvdpLU6vouwJO",

            "service": "3",

            "stock": "4",

            "userID": "HCTNAT003",

            "wait": "2"

          }

        }

      ]

    }

  ],

  "users": {

    "HCTNAT003": {

      "email": "HCTNAT003@myuct.ac.za",

      "firstName": "Nathaniel",

      "lastName": "Hector",

      "studentNumber": "HCTNAT003"

    },

    "STAFFNUM1": {

      "email": "STAFFNUM1@myuct.ac.za",

      "firstName": "Jane",

      "lastName": "Doe",

      "staffNumber": "STAFFNUM1"

    },

    "1": {

      "email": "AfriQuezeen@gmail.com",

      "firstName": "Alex",

      "imageResourceId": "",

      "lastName": "Mitchell",

      "stringResourceId": "",

      "vendorNumber": "1"

    }

  },

  "vouchers": {

    "HCTNAT003": {

      "-NeEGtSwKs4ak5nRHuEN": {

        "date": {

          "date": 13,

          "day": 3,

          "hours": 18,

          "minutes": 2,

          "month": 8,

          "seconds": 28,

          "time": 1694620948348,

          "timezoneOffset": -120,

          "year": 123

        },

        "status": "Redeemed",

        "vendorNumber": "1",

        "voucherNumber": "-NeEGtSwKs4ak5nRHuEN"

      }

    }

  }

}

# Appendix B — Comprehensive Test Results

Table 4: Comprehensive Test Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Descriptions | Test Condition | Test Data | Expected Results: System Behaviour or State | Status |
| Tests as a Student: | | | | |
| Sign Up | Sign up approved. | ABCDEF123, ABCDEF123@myuct.ac.za, Anna, Smith, password | Confirmation message and profile screen displayed. | Passed |
| Sign Up | Sign up refused. | ABCDEF12, ABCDEF13@myuct.ac.za, A, S, 123 | Incorrect inputs highlighted and error messages displayed. | Passed |
| Log In | Log in approved. | ABCDEF123, password | Confirmation message and profile screen displayed. | Passed |
| Log In | Log in refused. | ABCDEFGHI, password | Error message displayed. | Passed |
| View Vendor Information | Information from database retrieved. | N/A | Vendor listing and homepage screens display information. | Passed |
| View Updated Vendor Information | Updated information from database retrieved. | N/A | Vendor listing and homepage screens display updated information. | Passed |
| Verify a Review | Review approved. | Correct QR code for vendor. | Review form screen displayed. | Passed |
| Verify a Review | Review refused. | Incorrect QR code for vendor. | Return to review listing screen. | Passed |
| Verify a Review | Review refused. | Incorrect QR code and vendor. | Return to review listing screen. | Passed |
| Submit a Review | Review submitted. | 5, 4, 3, 2, Great! | Review appears on review listing screen. | Passed |
| Submit a Review | Review submitted. | N/A | Review appears on review listing screen. | Passed |
| Redeem a Voucher | Voucher issued. | N/A | Profile screen updated with voucher and new points total. | Passed |
| Redeem a Voucher | Voucher not issued. | N/A | Insufficient points error message displayed. | Passed |
| Log Out | QR code access refused. | N/A | Guest profile activated. | Passed |
| Delete Account | Log in refused. | N/A | Guest profile activated. | Passed |
| Tests as a Staff Member: | | | | |
| Sign Up | Sign up approved. | ABCDEF123, ABCDEF123@myuct.ac.za, Anna, Smith, password | Confirmation message and profile screen displayed. | Passed |
| Sign Up | Sign up refused. | ABCDEF12, ABCDEF13@myuct.ac.za, A, S, 123 | Incorrect inputs highlighted and error messages displayed. | Passed |
| Log In | Log in approved. | ABCDEF123, password | Confirmation message and profile screen displayed. | Passed |
| Log In | Log in refused. | ABCDEFGHI, password | Error message displayed. | Passed |
| View Vendor Information | Information from database retrieved. | N/A | Vendor listing and homepage screens display information. | Passed |
| View Updated Vendor Information | Updated information from database retrieved. | N/A | Vendor listing and homepage screens display updated information. | Passed |
| Verify a Review | Review approved. | Correct QR code for vendor. | Review form screen displayed. | Passed |
| Verify a Review | Review refused. | Incorrect QR code for vendor. | Return to review listing screen. | Passed |
| Verify a Review | Review refused. | Incorrect QR code and vendor. | Return to review listing screen. | Passed |
| Submit a Review | Review submitted. | 5, 4, 3, 2, Great! | Review appears on review listing screen. | Passed |
| Submit a Review | Review submitted. | N/A | Review appears on review listing screen. | Passed |
| Redeem a Voucher | Voucher issued. | N/A | Profile screen updated with voucher and new points total. | Passed |
| Redeem a Voucher | Voucher not issued. | N/A | Insufficient points error message displayed. | Passed |
| Log Out | QR code access refused. | N/A | Guest profile activated. | Passed |
| Delete Account | Log in refused. | N/A | Guest profile activated. | Passed |
| Tests as a Vendor: | | | | |
| Sign Up | Sign up refused. | N/A | Contact administrator error messages displayed. | Passed |
| Log In | Log in approved. | 1, password | Confirmation message and profile screen displayed. | Passed |
| Log In | Log in refused. | 1989, password | Error message displayed. | Passed |
| Add Menu Item | Item added. | Pizza, 25 | Edit menu screen updated with new item. | Passed |
| Add Menu Item | Item ignored. | N/A | Error message displayed. | Passed |
| Update Stock Availability | Stock availability updated. | N/A | Edit menu screen updated with changes. | Passed |
| Remove Menu Item | Item removed. | Check checkboxes of items. | Edit menu screen updated with changes. | Passed |
| Remove Menu Item | Operation ignored. | No item checkboxes checked. | Original edit menu screen displayed. | Passed |
| Update Busyness | Busyness updated. | Low | User profile screen updated with new busyness. | Passed |
| Reply to a Review | Reply accepted. | Thank you | User profile screen updated with new reply. | Passed |
| Reply to a Review | Reply rejected. | N/A | Error message displayed. | Passed |
| Edit a Reply | Edited reply accepted. | Thank you very much | User profile screen updated with updated reply. | Passed |
| Edit a Reply | Edited reply rejected. | N/A | Error message displayed. | Passed |
| Verify a Voucher | Voucher approved. | ABDEF123-955-721-30 | Confirmation message displayed. | Passed |
| Verify a Voucher | Voucher refused. | 1 | Error message displayed. | Passed |
| Log Out | QR code access refused. | N/A | Guest profile activated. | Passed |
| Tests as a Guest: | | | | |
| View Vendor Information | Information from database retrieved. | N/A | Vendor listing and homepage screens display information. | Passed |
| View Updated Vendor Information | Updated information from database retrieved. | N/A | Vendor listing and homepage screens display updated information. | Passed |
| Verify a Review | QR code access refused. | N/A | Log in error message displayed. | Passed |