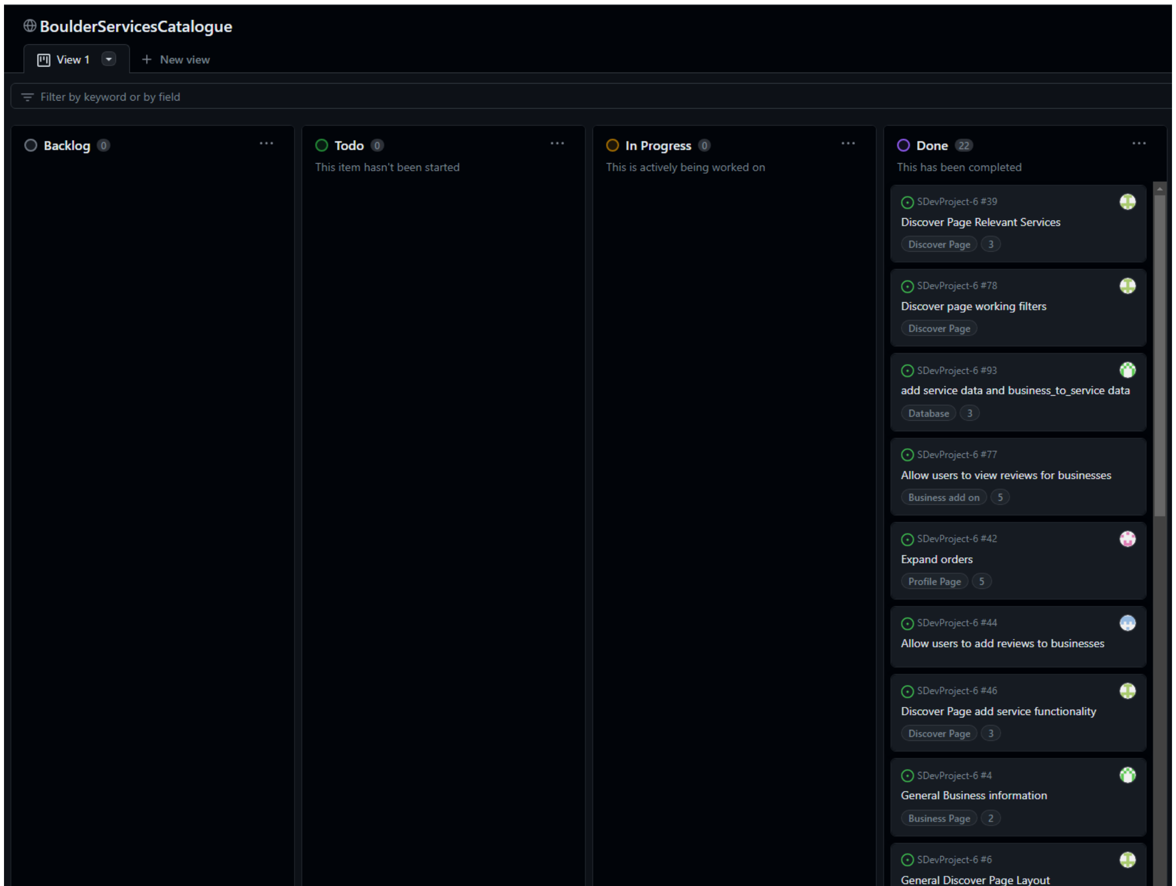
Title: Boulder's List

Who: Jon Jablonski, Oscar Carlek, Moses Williams, Eldin Basic, Dan Gagnier, and Harald Riisager

Project Description: Boulder's List is a local business finder platform specifically designed for the Boulder community. It serves as a bridge connecting residents with a wide array of local businesses and services. The platform is intuitive and user-friendly, allowing users to easily browse and discover what local businesses offer. It also enables registered users to book services and write reviews for businesses they have utilized, fostering a community-centric environment. Additionally, this platform allows for savvy entrepreneurs to upload their own businesses and services to advertise to the local audience in Boulder. The technology stack and tools used in Boulder Services are carefully chosen to support its functionality and user experience. The server-side runtime environment for executing JavaScript was Node.js, PostgreSQL was chosen for the database, and embedded JavaScript was used for crafting the frontend interface. We utilized RapidAPI for getting real time local business data and finally containerized the application in Docker container to ensure consistency across different development and deployment environments.

Project Tracker - GitHub project board:

https://github.com/users/osca3666/projects/1/views/1



SDevProject-6 #39
Discover Page Relevant Services
Discover Page 3

SDevProject-6 #78
Discover page working filters
Discover Page

SDevProject-6 #93
add service data and business_to_service data
Database 3

SDevProject-6 #77
Allow users to view reviews for businesses
Business add on 5

SDevProject-6 #42
Expand orders
Profile Page 5

SDevProject-6 #44
Allow users to add reviews to businesses

SDevProject-6 #46
Discover Page add service functionality
Discover Page 3

SDevProject-6 #4
General Business information
Business Page 2

SDevProject-6 #6
General Discover Page Layout

Video: https://youtu.be/W7vTMOfvv60

VCS: https://github.com/osca3666/SDevProject-6

Contributions:

Jon Jablonski: Created the individual business page which shows the business information as well as the services attached to each business. Also created the service page which allows users to see more service information and place an order for that service as well as getting the external database data downloaded into our own local database. Assisted in the functionality of the home page.

Oscar Carlek: Created the business page originally called the discover page of the website. Made the business cards layout, styling, and functions. Connected the business page to the local services API in order to obtain real time data of local services. Also connected the business table of the local database to the business page so locally submitted businesses can be displayed. Implemented a drop down menu to select the type of business and filter the business results based on the type of business like plumbing or roofing for example. Worked on any bugs and errors generated by the business page to ensure a polished final product.

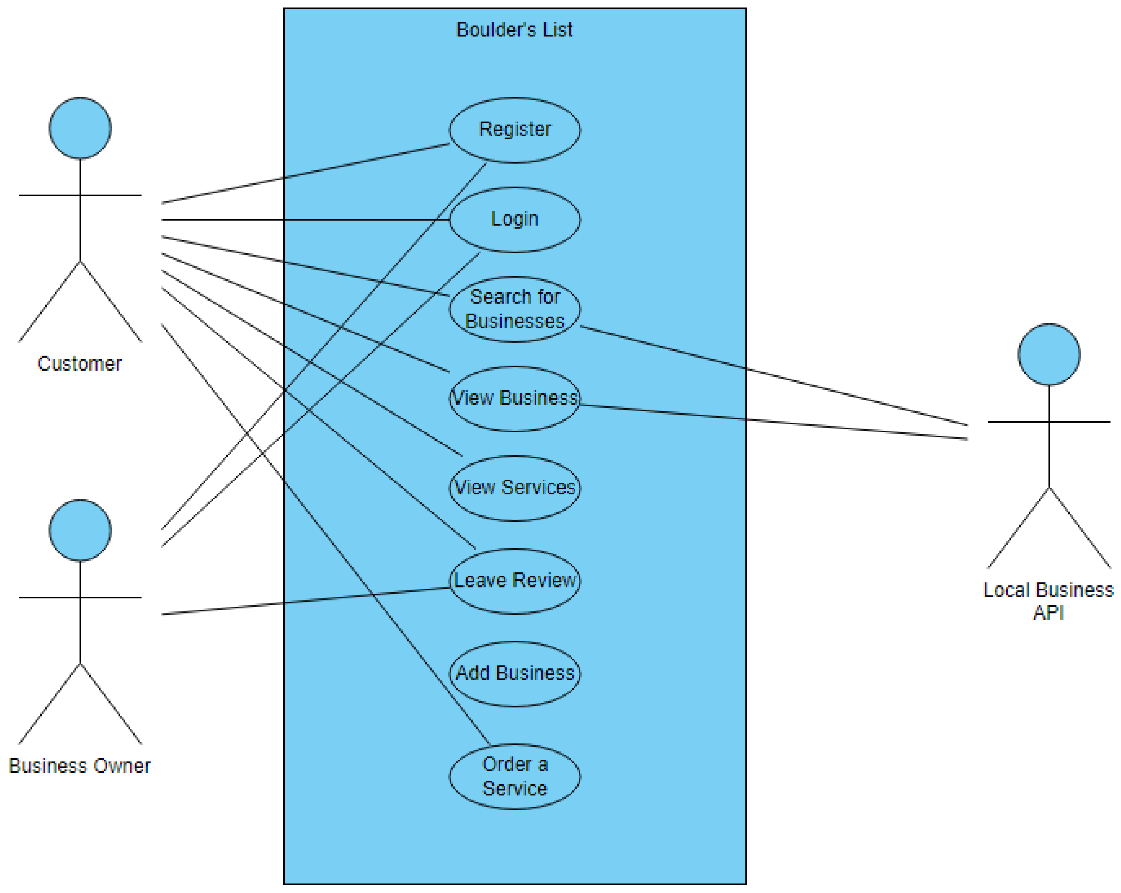
Eldin Basic: Created functionality for placing and adding orders within the database on the website. Implemented the profile page that dynamically displays information about the user along with dynamically displaying orders on the page when they would come in with details about what the user purchased. Orders were organized by time to allow the newest order to be at the bottom of the page. Also, worked on fixing the broken links on each page and making sure that each link sent you to a proper part of the website without any white/black screens. Additionally, made the logo for the business

Moses Williams: Created the home page which displays the top 3 rated businesses as well as links to various filters on the discover page. Added custom styling for the business cards on the homepage to improve user experience. Also created the ability for someone to add their own business to our local database which will then show up as an option in our businesses page. Helped make sure the home page doesn’t have any non functional components or unaccounted for error page generations.

Harald Riisager: Assisted in creating the reviews system, made the reviews modal using bootstrap, and made the reviews show up under each business. Also helped in the creation of the login and register system. Created the filter by rating system on the discover page so that users can find more relevant businesses for their needs. Furthermore I helped with several backend functions, such as ensuring that the DOM was working correctly. Wrote the readme.md page in github to give the motive behind the project and directions on how to access Boulder's List.

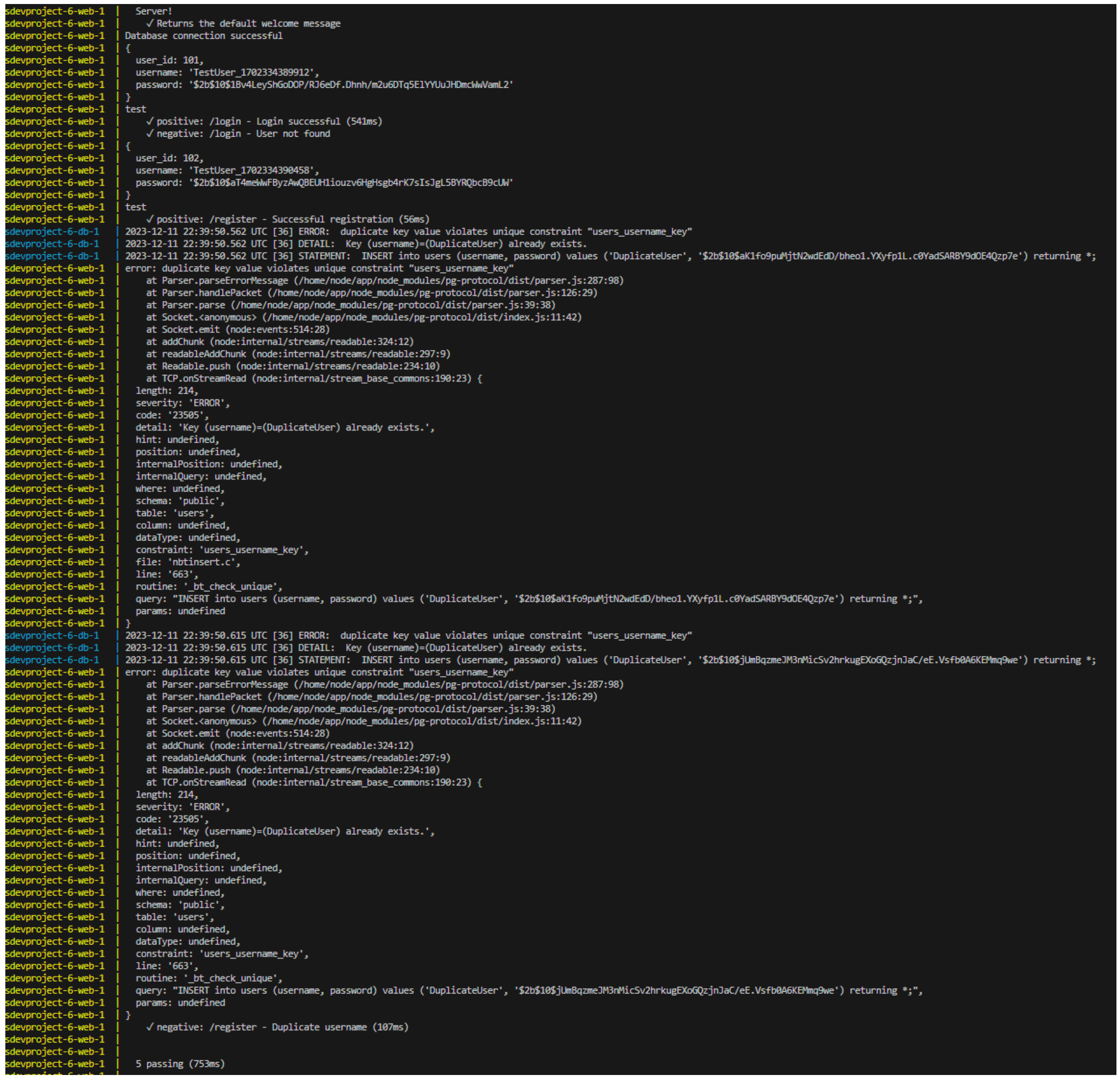
Dan Gagnier: Assisted in creating the reviews system. Created the login/registration features to allow users to register with their own account and then login to access the rest of the website. Implemented user sessions so that when users who have logged in once aren't asked to log in again provided the session remains active. Cleaned up the entire website with CSS to make it look more coherent. Deployed the application to Azure to allow anyone to access it but the link doesn't work currently due to the using the free version of Azure.

Use Case Diagram:



Test results:

/login and /register Test Cases Required for Implementation in Lab 11



```
sdevproject-6-web-1  |   Server!
sdevproject-6-web-1  |     ✓ Returns the default welcome message
sdevproject-6-web-1  | Database connection successful
sdevproject-6-web-1  | {
sdevproject-6-web-1  |   user_id: 101,
sdevproject-6-web-1  |   username: 'TestUser_1702334389912',
sdevproject-6-web-1  |   password: '$2b$10$1Bv4LeyShGoDOP/RJ6eDf.Dhnh/m2u6DTq5ElYYUuJHDmcWwVamL2'
sdevproject-6-web-1  | }
sdevproject-6-web-1  | test
sdevproject-6-web-1  |     ✓ positive: /login - Login successful (541ms)
sdevproject-6-web-1  |     ✓ negative: /login - User not found
sdevproject-6-web-1  | {
sdevproject-6-web-1  |   user_id: 102,
sdevproject-6-web-1  |   username: 'TestUser_1702334390458',
sdevproject-6-web-1  |   password: '$2b$10$aT4meWwFByzAwQBEUH1iouzv6HgHsgb4rK7sIsJgL5BYRQbcB9cUW'
sdevproject-6-web-1  | }
sdevproject-6-web-1  | test
sdevproject-6-web-1  |     ✓ positive: /register - Successful registration (56ms)
sdevproject-6-db-1   | 2023-12-11 22:39:50.562 UTC [36] ERROR:  duplicate key value violates unique constraint "users_username_key"
sdevproject-6-db-1   | 2023-12-11 22:39:50.562 UTC [36] DETAIL:  Key (username)=(DuplicateUser) already exists.
sdevproject-6-db-1   | 2023-12-11 22:39:50.562 UTC [36] STATEMENT:  INSERT into users (username, password) values ('DuplicateUser', '$2b$10$aK1fo9puMjtN2wdEdD/bheo1.YXyfp1L.c0YadSARBY9dOE4Qzp7e') returning *;
sdevproject-6-web-1  | error: duplicate key value violates unique constraint "users_username_key"
sdevproject-6-web-1  |     at Parser.parseErrorMessage (/home/node/app/node_modules/pg-protocol/dist/parser.js:287:98)
sdevproject-6-web-1  |     at Parser.handlePacket (/home/node/app/node_modules/pg-protocol/dist/parser.js:126:29)
sdevproject-6-web-1  |     at Parser.parse (/home/node/app/node_modules/pg-protocol/dist/parser.js:39:38)
sdevproject-6-web-1  |     at Socket.<anonymous> (/home/node/app/node_modules/pg-protocol/dist/index.js:11:42)
sdevproject-6-web-1  |     at Socket.emit (node:events:514:28)
sdevproject-6-web-1  |     at addChunk (node:internal/streams/readable:324:12)
sdevproject-6-web-1  |     at readableAddChunk (node:internal/streams/readable:297:9)
sdevproject-6-web-1  |     at Readable.push (node:internal/streams/readable:234:10)
sdevproject-6-web-1  |     at TCP.onStreamRead (node:internal/stream_base_commons:190:23) {
sdevproject-6-web-1  |   length: 214,
sdevproject-6-web-1  |   severity: 'ERROR',
sdevproject-6-web-1  |   code: '23505',
sdevproject-6-web-1  |   detail: 'Key (username)=(DuplicateUser) already exists.',
sdevproject-6-web-1  |   hint: undefined,
sdevproject-6-web-1  |   position: undefined,
sdevproject-6-web-1  |   internalPosition: undefined,
sdevproject-6-web-1  |   internalQuery: undefined,
sdevproject-6-web-1  |   where: undefined,
sdevproject-6-web-1  |   schema: 'public',
sdevproject-6-web-1  |   table: 'users',
sdevproject-6-web-1  |   column: undefined,
sdevproject-6-web-1  |   dataType: undefined,
sdevproject-6-web-1  |   constraint: 'users_username_key',
sdevproject-6-web-1  |   file: 'nbtinsert.c',
sdevproject-6-web-1  |   line: '663',
sdevproject-6-web-1  |   routine: '_bt_check_unique',
sdevproject-6-web-1  |   query: "INSERT into users (username, password) values ('DuplicateUser', '$2b$10$aK1fo9puMjtN2wdEdD/bheo1.YXyfp1L.c0YadSARBY9dOE4Qzp7e') returning *;",
sdevproject-6-web-1  |   params: undefined
sdevproject-6-web-1  | }
sdevproject-6-db-1   | 2023-12-11 22:39:50.615 UTC [36] ERROR:  duplicate key value violates unique constraint "users_username_key"
sdevproject-6-db-1   | 2023-12-11 22:39:50.615 UTC [36] DETAIL:  Key (username)=(DuplicateUser) already exists.
sdevproject-6-db-1   | 2023-12-11 22:39:50.615 UTC [36] STATEMENT:  INSERT into users (username, password) values ('DuplicateUser', '$2b$10$jUmBqzmeJM3nMicSv2hrkugEXoGQzjnJaC/eE.Vsfb0A6KEMmq9we') returning *;
sdevproject-6-web-1  | error: duplicate key value violates unique constraint "users_username_key"
sdevproject-6-web-1  |     at Parser.parseErrorMessage (/home/node/app/node_modules/pg-protocol/dist/parser.js:287:98)
sdevproject-6-web-1  |     at Parser.handlePacket (/home/node/app/node_modules/pg-protocol/dist/parser.js:126:29)
sdevproject-6-web-1  |     at Parser.parse (/home/node/app/node_modules/pg-protocol/dist/parser.js:39:38)
sdevproject-6-web-1  |     at Socket.<anonymous> (/home/node/app/node_modules/pg-protocol/dist/index.js:11:42)
sdevproject-6-web-1  |     at Socket.emit (node:events:514:28)
sdevproject-6-web-1  |     at addChunk (node:internal/streams/readable:324:12)
sdevproject-6-web-1  |     at readableAddChunk (node:internal/streams/readable:297:9)
sdevproject-6-web-1  |     at Readable.push (node:internal/streams/readable:234:10)
sdevproject-6-web-1  |     at TCP.onStreamRead (node:internal/stream_base_commons:190:23) {
sdevproject-6-web-1  |   length: 214,
sdevproject-6-web-1  |   severity: 'ERROR',
sdevproject-6-web-1  |   code: '23505',
sdevproject-6-web-1  |   detail: 'Key (username)=(DuplicateUser) already exists.',
sdevproject-6-web-1  |   hint: undefined,
sdevproject-6-web-1  |   position: undefined,
sdevproject-6-web-1  |   internalPosition: undefined,
sdevproject-6-web-1  |   internalQuery: undefined,
sdevproject-6-web-1  |   where: undefined,
sdevproject-6-web-1  |   schema: 'public',
sdevproject-6-web-1  |   table: 'users',
sdevproject-6-web-1  |   column: undefined,
sdevproject-6-web-1  |   dataType: undefined,
sdevproject-6-web-1  |   constraint: 'users_username_key',
sdevproject-6-web-1  |   line: '663',
sdevproject-6-web-1  |   routine: '_bt_check_unique',
sdevproject-6-web-1  |   query: "INSERT into users (username, password) values ('DuplicateUser', '$2b$10$jUmBqzmeJM3nMicSv2hrkugEXoGQzjnJaC/eE.Vsfb0A6KEMmq9we') returning *;",
sdevproject-6-web-1  |   params: undefined
sdevproject-6-web-1  | }
sdevproject-6-web-1  |     ✓ negative: /register - Duplicate username (107ms)
sdevproject-6-web-1  |
sdevproject-6-web-1  |
sdevproject-6-web-1  |   5 passing (753ms)
```

This part is shown to demonstrate that we met all the requirements for the Lab 11 test cases and this was pushed before the due date. It is pulled from the November 14th commit to show no requirements for the lab were implemented afterwards.

UAT 1(Locating Proper Service):
The User Service Selection feature, integral to our application, is designed for users to choose services based on their needs. To test this functionality, we've devised comprehensive user acceptance test cases. We'll verify button functionality for accurate service selection, conduct a navigation test to ensure users are redirected to the correct service page, and assess service presentation on the selected page. The test data to test this feature includes status information, outputting messages and json formatted data that describes the results of each interaction with the feature. These tests will be executed in the development environment to simulate user interactions without affecting the live production environment. The expected outcome is a seamless user experience where users are directed to the chosen service page, and relevant services are displayed. Test results will be documented, and user acceptance testers, representing our diverse audience, will play a vital role in validating the feature's alignment with user expectations. This thorough test plan aims to ensure the robust functionality of the User Service Selection feature by the fourth week of the project.

Observations:
We observed that users were able to select the service that they desired from the business page that they selected. The users found the flow of our website very clear and easy to use which followed our desired output. Something a user pointed out was how getting to order a service feature requires them to go to the businesses first and this led us to reconsidering how we should present the product. As a result we made the website a businesses oriented application over a service oriented one. Another thing that we observed was that users could not go back to a business page from the service page which led us to add a back button on the service page.

UAT 2 (Register):
For the user registration feature, our test plan involves validating user inputs and ensuring password adherence to specific requirements. Specific test cases include confirming that users input both a username and password, and that the password meets specified criteria. Test data will include sample username-password pairs and passwords that intentionally violate or comply with requirements. The testing environment will be the development environment to replicate user interactions without affecting the live system. Successful test results entail accurate database insertion of user information, triggering a redirection to the login page. In this case, the user acceptance testers will be the development team responsible for the feature, ensuring that the registration process aligns with project expectations and user needs. This well-structured plan aims to streamline testing processes by the fourth week of the project.

Observations:
We observed that users were able to register with their own accounts. They were met with our desired error messages if our registration conditions were not met. Originally this was just the json error message displayed on the page for testing purposes but as we added the finishing touches to the project, we made sure there was no random error message generated and instead provided a professional error message dialogue. One thing we noticed was users were not able to find the login page if they already had an account, so we added a button to redirect

users to the login page. We also made sure that there was no way to accidentally register an account while already logged in as that would create complications for specific hashed keys. It was clearly apparent to the user which part of the register process they were in, whether it would be having a duplicate user, invalid password, or successful login into the system. We knew that they were going to get into the businesses one way or another

UAT 3(User Profile):
To comprehensively test the user profile feature, we will focus on ensuring that user-associated orders and purchases accurately reflect on their profile. Specific test cases involve confirming that orders linked to the user and their corresponding purchases are correctly displayed. Test data will include simulated order and purchase records in the development environment, reflecting various user activities. The development environment is chosen for testing to replicate real user interactions without impacting the live system. Successful test results will confirm that the user profile accurately reflects the user's actions and entered information, showcasing a complete and coherent overview of their orders and purchases. The user acceptance testers for this feature will include members of the quality assurance team, who will meticulously assess the alignment of the user profile functionality with project specifications. This detailed test plan is designed to expedite the testing process and ensure the robust functionality of the user profile feature by the fourth week of the project.

Observations:
We observed that users were able to correctly navigate to the profile page and see the information related to their account. They were able to see the services that they had ordered. We noticed that they wanted to see more information in regards to their order so we added a little drop down to each service order to provide them with more details. We wanted to make sure that the user was able to clearly see and access all of the necessary information that they have accumulated using the website without having to search long. We tested to see if each button functioned properly for adding information to the profile and dynamically allocating new parts based on order statuses or personal details that may be displayed uniquely to that specific user.

Deployment:

Prerequisites:

Before you begin, ensure you have met the following requirements:

- Node.js: The application requires Node.js to run JavaScript code on the server. Download and install Node.js from (https://nodejs.org/).
- npm (Node Package Manager): npm is used to install dependencies for the project. It comes bundled with Node.js, so installing Node.js should also install npm.
- PostgreSQL: Our application uses PostgreSQL as its database. You can download and install it from (https://www.postgresql.org/download/).
- Git: You will need Git for version control and to clone the repository. You can download it from (https://git-scm.com/downloads).
- Docker: For containerization and to ensure a consistent environment across different setups. Install Docker from (https://www.docker.com/get-started).

Please install all the above prerequisites before proceeding with the installation of the application.

Running the Application Locally

To run this application locally, follow these steps:

1. Clone the Repository

First, clone the repository to your local machine using Git:

```
git clone https://github.com/osca3666/SDevProject-6.git
cd SDev Project-6
```

2. Start the Application

Run the following command in the root directory of your project:

```
docker-compose up
```

This command will build and start all the services defined in your docker-compose.yml file. It may take a few minutes the first time as Docker needs to download the base images and build the containers. If you wish to run docker with any flags, please check https://docs.docker.com/engine/reference/commandline/compose_up/ for how to do this.

3. Access the Application

Once the containers are running, you can access the application in your web browser. By default, it will be available at http://localhost:3000 (unless you have specified another port in your Docker configuration).

4. Stopping the Application
To stop the application, use:

```
docker-compose down
```

This command will stop and remove the containers, networks, and volumes created by `docker-compose up`. For composing down with flags, please check https://docs.docker.com/engine/reference/commandline/compose_down/ for more information