CNG 495

Fall – 2022

Term Project Final

Report

PopTracker

by

Ibrahim Ozkan – 2456275

(Adil Bozkurt Kebapcioglu – 2455954)

Table of Contents

1. Introduction

The PopTracker application enables businesses to share their live indoor population information with their customers. The system involves placing Arduino devices equipped with sensors in the entrances and exits of indoor areas. These devices will detect if a person enters or exits the area and sends this information to a backend running in a server hosted by amazon web services. The backend processes the data and stores it in an Amazon RD2 database. The clients will have a mobile application where they can view the population data of specific businesses in real time which is provided by the backend. The system will also have an admin that has extra privileges such as business management. The backend will store the population data of each business on regular intervals. The admin can access this extra information regarding population history for each business.

Through the application, businesses will have another service they can provide for their customers and have access to important information regarding the popularity of their business on certain times of day or on certain days of the week. The customers will have the ability to see if a certain business is crowded and may choose to go at another time.

Usually, a population tracker like PopTracker is implemented by businesses themselves which is specific for their own use and for their own customers. For an example, this can be seen on high-end gyms. However, in reality, most businesses do not own such systems due the costs involved in implementation. Our application provides a general solution that can meet the needs of any business in a cost-effective manner.

GitHub Repo of PopTracker: https://github.com/ibrahimozkn/CNG495-F22-CloudComputing.git

2. Project Structure

2.1. Cloud Services Utilized

The project consists mainly of 2 components: the frontend and the backend. The frontend communicates with the backend through HTTP requests and does not utilize any other cloud services. The backend utilizes the following cloud services:

- AWS EC2 (Amazon web services Elastic Compute Cloud): The backend runs on a virtual Linux machine provided by amazon web services. The service also provides a public IP address making the backend services accessible through the internet.
- AWS RD2 (Amazon web services Relational Database Service): Instead of using a local database, the backend uses RD2 service. The service provides extra features such as security and scalability.

2.2. Technologies Used

Backend: PHP framework Laravel, MySQL,

Frontend: Flutter framework, Location services

2.3. Application Functionalities

2.3.1. Account Creation

Firstly, all users who want to use the application needs have an account. Therefore, users must register using the following registration page:

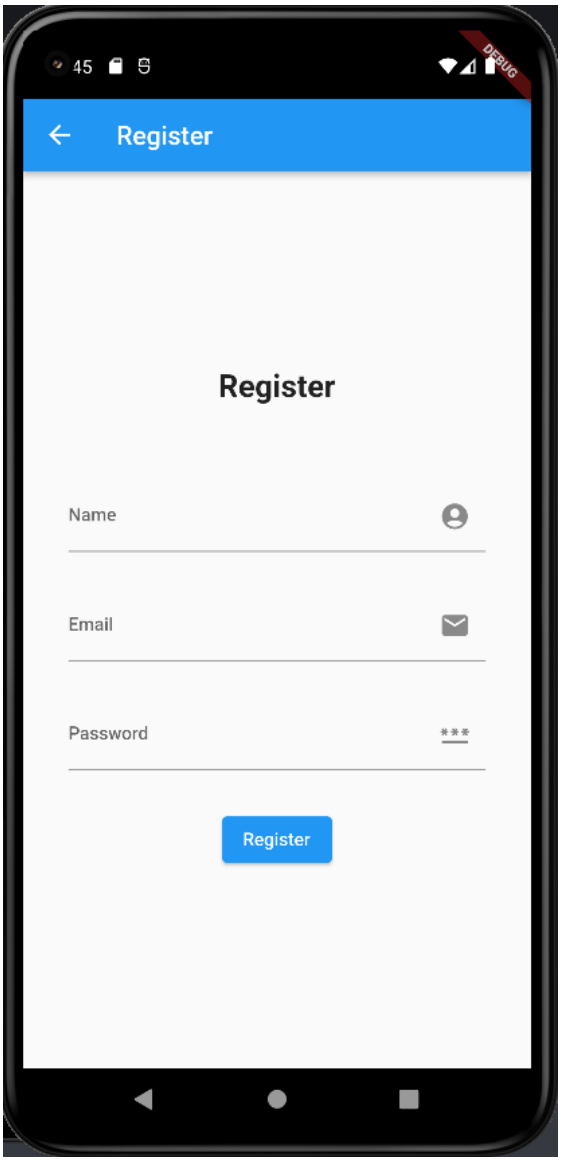

Figure 1 Registration Page

After user fills the form and clicks on the “Register” button, the application sends a HTTP POST request with the provided details to submit account registration request to the backend server. The server validates the given details and creates a new user with an encrypted password. After everything goes successful, application navigates user back to the login screen.

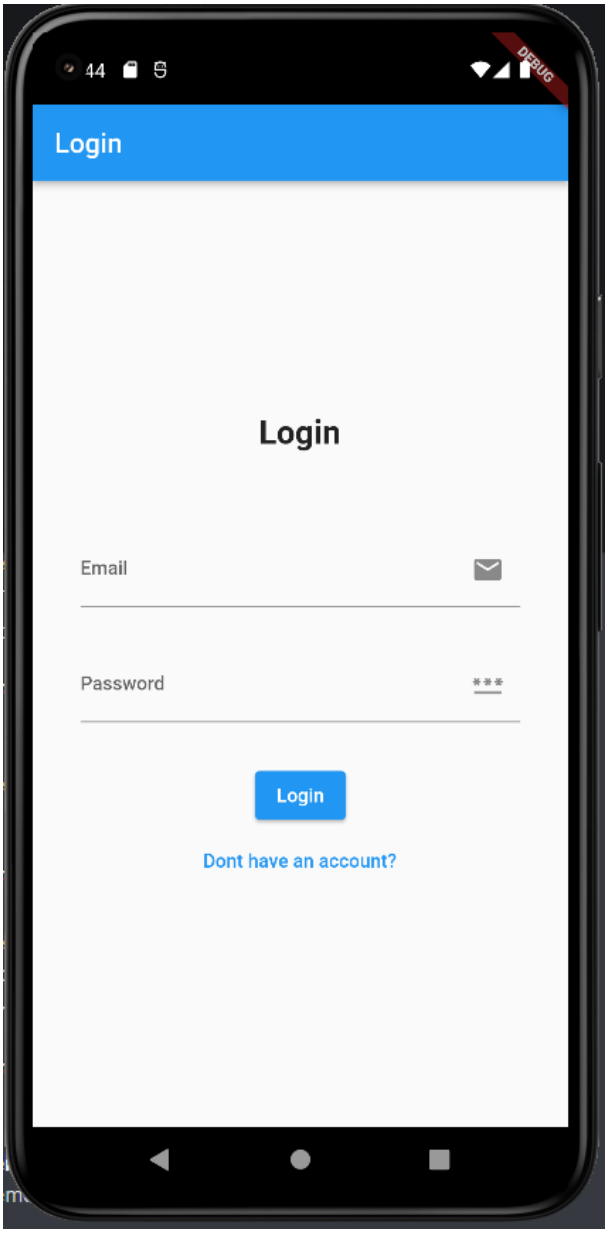

Figure 2 Login Page

Here, user must provide its credentials and press the login button. When the button is pressed, application sends a HTTP POST request to the backend server with the provided credentials to login. The server validates the given credentials then if credentials are matching with an existing account, it creates a session token which is returned together with the role indicator (admin or user).

2.3.2. Role Based User Authentication

Both users and admins share the same UI with different additional functionalities. When the role response is returned with the login operation, application decides whether to hide or show the admin operation in the UI based on the user role. For example, in figure 3, deleting business and add business buttons (plus button at the top) are only visible for user with an admin role.

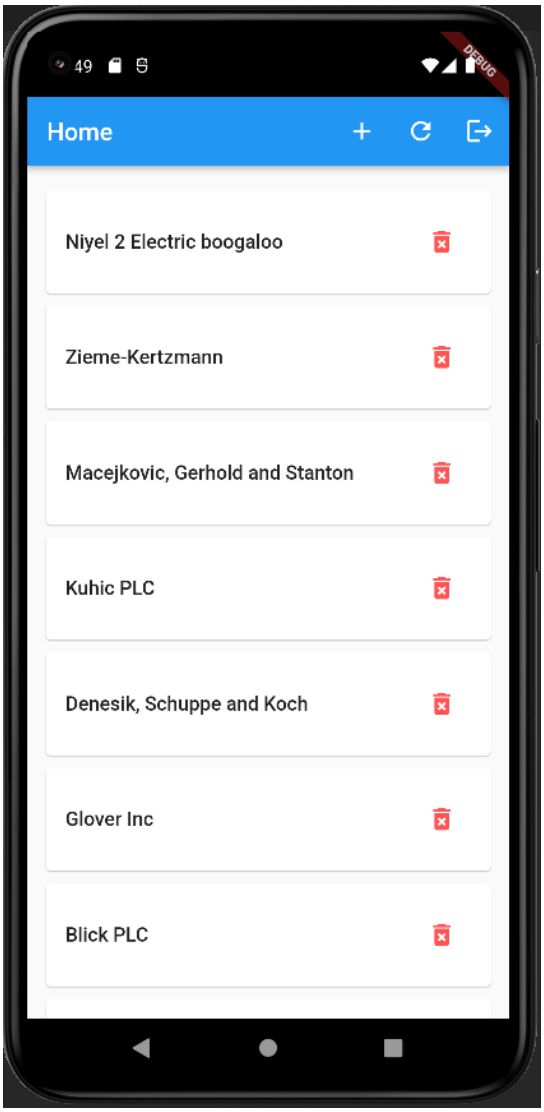

Figure 3 Business Listing Screen

2.3.3. Business Listing

After user logs in, application sends a HTTP GET request to the backend server with session token. Then, backend server returns the list of businesses to the application which are listed which can be seen in the figure 3.

2.3.4. Business Information Provider

When a user clicks on any of the businesses listed in the listing screen, application navigates user to the business information screen where business name, address and current population inside the premises can be seen like in figure 4.

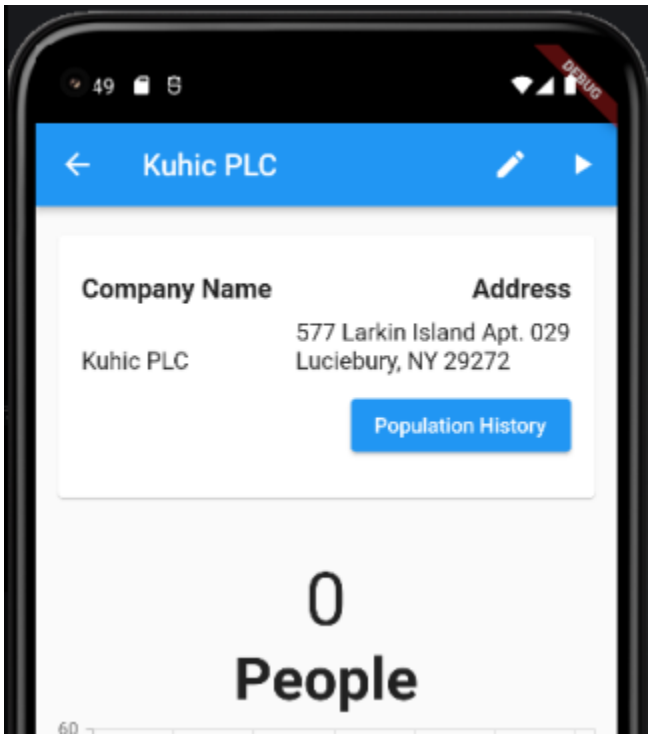

Figure 4 Business Information Screen

2.3.5. Current In-door Population Counter

When business information page gets loaded, it sends a HTTP GET request to the backend server with the business id. Then, backend server matches the business id with the business in the database and returns its current population count which can be seen in the figure 4.

2.3.6. Business Management (Admin)

Admins can perform several operations related with the businesses such as business creation, editing and deletion.

To create a new business, admin must click on the “plus” button at the top bar of the home page as seen in figure 5.

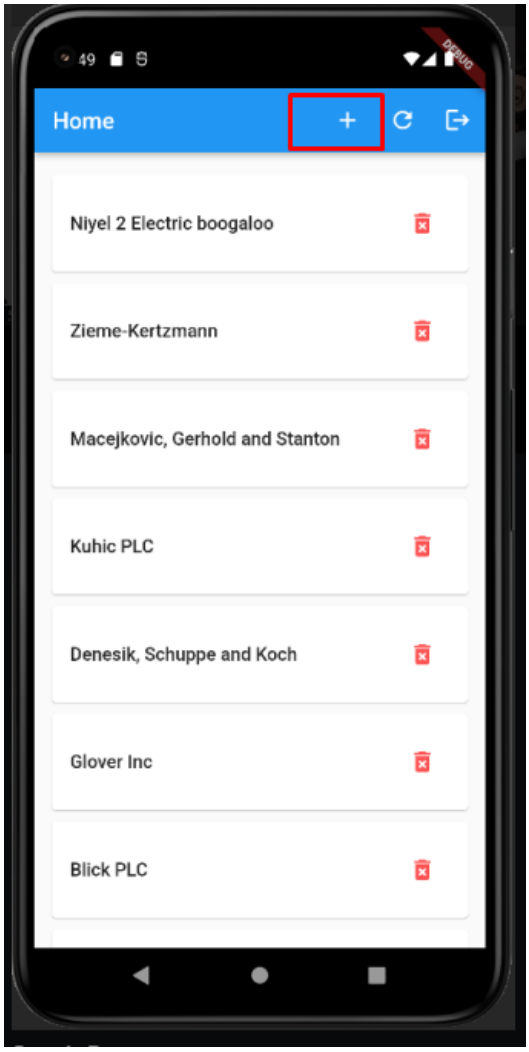

Figure 5 Business Listing Screen (Add Business)

After clicking on the create button, application navigates the user to the add business screen. Also, application records the current location of the user when page loaded for business location. Then, user must fill the form and hit the “add business” button to

create a new business. After clicking, application sends a HTTP Post request to the backend server with provided business information (name, address and location in lat and long). The backend validates the provided information and creates a new population record in the database and then creates a business record with the newly created population id.

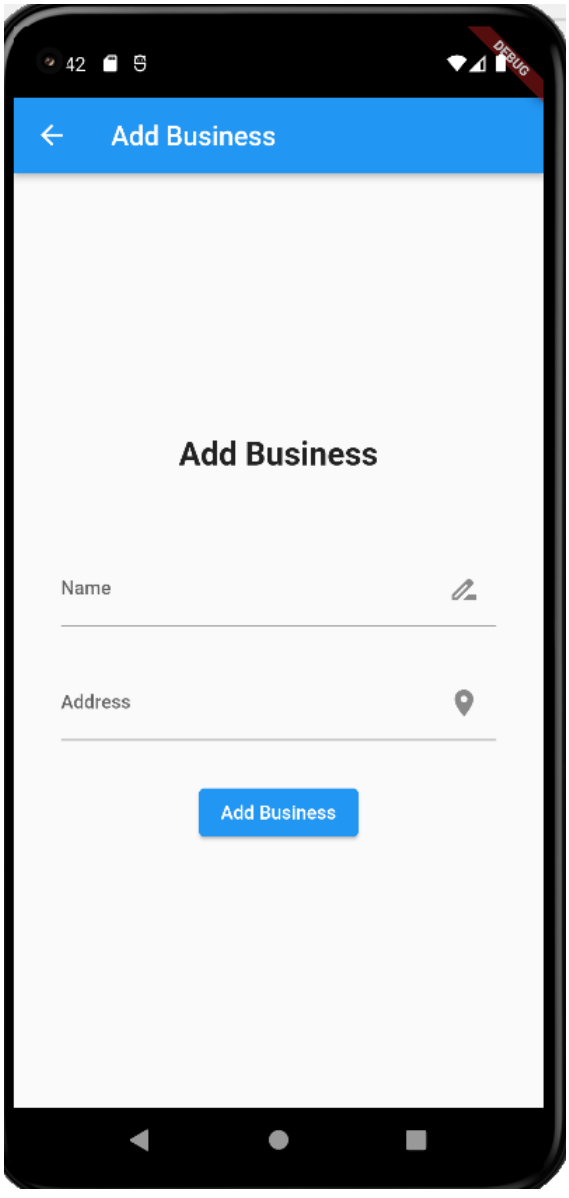

Figure 6 Add Business Screen

To delete a business, user must click on the trash icon at the right of each business listing which can be seen in the figure 7. Here, application sends a HTTP Post request to the backend server with the business id. Backend server, finds business with the

given id and then finds corresponding population record, first deletes the population and then the business.

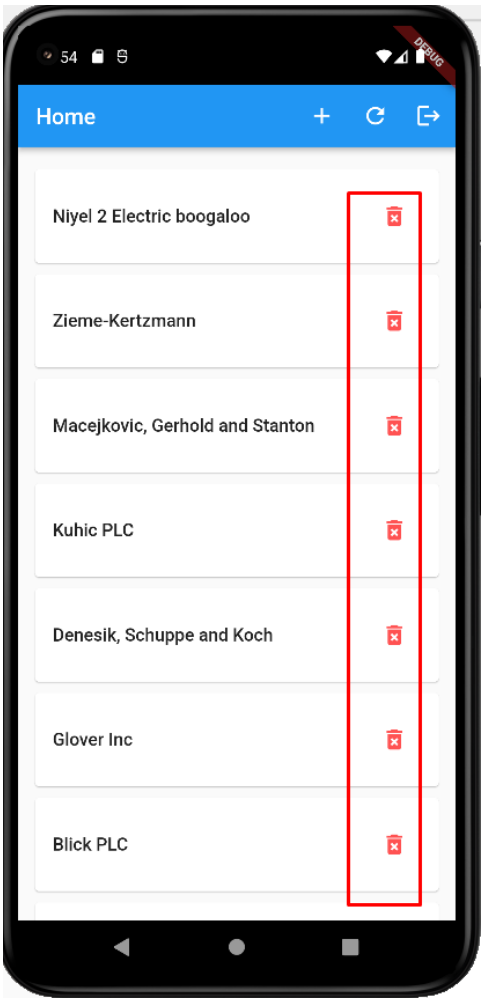

Figure 7 Business Listing Screen (Remove Business)

To update the information of any business, user must click on the “pencil” icon at the top bar which can be seen in the figure 8. Then, application navigates user to the edit business screen which user can edit the details. After the editing is done, user must press on “edit business” button where application sends a HTTP Post request with the changed information and business id to the backend server. The backend server finds the corresponding business and checks whether the request contains name or address and updates the record accordingly.

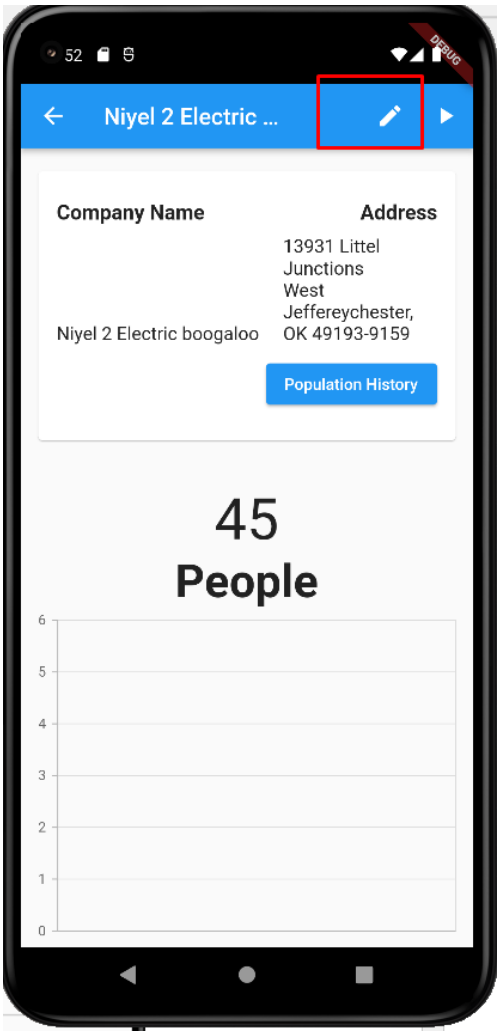

Figure 8 Business Info Screen (Edit Business Button)

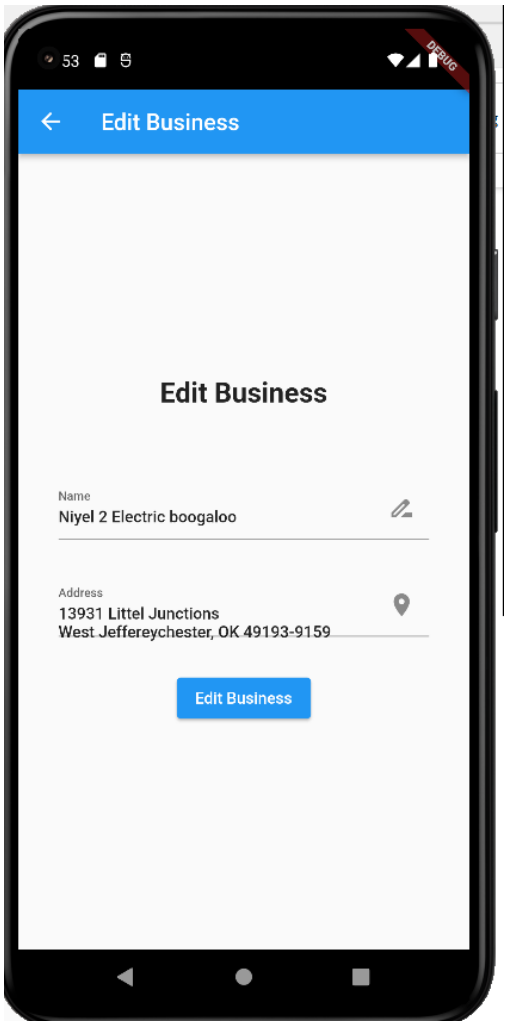

Figure 9 Edit Business Screen

2.3.7. Population Simulator (Admin)

Since we couldn't obtain the required hardware for Arduino UNO setup to implement entrance or departure of people, we decided to apply a different approach to simulate this. Application uses Poisson Algorithm for time interval of each entrance or departure where waits and generates a HTTP POST request for randomized entrance or departure with the provided business id. The backend server increments or decrements the population record of the given business depending on the request sent by the application.

If an admin wants to start the simulation for a specific business, can click on the play button at the top bar which can be seen in the figure 9.

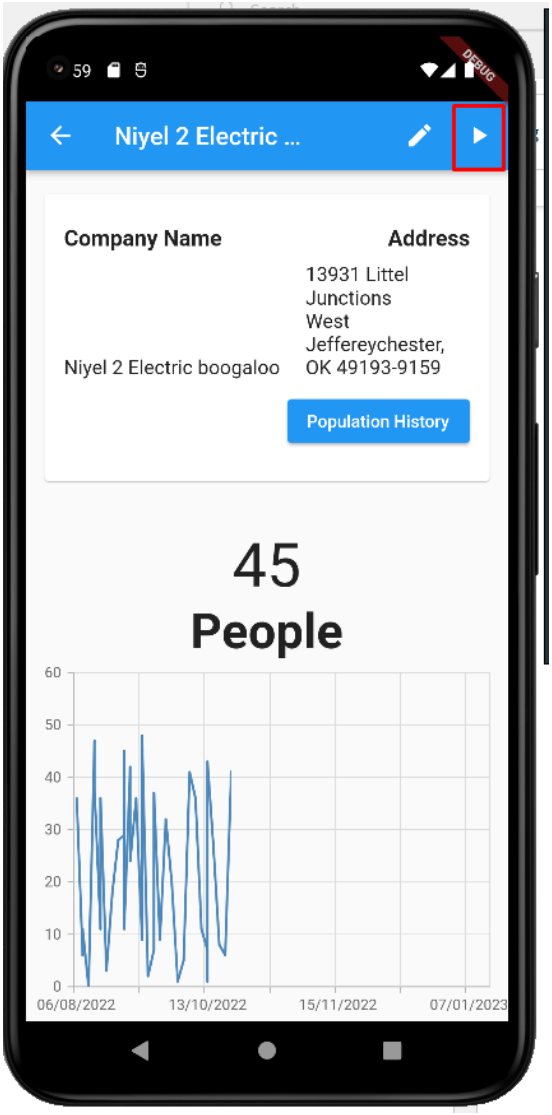

Figure 10 Business Info Screen (Simulate Button)

2.3.8. Business Population Statistic Generation (Admin)

Admins who wants to see the statistic of any business population through different date times, can access these details inside of the business information page. When business info screen first gets loaded, it sends a HTTP GET request to the backend server with the business id. The backend server finds the corresponding business and accesses the population history table to gather every population history record of the given business. Then, backend server returns a list of population history data. Application generates a

graph with the given date and population number which can be seen in the figure 10. Also, admins can access to the list of population history by clicking on the “population history” button where the listing can be seen in the figure 11.

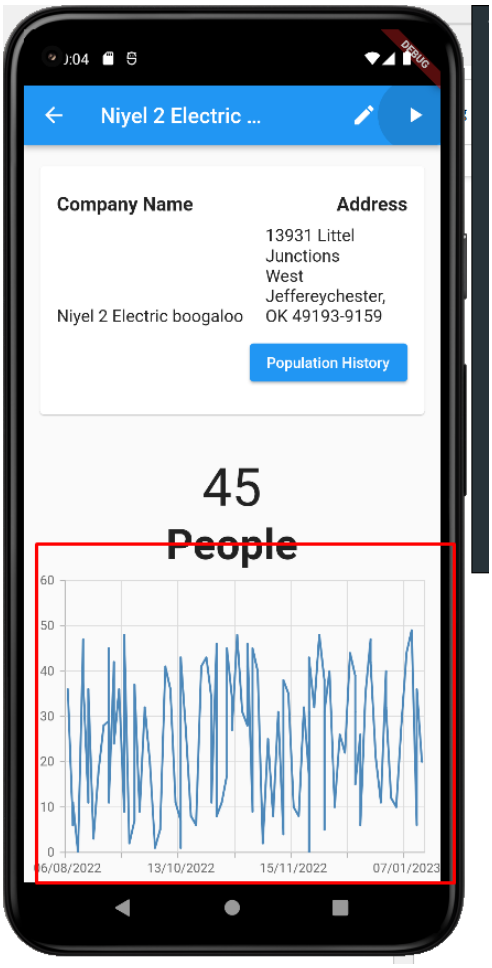

Figure 11 Business Info Screen (Population Graph)

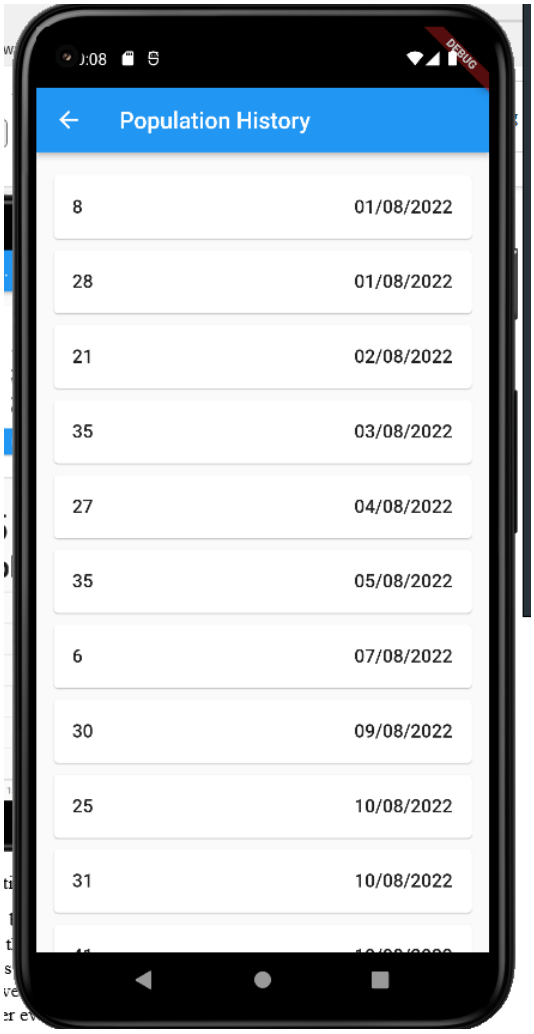

Figure 12 Population History Screen

2.3.9. Scheduled Backup Handler

The system administrator can launch the “backup.bat” file to generate a backup of the database. Backup file can be found in the “storage/app/backups” directory.

3. Project Statistics

3.1.Time Frame & Work Division

Date (Week)	Task	Responsible
Week 4 (October 24 - October 30)	Initiate AWS EC2 Server	Ibrahim Ozkan
Week 4 (October 24 - October 30)	Initiate AWS RDS Service	Adil B. Kebapcioglu
Week 4 (October 24 - October 30)	Designing API Database Structure	Adil B. Kebapcioglu
Week 5 (October 31 - November 6)	Implementing Database Structure	Adil B. Kebapcioglu
Week 5 (October 31 - November 6)	Designing Mobile Application Interfaces	Ibrahim Ozkan
Week 5 (October 31 - November 6)	Implementing Auth API	Adil B. Kebapcioglu
Week 5 (October 31 - November 6)	Integrating Auth API to mobile application (login & register)	Ibrahim Ozkan
Week 6 (November 7 - November 13)	Implementing Business CRUD API (Create, delete, update, read)	Adil B. Kebapcioglu
Week 6 (November 7 - November 13)	Integrating Business CRUD API to mobile app (Business listing, business creation, removal, information read, editing business)	Ibrahim Ozkan

Week 7 (November 14 - November 20)	Implementing Population API	Adil B. Kebapcioglu
Week 7 (November 14 - November 20)	Integrating Population API to mobile app (Current business population counter, simulation of enterance and departure, population history listing, population history stats)	Ibrahim Ozkan
Week 8 (November 21 - November 27)	Developing a database seeder at backend to generate test data	Adil B. Kebapcioglu
Week 9 (November 28 - December 4)	Hiding admin operations from user interface in mobile app	Ibrahim Ozkan
Week 10 (December 5 - December 11)	Creating scheduled backup at backend	Adil B. Kebapcioglu
Week 11 (December 12 - December 18)	Creating batch script for easy backup	Ibrahim Ozkan
Week 12 (December 19 - December 25)	Deploying project to EC2 instance with RDS connection	Ibrahim Ozkan
Week 13 (December 26 - January 1)	Testing the application	Ibrahim Ozkan, Adil B. Kebapcioglu

Week 14 (January 2 - January 8)		

Table 1 Time Division Table

3.2. Lines of Code

Language Used	Lines of Code
PHP	2000
Dart	1700
Batch Script	30

Table 2 Lines of Code Table

3.3.Hardware Requirements

Server requirements (Recommended):

- 4GB Ram
- Intel Xeon E5 2697v2 4 core
- 40GB SSD

Application Requirements:

- Android 9 Pie
- SDK version 30

3.4.Database Technologies

MYSQL (Provided by Cloud):

- 20GB DB Storage
- 1GB memory
- 1 vCPU 3.3 GHz Intel Scalable Processor