Ramapo College of New Jersey

Honors Senior Project

School of Theoretical and Applied Science

Foodster

A platform for small food businesses to share their content to the world

Designed and Developed by Bibhash Mulmi

Faculty Sponsor: Dr. Scott E. Frees

Faculty Readers: Dr. Benjamin Fine

Abstract

This paper summarizes an Honors Senior Project on the research and development of a cross-platform mobile application. This application has two main User Interfaces, one for *users*, and another for *vendors* (restaurants). The user-end enables users to follow restaurants of their choice to stay updated on the deals and coupons they offer. It allows them to search the restaurants by their name and also locate nearby restaurants on a map. The vendor-end allows restaurant owners to create posts of their deals and coupons that are shared with all its followers.

This report explains the technologies used in the development of this project. It also discusses the structure of the application and its workflow

Introduction

As smartphones have progressed exponentially in the recent decade, the need for innovative mobile applications is essential now more than ever. Computer Scientists have developed a wide range of platforms and system architectures to simplify the development of mobile applications while making them more robust. These developments create a need for a new depth of knowledge that developers must understand and learn. There are numerous programming languages, various modules and libraries, and plenty of platforms to develop a mobile web application. I wanted to learn those libraries, modules, and frameworks that would help me develop a cross-platform mobile application that can run on any device.

I came up with the idea of Foodster because it encompassed all my areas of interest and all the skills I wanted to learn while working on this project. I wanted to learn to use geo-location to target nearby users for restaurants to advertise themselves. It was in my interest to implement the feature of users following restaurants and receiving their updates on deals and coupons via posts on their home page. I desired to develop a prototype of a large-scale application to learn the perks of systems architecture and design.

Technology Stack

Client-Side:

ReactJS

ReactJS is a JavaScript library developed and maintained by Facebook that lets us build interactive User Interfaces. I used ReactJS for building the entire front end of the application. Its component-based structure is highly reusable and efficient in updating and rendering every time the data changes. So, this library was a perfect fit for developing my mobile application

HTML, CSS

The basic structure of the application is written in HyperText Markup Language (HTML). I defined the root component of my application using HTML. All the React components are rendered to this root HTML component. I also used Cascading Style Sheets (CSS) to design fonts and color schemes of the application.

Ionic

Ionic is an open-source platform that lets us build cross-platform mobile web applications with a single code base. It offers high-quality mobile-optimized UI toolkits and components that can be used to generate exquisite interfaces for user interaction. Every component of Foodster is developed using Ionic.

Axios

Axios is a JavaScript library that lets us make HTTP requests from the browser. It has all the required features that my application needs. From posting requests to canceling the requests to asynchronous calls. I used Axios Library in ReactJS to make API calls to my server.

Capacitor

Capacitor is a cross-platform app runtime which lets us build web applications that run on Android and iOS devices. I used Capacitor with Ionic to develop my project by installing it directly into my project. This way, I deployed my application on android devices for testing and debugging.

Android Studio

While deploying Foodster to android devices, I used Android Studio. It provides several tools to build apps on any android device. I generated the apk file for Foodster using this platform.

Server-Side:

Flask

Flask is a web frame of Python that provides libraries, tools, and modules to help build web servers for web-applications. It is highly efficient in handling HTTP requests on the server-end. Also, it is completely Web Server Gateway Interface (WSGI) compliant. Thus, I used Flask in Python to build the entire back end of the application since the client-side of the application makes API calls only.

PyMongo

PyMongo is a module in Python that contains the tools for working with MongoDB, NoSQL databases. As Foodster uses Microsoft’s Azure Cosmos Database, which is a NoSQL database, I used PyMongo to handle the database queries from the server.

Azure Cosmos DB

Azure Cosmos DB is Microsoft's globally distributed database service that lets us develop and manage NoSQL databases. It is highly scalable, has fast, single-digit-millisecond data access, and is extremely reliable. I used Azure Cosmos DB to match the scalability of the application.

External Application Programming Interfaces (APIs)

Mapbox

Mapbox is an open-source mapping platform that provides various features and tools for geocoding. I used the Mapbox API to display restaurant locations all around the globe that exists in my database. I was also able to integrate location access and reverse geocoding into the application. The coordinates obtained from reverse geocoding is stored in the database.

Firebase Authentication

Firebase is Google's mobile platform that provides tools to develop high-quality applications. Firebase Authentication is one of the features of Firebase that I used to authenticate users in the application. It lets the app securely save the user's password in the cloud. Among its various ways of authenticating users, I used E-mail authentication in the application.

Cloud Server

Microsoft Azure

Microsoft Azure is a cloud computing platform that provides a wide variety of services such as analytics, virtual computing, storage, server hosting and more. I used Microsoft Azure to host my Flask server so that any mobile device that uses Foodster can make API calls through the internet.

Structure of the application

The entire application can be divided into three major parts: server, client, and database.

Server

Foodster's server is developed entirely in Flask framework in Python. It has a robust RESTful web service architecture. The server handles the requests from the users. These requests range from retrieving vendor information and recent posts to adding the vendors to the user's 'following' list. It also handles user and vendor details while they sign-up for Foodster by making necessary calls to the database module functions.

Client

The clients are the users and vendors that use the application. I used Android application as the client for the development of Foodster. The client-side program is developed entirely in ReactJS. It uses various frameworks and libraries for developing all the features that currently exist in the application. The client-side application can be divided into two parts:

I. User

The user has the following features:

- Login and Signup
- Search for Restaurants
- Locate Restaurants in Map
- View Restaurant's profile and posts
- Follow Restaurants
- View all posts of followed restaurants in the user's wall

II. Vendor (Restaurant Owners)

The vendor has the following features:

- Login and Signup
- Create posts
- View its posts in its wall

Database

Azure Cosmos DB stores the data of Foodster's users. There are two collections in the Foodster Database:

I. Foodster User

This collection stores the first name, last name, username (email), date of birth, and the list of restaurants (_id) followed by the user.

II. Foodster Vendor

This collection stores the name, location, location coordinates, username (email), description, and the list of posts of the restaurant.

Workflow and Features

The application starts at the login page for users. Users can navigate to the sign-up page, vendor login page, and vendor sign-up page from here.

User/Vendor Login

When the client logs into the application, it sends the user credentials with an API call to Firebase for verification. After the credentials are verified, the application then navigates the user to their homepage. If the user inputs incorrect credentials, then they are prompted with an error message at the login screen describing their mistake.

User/Vendor Signup

When the client signs up for Foodster, all the details are sent to the server and a database query is made to insert the user details into the respective collection. The application then automatically logs the user into Foodster and navigates them to their homepage.

User/Vendor Wall

The ‘wall’ is the client’s homepage. For vendors, it is the tab that shows their posts, whereas, for users, it is the tab that shows all the posts from the restaurants that they follow. All of these posts are listed in a chronological style. This sorting of posts is done by referring to the time they were posted.

Map

This tab shows the user’s location with all the restaurants that are within a certain range of the user. There is a button on the top-right corner of the map when pressed, will locate the user on the map. This component of the application uses the Mapbox API to handle all the map related calls.

Search

The search button opens a new page in the application where the user can search any restaurant by its name. The search filters the result according to the user’s input. When the user selects one of the results, the application navigates them to the profile page of that restaurant.

Vendor page

This is the profile page that has all the details of the restaurant: name, location, description, and posts.

Menu

The menu tab contains a list of pages categorized by their types: General, Help & Support, Settings & Privacy. All of these pages currently navigate the user to an “under construction” page. The Logout button at the bottom of the Menu, when clicked, logs users out of the application.

Create Post

The Create Post tab in the vendor’s home contains input options: description, valid from and valid until dates, for restaurants to post them on their wall. When the vendor makes a post, a call is made to the server that handles this information. It queries the database and stores the post.

Conclusion

Developing this project with no prior knowledge of a mobile web application turned out to be a great learning curve for me. Every step of the development process required thorough research and testing. Every implementation of the application feature required a complete understanding of the underlying concepts of a mobile web application.

Hence, my honors senior year project has honed my programming skills. Moreover, it has helped me grow as a programmer. I believe I have overcome every challenge that I had faced throughout the development phase of Foodster. After the final submission, I will keep on adding new features to this project.

The source code of my project can be found in the links below:

https://github.com/bmulmi/Foodster

https://github.com/bmulmi/Foodster_server