

Route-Constrained Family Shopping Optimization

Christian Baer, Erich Brandt, Elizabeth
Strzelczyk, Colin Thurston, Colin Willenborg,
Tavion Yrjo
Team: sdmay21-34



Project Overview

- Provide multiple heterogeneous groups with an application to optimize shopping routes
- Each group can create lists and update them (i.e., add items)
- An individual can participate in multiple groups
- Committing to a subset of items to be purchased is propagated to other participants.

- Application creates a route based on locations of users and the stores
- Have constraints the group can put on the route
 - Distance, Time
- Have multiple members start at different locations

- Both Android and Web application
 - Kotlin and React respectively



Skills Learned

- ReactJS
- Web Scraping
- ASP.NET
- Python
- Kotlin
- Communication
- Teamwork



Requirements

Constraints

- Radius of the map of stores and locations
- The time it takes to travel to different stores
- Starting the trip from home vs. varying locations
- Start time of the trip

Functional Requirements

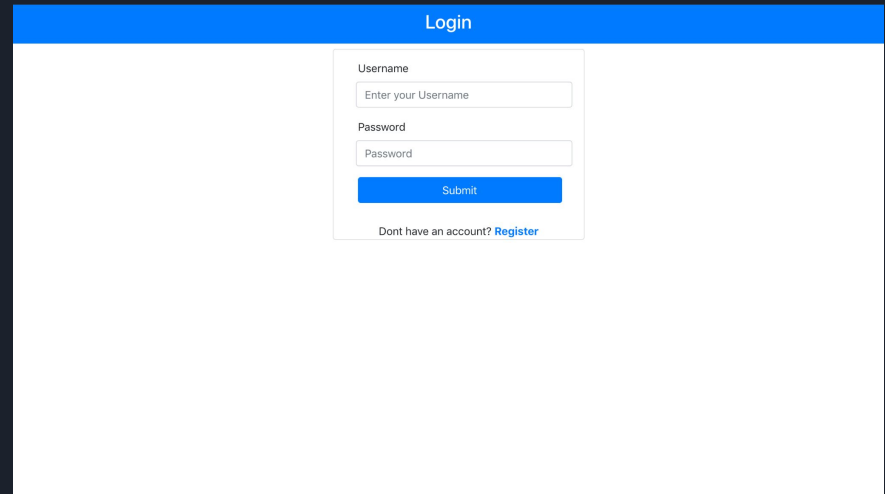
- Store location accuracy
- Outputting the closest store with desired items with respect to distance/time to travel
- Output fastest travel time to any given store at desired start time

Nonfunctional Requirements

- Routes must generate in real time
- SQL Data must be in real time
- Application must be intuitive and easy to read

Design Decisions - Overview

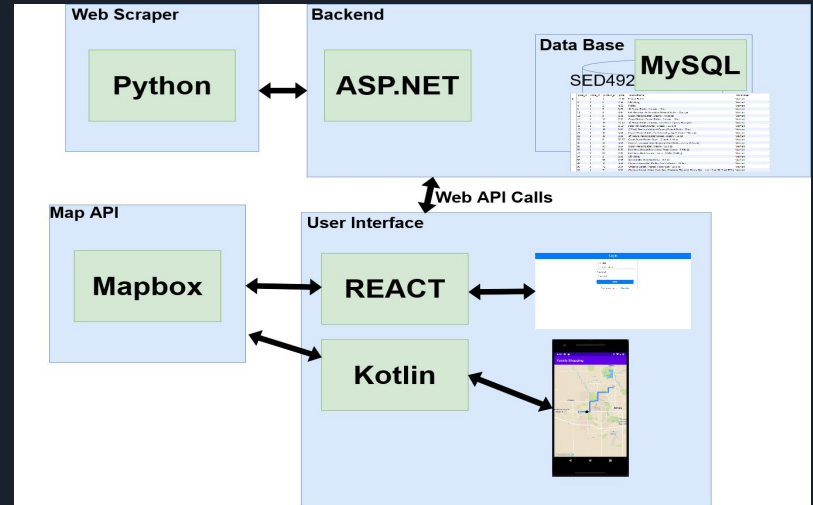
- UI
 - React
 - Web Application
 - Kotlin
 - Android
- Database
 - MySQL
 - Connection through ASP.net
- Algorithms
 - Web Scraper on the server
 - Route generation on both application
- Project Constraints
 - Distance
 - Time



The screenshot shows a login page with a blue header bar containing the word "Login". Below the header is a white rectangular form. Inside the form, there are two input fields: "Username" with the placeholder text "Enter your Username" and "Password" with the placeholder text "Password". Below these fields is a blue "Submit" button. At the bottom of the form, there is a link that says "Don't have an account? Register".

System Architecture

- Four Main Modules
 - Backend
 - User Interface
 - Map API
 - Web Scraper
- ASP.NET API retrieves and changes information from the MYSQL DB
- Both Mobile and Web Applications use API as a middleman to our DB
- Python Web Scraper deposits item information from web to DB
- Store data is sent to Mapbox via User Interface and map data is returned



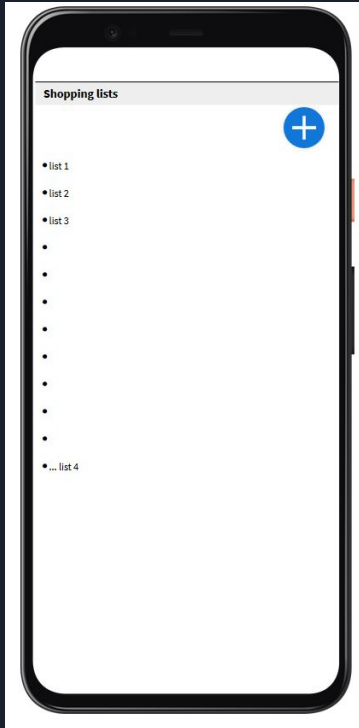
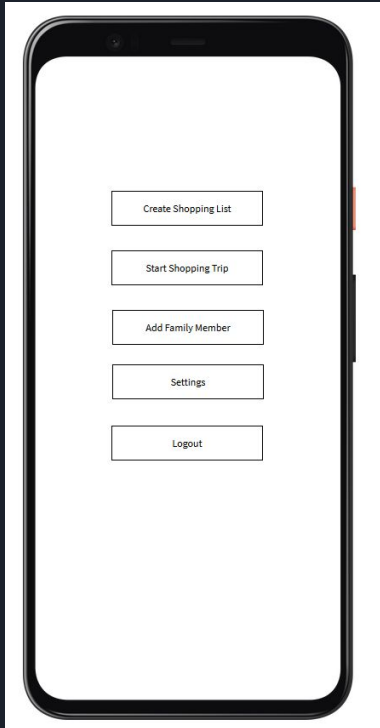


Components

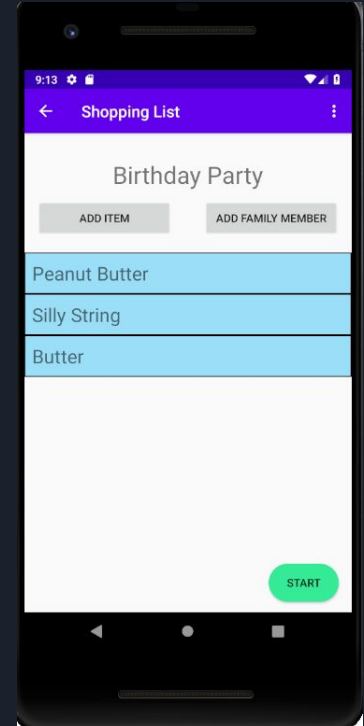
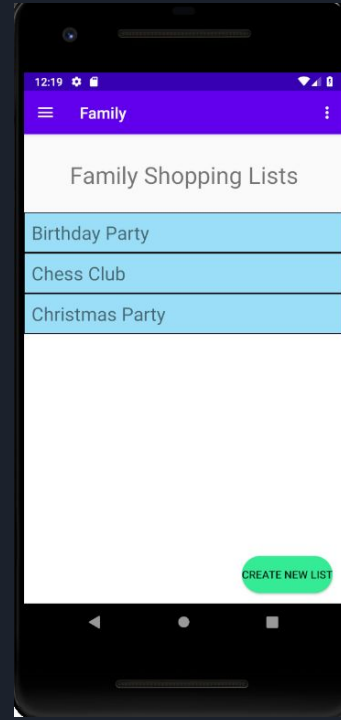
- Mapbox API to create Map in application
- User Interface is only for user interaction
- MYSQL DB to store any data that needs to be saved.
- Web API used to communicate with DB
- Web scraper to populate store inventory

Changes from UI design to implementation

Before



After





Other design changes

- Removed constraints
 - Price
- Switched the search engine
 - Google -> Bing
- Changed Shortest Path Algorithm
 - A* Algorithm -> Dijkstra's Algorithm
- Changed the focus of the project from Family to Group
 - Users can abandon and join other groups
- Live updating of shopping lists



Standards

- IEEE/ISO/IEC 14764-2006 Software Life Cycle Maintenance Standards
 - Ensure that updates are beneficial to project
 - Updates do not break other functionality in the project.
- IEEE 1008-1987 Software Testing Standards
 - Unit testing
 - Functional testing
 - Backbone for ensuring functionality



Timeline of Completion

- February 8th
 - Completed storyline for user in the application
 - Completed preliminary UI design
- February 22nd
 - Created preliminary mobile app and web app w/ basic functionality
 - Basic web scraper functionality
 - Completed initial WebAPI
- March 1st
 - Improved functionality of web app and mobile app
 - Started implementation of path finding algorithms
- March 15th
 - Connected web and mobile app to the backend
 - Improved database structure
 - Initial Mapbox implementation
- March 29th
 - Web application is able to show items from database
 - User group creation and deletion implementation
 - Web scraper complete
- April 11th
 - Routing algorithm complete
 - Added propagation of list updates to groups
 - Mapbox implementation complete on both mobile and web
 - Mobile and web app now get all necessary information from API



Testing

- Unit
 - Displaying a route
 - Updating a DB
 - UI testing of Mobile and Web
- Interface
 - Web scraping for items' locations and DB
 - Displaying items and routes from algorithms on the frontend
 - Database able to accept and send data to frontend
- Acceptance
 - Verify use cases were met
 - Advisor signed off on demo
 - Review requirements/functionality of project

```
import unittest
import WebScraping
class TestWebScrapper(unittest.TestCase):
    def test_NonNull(self):
        self.assertIsNotNone(WebScraping.productJson, "Should return the obj")

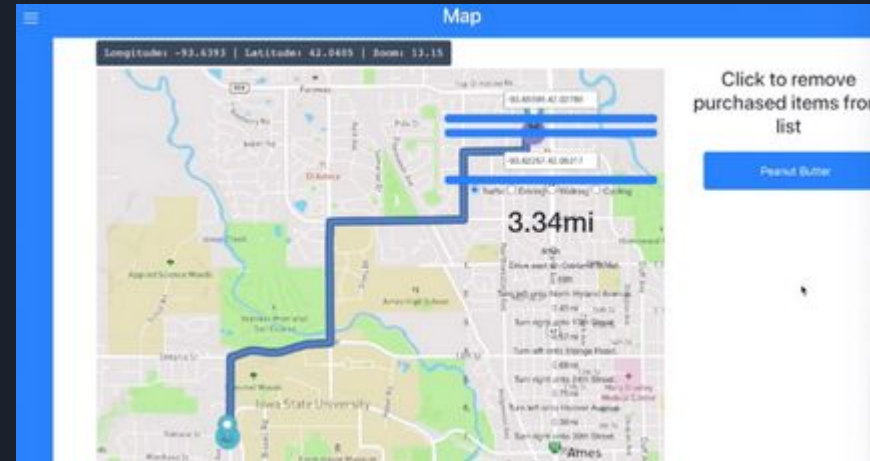
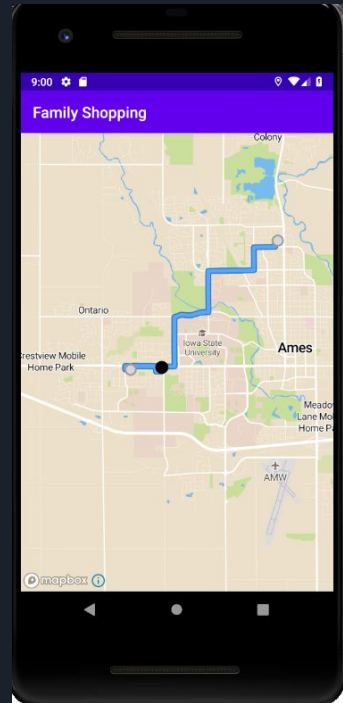
    def test_Around10(self):
        self.assertGreaterEqual(len(WebScraping.productJson), 10, "Should have 10 results")
```

Demo



Final Outcome

- We have completed our design and implemented all steps of functionality for our project.
- Web Application
 - React and hosted on cyshopper.gear.host
- Android application
- Web API
 - Bridge between frontend and backend
- Database
- Web Scraper
 - On server called by frontend
- Route Generator
 - On server called by frontend





Future Extensions

- Have a more in depth security and privacy
 - Our project assumes that security is done on a basic level
- Have notifications for sales on items
 - Promotes getting certain items
- Extend options to certain stores to better the item database
 - Better idea for availability, price, and location
- Stores can promote sales on the applications
 - Incentive for stores to participate in apps

Questions?

