

Florida International University
Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: BudgetMunch

Team Members: Trevor Simpson, Tito-Maurice Fynn, Karina Mora, Arely Correa-Perez

Product Owner(s): Arely Correa-Perez

Mentor(s): N/A

Instructor: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

The BudgetMunch application processes user input (address and budget) through the frontend using Axios requests, which sends the data to the backend. The frontend allows for main user customization and login credentials for the user's profile. The backend retrieves this input, uses the Google Places API to fetch data filtered by the user's location and budget, and sends the results back to the React table on the frontend. Built with Java and Spring, the backend manages HTTP requests, while PostgreSQL serves as the central database. Docker is used to hold the container image of the application allowing the user to interact with sophistication of the application. The user interface, developed with React and JavaScript, delivers a smooth and interactive experience for our fellow Budget Munchers.

Table of Contexts

Introduction.....	5
Current System	5
Purpose of New System.....	5
User Stories.....	5
Implemented User Stories.....	5
Pending User Stories.....	6
Project Plan.....	6
Hardware and Software Resources.....	6
Sprints Plan.....	7
Sprint 1.....	7
Sprint 2.....	7
Sprint 3.....	7
Sprint 4.....	7
Sprint 5.....	8
Sprint 6.....	8
Sprint 7.....	8
System Design.....	9
Architectural Patterns.....	9
System and Subsystem Decomposition.....	9
Deployment Diagram.....	9
Design Patterns.....	9
System Validation.....	10
Glossary.....	11
Appendix.....	12
Appendix A – UML Diagrams.....	12
Appendix B – User Interface Design.....	14
Appendix C – Sprint Review Reports.....	15
Appendix D – Users Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents.....	17
References.....	19

INTRODUCTION

BudgetMunch is an application that gives users a straightforward way of looking up a restaurant that is close to the address of their choice, that fits the price level that is within their budget range. By doing so, our application allows users to spend less time researching restaurants and more time enjoying food and company.

Current System

Google Maps currently has a feature where it displays dollar signs to give an indication of what the price level of a searched restaurant may be, it is quite ambiguous to the user without doing some more digging. Other applications, such as Uber, show specific prices once a user selects a particular restaurant and menu.

Purpose of New System

This application is designed to give users explicit price ranges for nearby (based on user provided address) restaurants based on the user's budget.

User Stories

There were eight user stories accepted during the production of BudgetMunch. Each user story was used to further enhance the application's progression. The following section provides an overview of the user stories implemented and possible future user stories.

Implemented User Stories

- Setting up the installation of tools, creating a GitHub and renewing licenses
- Connect the Google Places API with the frontend
- Create a user registration page
 - Make it functional to create and save user information
- Enabling address translation to coordinate detail to locate information on google maps
- Design and style a homepage input page

- Create a Web UI Experience
 - Allow users to navigate to different pages
 - Allow users to login and logout of personalized profile
- Merge all different GitHub branches
- Create a group poster and individual posters
- Test the code for flaws in design and errors

Pending User Stories

- Hosting the application.
- Authentication of users
- Displaying menus of budget meals

PROJECT PLAN

Have you ever struggled with finding where to eat and if it's affordable. BudgetMunch is an application that helps the user find local food options within the price range that the user finds sufficient. BudgetMunch is an application that offers users the ease of looking up a restaurant nearby that fits the price level that is within their budget range. By doing so, our application allows users to spend less time researching restaurants and more time enjoying food & company.

Hardware and Software Resources

Hardware and IEDs

- MacOS
- Windows
- IntelliJ
- Visual Studio Code

Framework and software

- CSS Bootstrap
- Docker
- Postman
- PowerPoint
- Canva
- Maven
- Spring

- React
- Axios
- Node

Languages

- PostgreSQL
- Java
- JavaScript
- Command line/Terminal

Sprints Plan

Sprint 1

As a team we decided to work on these following items/user stories for this sprint:

- Developers reviewed and installed all the necessary licenses to the tools we used throughout the application. **[5 Points: Trevor, Tito, Arely, Karina]**

Sprint 2

As a team we decided to work on these following items/user stories for this sprint:

- Developers focused on connecting to the google API information to the database set up in from the previous sprint while also having that information displayed in the front end. **[8 Points: Trevor, Tito, Arely, Karina]**

Sprint 3

As a team we decided to work on these following items/user stories for this sprint:

- Worked on translating the input address to coordinates that could be read on a map to track current location. **[8 Points: Trevor, Tito]**
- Developers created a registration form to start the process of creating a personalized user experience. **[8 points: Arely, Karina]**

Sprint 4

As a team we decided to work on these following items/user stories for this sprint:

- Created a user interface that is easily accessible for users to input their current address and retrieve the information of the restaurants that are nearby. **[5 Points: Trevor, Tito]**
- Connected the front end with the backend to display the retrieved information of nearby restaurants. **[5 Points: Karina, Arely]**

Sprint 5

As a team we decided to work on these following items/user stories for this sprint:

- Developers worked on the various pages to route them to selected buttons. **[8 Points: Karina, Tito, Trevor]**
- A navigation bar and sidebar were created to allow users to navigate between the various pages. **[8 Points: Karina, Arely, Tito]**
- Created a verification of email addresses and other basic user profile buttons to allow for ease of use. **[5 Points: Arely, Trevor]**

Sprint 6

As a team we decided to work on these following items/user stories for this sprint:

- Merged all the branches to resolve any frontend and backend errors or conflicts through GitHub. **[13 Points: Arely, Karina, Tito, Trevor]**

Sprint 7

As a team we decided to work on these following items/user stories for this sprint:

- Developers worked on creating the posters and working on the final documentation. **[5 Points: Karina, Tito, Trevor]**
- Prepared the presentation materials for the showcase. **[5 Points: Arely]**
- The application was tested to check for any errors that needed to be corrected. **[8 Points: Trevor, Tito, Arely, Karina]**

SYSTEM DESIGN

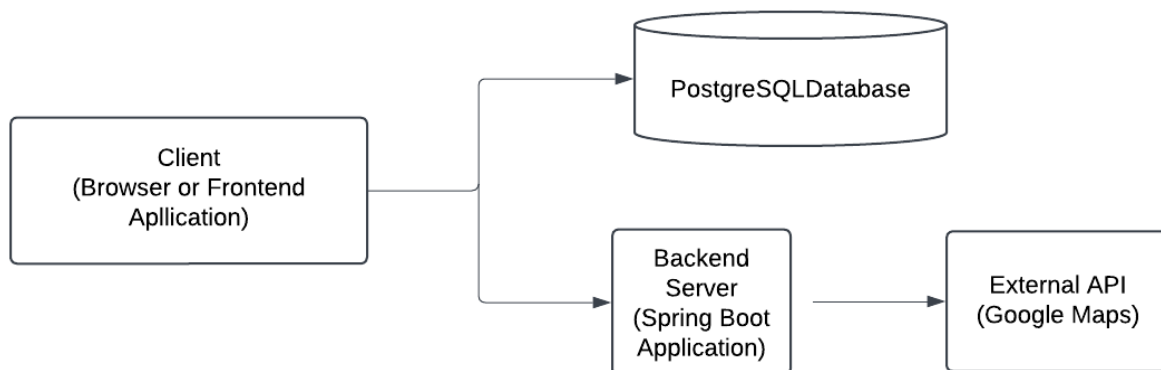
BudgetMunch is an application that utilizes Java and Spring on the backend with Maven as the dependency management. The application relies on Spring web and JPA to communicate with the frontend's React and Axios to allow JavaScript to execute the webpage's logic. Docker is used to host an image container for the backend on a URL server to host the WAR file from spring.

Architectural Patterns: *Controller Layer* (Handles incoming HTTP requests and responses), *Service Layer* (manages operations such as user verification, address validation, and interaction with external APIs Google Maps API), *Data Access Layer* (Provides data persistence using Spring Data JPA and its CRUD Repository), *React and Axios* (Used to stage the UI for the users' inputs and profile)

System and Subsystem Decomposition

- 1) **User Management Subsystem:** Handles user registration, authentication, and password reset.
- 2) **Address and Geolocation Subsystem:** Manages user addresses and retrieves geolocation data using Google Maps API.
- 3) **Email Notification Subsystem:** Sends emails for password reset and other notifications.

Deployment Diagram



Design Patterns: *Model-View-Controller (MVC)*

- Model: Represents data (e.g., User, UserAddress, Result).
- View: Not applicable here as the backend is a REST API.

- Controller: Handles HTTP requests (MapController)

System Validation: The application successfully functions on the backend to store the user's information such as username and password. The information once stored will save the user's profile preferences on a remote database to use as authentication for the frontend. Once the frontend receives the verification, the user will receive a response for either successful or invalid (login and registration).

1) FRONTEND: USER INPUT = 11200 SW 8 ST, MIAMI, FL + \$PRICE

HTTPS://MAPS.GOOGLEAPIS.COM/MAPS/API/PLACE/TEXTSEARCH/JSON?QUERY=11200%20SW%208%20ST%20&KEY=**APIKEY**

2) BACKEND: PARSES (**LATITUDE & LONGITUDE**)

HTTPS://MAPS.GOOGLEAPIS.COM/MAPS/API/PLACE/NEARBYSEARCH/JSON?&KEYWORD=RESTAURANT&LOCATION=" + **LAT** + ',' + **LNG** + "&RADIUS=1000&KEY="+**APIKEY**

"RESULTS" : (THE LOCATION OF THE RESTAURANTS IN THE AREA)

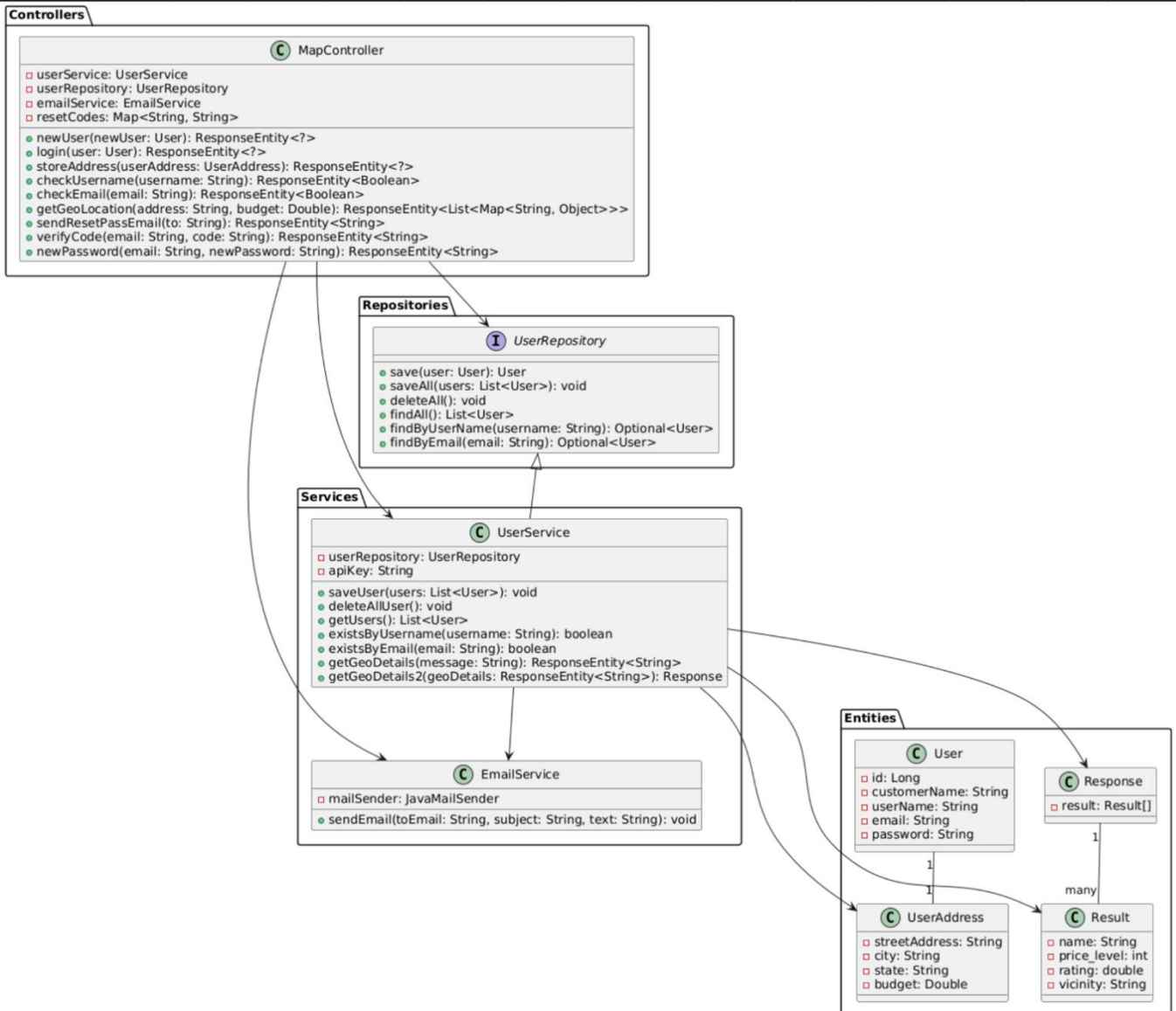
```
[
  {
    "BUSINESS_STATUS" : "OPERATIONAL",
    "GEOMETRY" :
    {
      "LOCATION" :
      {
        "LAT" : 25.7476133,
        "LNG" : -80.38917699999999
      },
      "HTTPS://MAPS.GSTATIC.COM/MAPFILES/PLACE_API/ICONS/V2/RESTAURANT_PINLET",
      "NAME" : "LOS CHINGONES MEXICAN GRILL"
    }
  }
]
```

GLOSSARY

- **Maven:** Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
- **Spring Boot:** makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.
- **Docker:** The Docker Engine powers your containerized applications with high performance and reliability. It provides the core technology for building and running containers, ensuring efficient and scalable operations.
- **Postman:** Prototype, document, test, and demo all your APIs in one place. Get early feedback by having conversations in the context of any API—private, public, or partner—not across scattered across tools.
- **Node.JS:** a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.
- **Axios:** Axios, which is a popular library, is used to send asynchronous HTTP requests to REST endpoints. This library is extremely useful for performing CRUD operations.
- **React:** React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook. React is a tool for building UI components.

APPENDIX

Appendix A - UML Diagrams

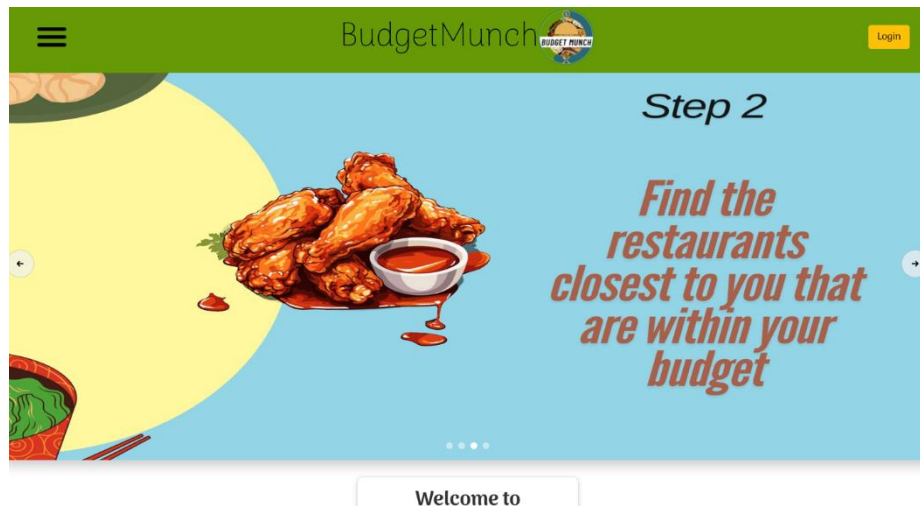


Database Schema

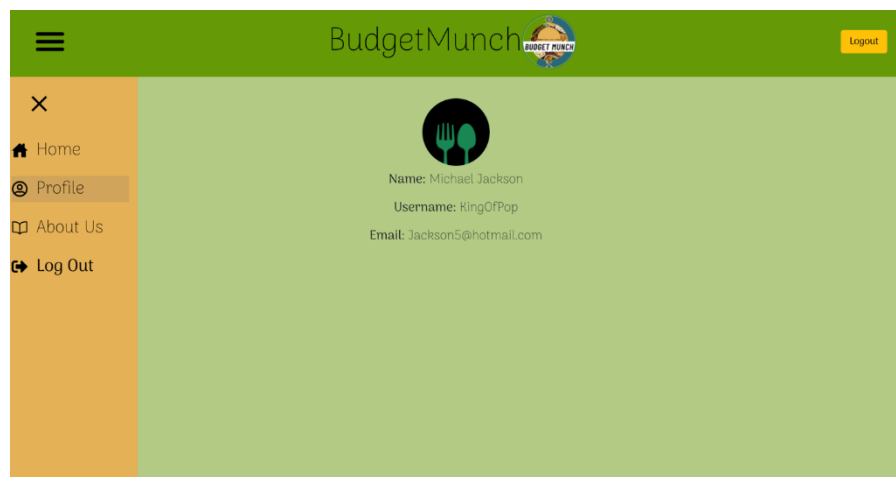
Users			
id		BIGINT	NN
name		VARCHAR(255)	NN
email		VARCHAR(255)	NN
username		VARCHAR(100)	NN
password		VARCHAR(255)	NN

Appendix B - User Interface Design

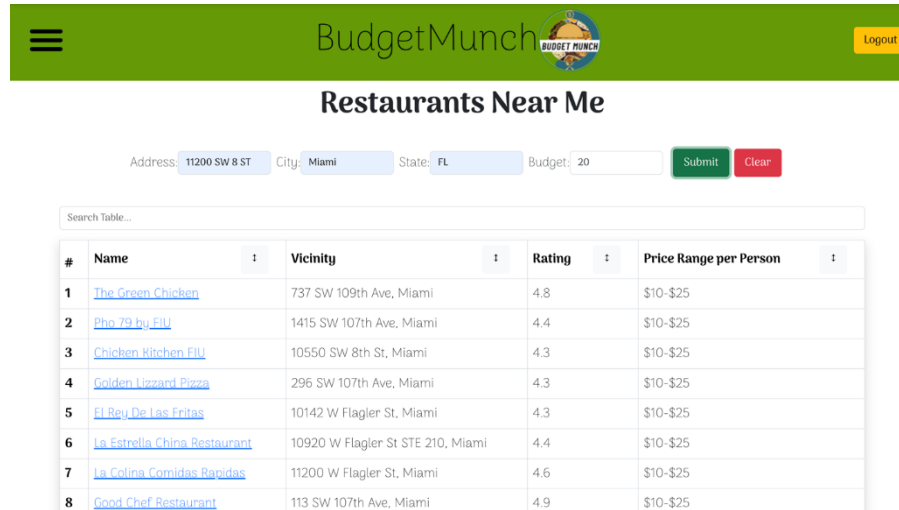
- **About Us page**



- **Profile page**



- **Home page**



The screenshot shows the BudgetMunch website interface. At the top is a green header with a hamburger menu icon, the "BudgetMunch" logo, and a "Logout" button. Below the header is the title "Restaurants Near Me". A search bar contains the following information: Address: 11200 SW 8 ST, City: Miami, State: FL, Budget: 20. There are "Submit" and "Clear" buttons next to the budget field. Below the search bar is a table with 8 rows of restaurant data.

#	Name	Vicinity	Rating	Price Range per Person
1	The Green Chicken	737 SW 109th Ave, Miami	4.8	\$10-\$25
2	Pho 79 by Fiu	1415 SW 107th Ave, Miami	4.4	\$10-\$25
3	Chicken Kitchen Fiu	10550 SW 8th St, Miami	4.3	\$10-\$25
4	Golden Lizard Pizza	296 SW 107th Ave, Miami	4.3	\$10-\$25
5	El Rey De Las Fritas	10142 W Flagler St, Miami	4.3	\$10-\$25
6	La Estrella China Restaurant	10920 W Flagler St STE 210, Miami	4.4	\$10-\$25
7	La Colina Comidas Rapidas	11200 W Flagler St, Miami	4.6	\$10-\$25
8	Good Chef Restaurant	113 SW 107th Ave, Miami	4.9	\$10-\$25

Appendix C - Sprint Review Reports

Sprint 1

Set up and installation of the project was a successful endeavor established by all team members.

- ☐ Set up and installation to be able to easily utilize the database tools, and frontend/backend environments.
- ☐ Confirmed the same starter code is being used to continue the project while ensuring all team members have the ability to merge and create branches.

Sprint 2

- ☐ Focus on storing restaurant information in the database.
- ☐ Connect the database with the frontend to display the information stored in the database.
- ☐ Be able to adjust the information from the frontend to limit how much information is shown on the frontend.

No user stories/work items were left unfinished this sprint.

Sprint 3

- ☐ Be able to get a user's location and filter the restaurants that were close by the user
- ☐ A user profile could be created with the user's profile information and stored in a database for future use.

No user stories/work items were left unfinished this sprint.

Sprint 4

- ☐ Send information to the backend from the frontend to store information variables
- ☐ The frontend table set up with populated restaurants.

These following user stories were unfinished (added to backlog):

- ☐ Host the website

Sprint 5

- ☐ Work on tutorials and webpage navigation for users.
- ☐ Transfer registered information about using to the profile page and user can sign out.
- ☐ Create a way to reset password for an account created.

These following user stories were unfinished (added to backlog):

- ☐ Create a heart icon to transfer favorite restaurants to a private list.

Sprint 6

- ☐ Test code for errors and resolve conflicts after a merge of different branches.
- ☐ Resolve styling conflicts from different branches and fix front-end errors.

These following user stories were unfinished (added to backlog):

- ☐ Host the website

Sprint 7

- ☐ Create posters to be used during presentation of project.
- ☐ Retest code to look for any errors and work to fix them.
- ☐ Prepare for the presentation of final project by creating PowerPoints, video and documentation.

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document, and other documents

Installation:

Frontend: Pre-reqs: Must have Node.js version >= installed

1. Clone the repository
2. CD into the correct repository at least twice.
 - a. First for FrontendBudgetMunch and then again for budgemunchfrontend.
3. Install npm by running the command 'npm install' in the terminal.
4. Install react-router-dom by running the command 'npm install react-router-dom' in the terminal.
5. Install react-icons by running the command 'npm install react-icons' in the terminal.
6. At this point, even if the backend is running or not running, you can enter 'npm start' to start the application on the web browser.
 - a. If the backend is running already, the application can be used as intended immediately after logging in.

1) Backend: Pre-reqs must have Java 17 Installed

- Spring initializer
- .env file must contain

```
DATASOURCE_URL=JDBC:POSTGRESQL  
DATASOURCE_USER=?  
DATASOURCE_PASSWORD=?  
API_KEY=?
```

- maven commands = mvn clean package (for .WAR file)
- java -jar target/'Snapshot version'.jar (Test Locally)
- docker must be installed, recommend docker desktop
- docker buildx build --platform linux/amd64 (**push project name**)
- Render.com host server for .WAR

SHORTCOMINGS/WISHLIST

- ☐ Incorporate actual menu items within the price range of the user's budget.
- ☐ Allow for restaurant searches based on user's current location.
- ☐ Allow purchase of menu items directly from the BudgetMunch application.
- ☐ Allow users to create a favorites list, which would be displayed in the user's profile.

REFERENCES

- 1) www.w3schools.com/REACT
- 2) www.geeksforgeeks.org/axios-in-react-a-guide-for-beginners
- 3) <https://nodejs.org/en>
- 4) www.postman.com
- 5) www.docker.com
- 6) <https://spring.io/>
- 7) maven.apache.org/