DALHOUSIE UNIVERSITY

CSCI5709 - Advanced Web Services
Summer 2024

Assignment 2

Submitted By:
Chinmaya Garg
B00925398

Table of Contents

About the Project: ***2***

Feature: ***2***

Target User Insight: ***2***

User Persona 1: **3**

User Persona 2: **4**

User Persona 3: **5**

Assumptions on Why Users Would Use This Application **5**

Scenarios and Use Cases **6**

Sarah Thompson, the Graphic Designer: 6

Mark Rodriguez, the Small Business Owner: 6

Emma Johnson, the Marketing Manager: 6

Requirements or Prerequisites for Users **6**

User Experience: ***7***

Shipping Details Form **7**

Task 1: Entering Shipping Information 7

Task 2: Address Validation 10

Task 3: Is Billing Address Same as Shipping Address Validation 12

User-Centered Design: ***14***

EcoMart Sitemap **14**

EcoMart Shipping Address Page Wireframe **15**

Information Architecture for EcoMart Shipping Address Page **15**

Contact Details Section: 16

Shipping Details Section: 16

Checkbox: 16

Action Button: 16

Order Summary Section: 16

Application Architecture: ***16***

Interaction Design: ***18***

Process Workflow: ***18***

Overall Workflow: **18**

Feature Specific Workflow: **20**

Folder Structure: ***22***

Frontend Codebase Folder Structure: **22**

Explanation of Key Directories and Files: 22

Backend Codebase Folder Structure: **24**

Explanation of Key Directories and Files: 24

References: ***25***

About the Project:

In today's fast-paced digital world, the need for effective online marketplaces is crucial. This project aims to develop a cutting-edge, user-centric e-commerce platform that addresses current gaps in the market. Unlike existing platforms, Ecomart will offer integrated delivery services with thorough quality checks, enhancing trust and satisfaction. Our goals include designing a user-friendly interface, implementing secure payment gateways, and promoting eco-friendly practices. We will provide tools for small businesses to manage inventory efficiently and expand their market reach. Additionally, Ecomart will foster professional networking and collaboration, offering resources like webinars and mentorship programs. By achieving these objectives, we aim to create a leading online marketplace that supports a sustainable and connected digital economy. [1].

Feature:

The shipping address feature is crucial for ensuring efficient deliveries on the Ecomart platform. It allows users to save multiple addresses, making it convenient for purchasing items for different locations, such as home and office. The feature includes tasks such as entering shipping information through a detailed form, automatic address validation to minimize errors, and an option to confirm if the billing address is the same as the shipping address. This user-friendly feature integrates seamlessly with our end-to-end delivery system, providing real-time tracking and updates. By reducing friction during the checkout process and ensuring accurate deliveries, the shipping address feature enhances the overall user experience. It supports our goal of providing a reliable and trustworthy marketplace, contributing to higher user satisfaction and loyalty. [2]

Target User Insight:

Our target users are diverse yet share common needs and values that our platform addresses effectively. They include busy professionals like Sarah Thompson, who seek a convenient and efficient way to buy and sell high-quality used items without compromising their hectic schedules. Environmentally conscious individuals like Emma Johnson are drawn to the platform's emphasis on sustainability and eco-friendly practices, appreciating features that support the sale and purchase of innovative and green tech products. Small business owners like Mark Rodriguez value the practical tools and reliable logistics solutions that help them expand their market reach and manage inventory seamlessly. Across these user personas, there is a strong demand for trust and security, which our platform meets through integrated delivery services, thorough quality checks, and secure payment gateways. By providing a user-friendly interface, excellent customer support, and resources for professional networking and collaboration, our platform caters to the varied needs of our users, ensuring a seamless and satisfying experience for all. [1]:

User Persona 1:

Sarah – Graphics Designer

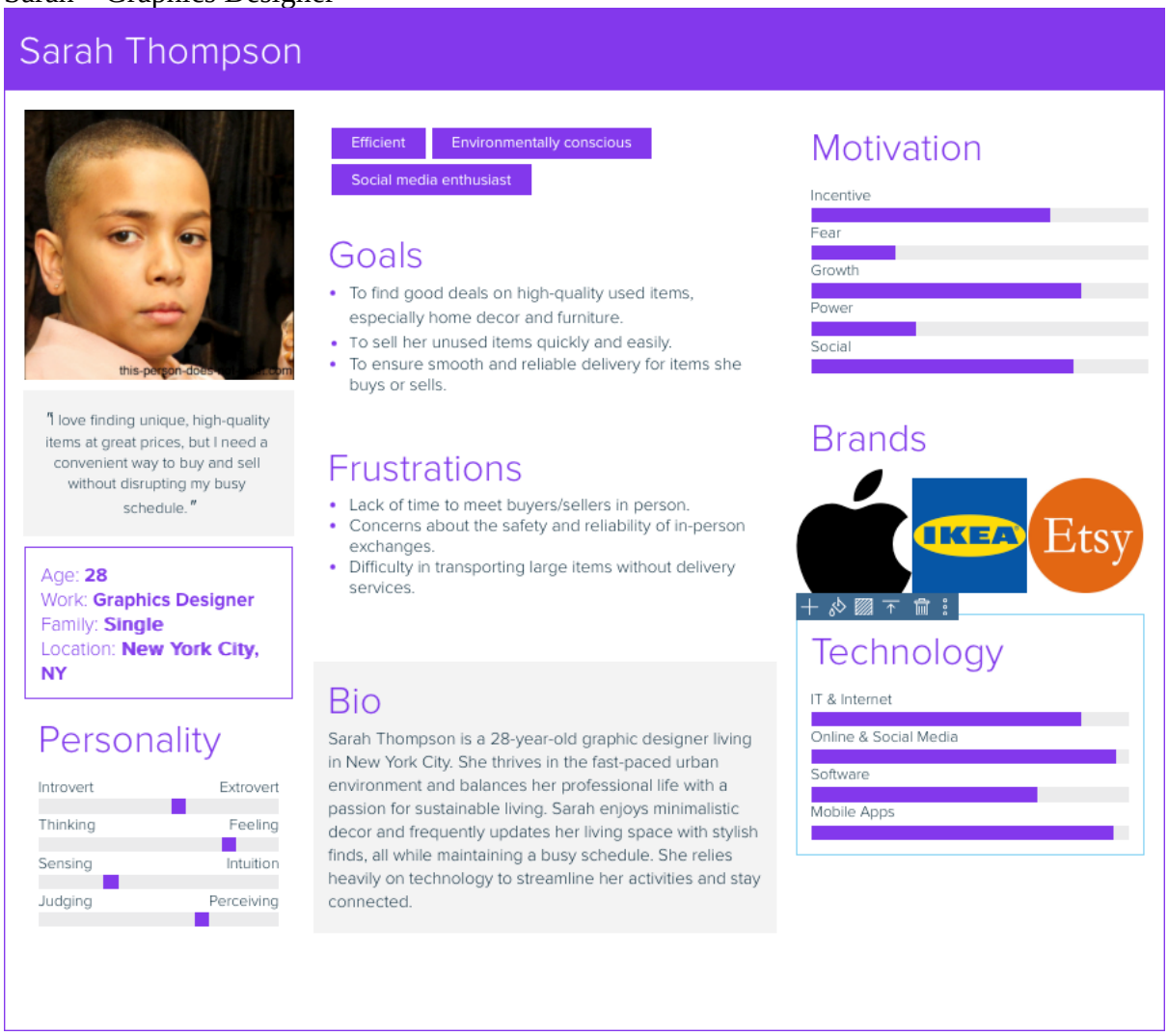

Figure 1. User Persona 1, created using Xtensio [4]

Sarah Thompson is a 28-year-old graphic designer living in New York City. She thrives in a fast-paced urban environment and balances her professional life with a passion for sustainable living. As a tech-savvy and eco-conscious individual, Sarah seeks platforms that offer high-quality used items, particularly home decor and furniture, to maintain her minimalist lifestyle. She values convenience and reliability, requiring efficient delivery services and secure methods for transactions to fit her busy schedule. [1]

User Persona 2:
Mark Rodriguez, the Small Business Owner

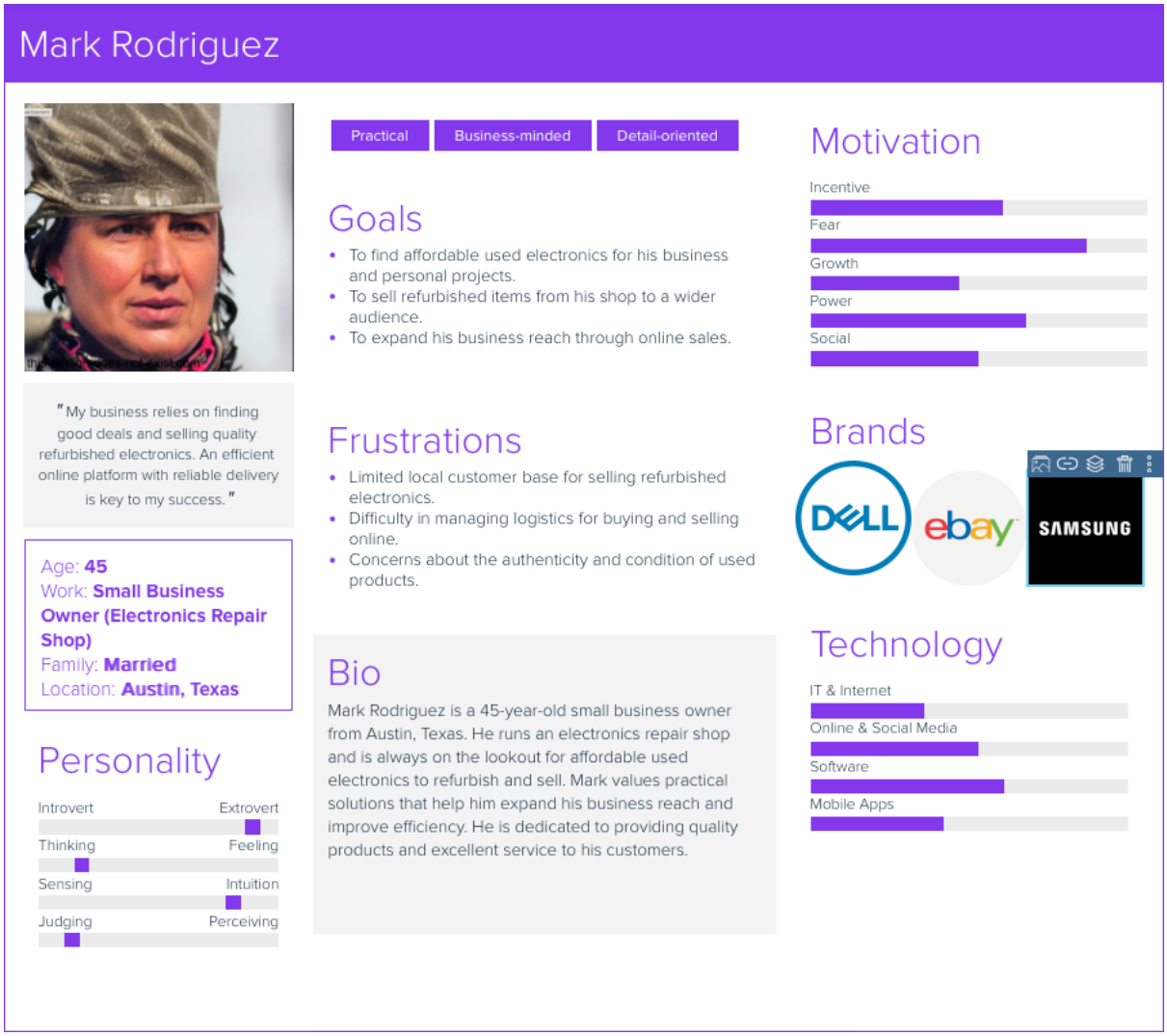

Figure 2. User Persona 2, created using Xtensio [4]

Mark Rodriguez is a 45-year-old small business owner from Austin, Texas, who runs an electronics repair shop. Constantly on the lookout for affordable used electronics to refurbish and sell, Mark values practical solutions that enhance his business efficiency. He needs a platform that provides a wide range of affordable products, reliable logistics for online transactions, and tools to reach a broader customer base. Mark's focus is on expanding his business reach and ensuring the authenticity and quality of the products he sells. [1]

User Persona 3:

Emma Johnson, the Marketing Manager

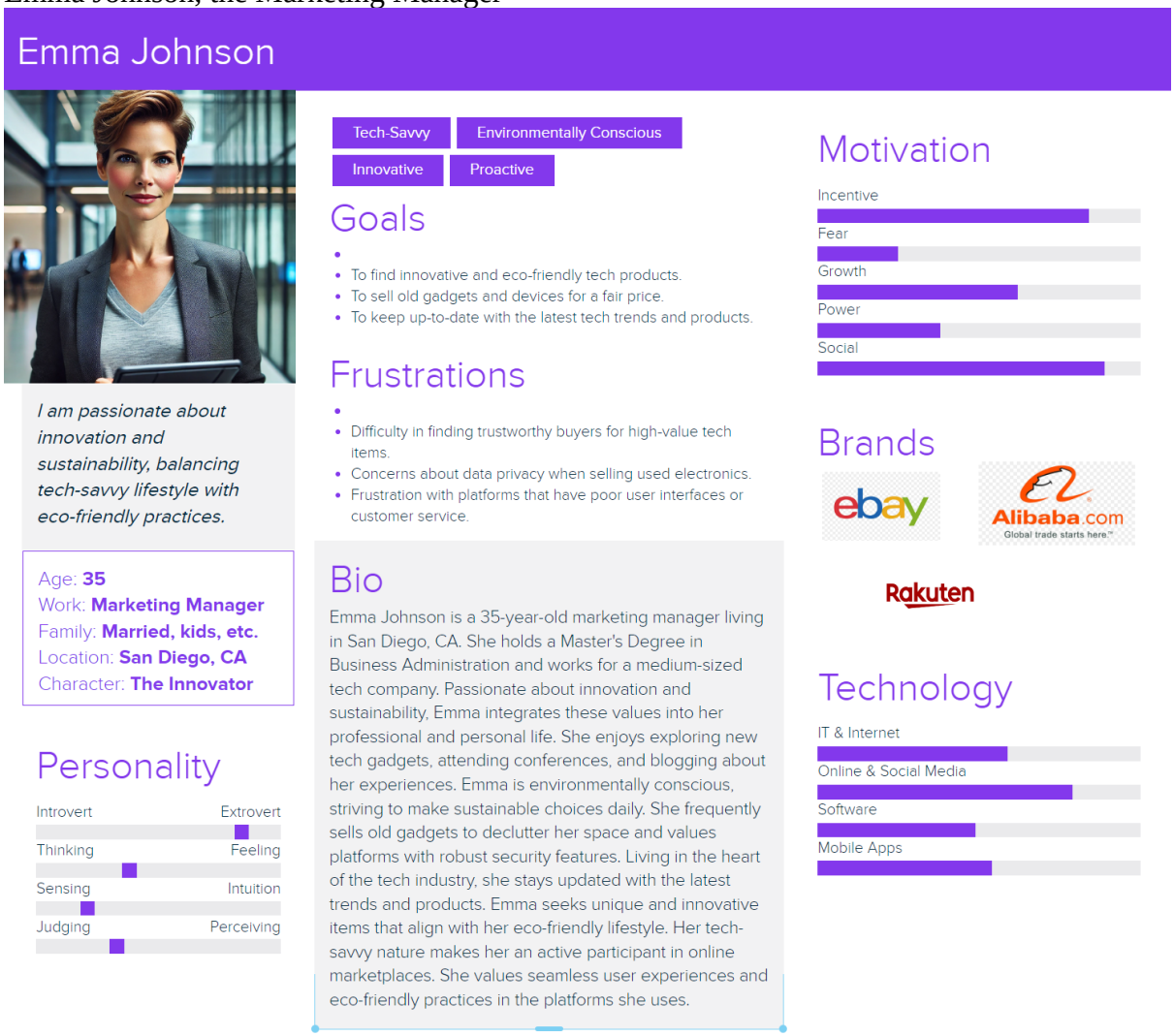

Figure 3. User Persona 3, created using Xtensio [4]

Emma Johnson is a 35-year-old marketing manager living in San Diego, CA, who balances her professional life with a passion for innovation and sustainability. She is always on the lookout for the latest eco-friendly tech products and values platforms that offer secure methods for selling used electronics. As a tech-savvy professional, Emma needs user-friendly interfaces and excellent customer support to facilitate her buying and selling activities. She appreciates platforms that align with her eco-friendly lifestyle and support her proactive approach to integrating technology and sustainability into everyday life. [1]

Assumptions on Why Users Would Use This Application

- Convenience and Efficiency:
  Users like Sarah Thompson, a busy graphic designer, value platforms that save time and provide a seamless buying and selling process. The integrated delivery services and user-friendly interface ensure a convenient experience for both buyers and sellers. [1]

- Trust and Security:

The thorough quality checks and secure payment gateways appeal to users like Mark Rodriguez, who prioritize the authenticity and condition of the products they purchase or sell. Users trust the platform to handle transactions securely and reliably. [1]

- Sustainability and Eco-Friendliness:
  Environmentally conscious users like Emma Johnson are drawn to the platform's promotion of eco-friendly and sustainable products. The application's focus on supporting innovative business models such as upcycling and refurbishment aligns with their values. [1]

- Comprehensive Delivery Services:
  Users appreciate the end-to-end delivery solutions that handle logistics from seller to buyer. The real-time tracking and updates provide peace of mind and enhance the overall user experience. [1]

- Wide Range of High-Quality Used Items:
  The platform offers a diverse selection of high-quality used items, catering to various needs and preferences. This attracts users looking for good deals on items such as electronics, home decor, and tech gadgets. [1]

Scenarios and Use Cases

Sarah Thompson, the Graphic Designer:

Scenario: Sarah is redecorating her apartment and is looking for high-quality, affordable used furniture and home decor.
Use Case: She uses the platform's advanced search and filtering options to find specific items, selects them, and utilizes the integrated delivery service to ensure they arrive safely at her doorstep.

Mark Rodriguez, the Small Business Owner:

Scenario: Mark needs a steady supply of used electronics to refurbish and sell in his repair shop.
Use Case: He browses the platform for bulk purchases of used electronics, verifies their condition through the platform's quality checks, and uses the delivery service to manage logistics efficiently, ensuring his shop's inventory is always stocked.

Emma Johnson, the Marketing Manager:

Scenario: Emma wants to sell her old tech gadgets to make space for new, eco-friendly devices while ensuring data privacy.
Use Case: She lists her used gadgets on the platform, uses the secure methods provided to erase data, and relies on the platform's quality assurance and customer support to handle transactions smoothly, ensuring a seamless selling experience.

Requirements or Prerequisites for Users

Device Requirements: Users must possess an internet-enabled device, such as a smartphone, tablet, or computer.
Account Registration: Users are required to create an account in order to purchase or sell items on the platform.

User Experience:

Shipping Details Form

Task 1: Entering Shipping Information

Scenario: Sarah Thompson, a busy graphic designer in New York City, needs to quickly and accurately input her shipping details when purchasing items from an e-commerce platform. She is visiting the checkout page to enter her contact and shipping details.

Application: EcoMart

User Persona: Sarah Thompson

Goal: To enter her shipping details efficiently and accurately.

Task: Enter and validate shipping details.

Use Case: Entering Shipping Information

1. User navigates to the checkout page. [user action]
    i. System displays the checkout page with a form to enter shipping details. [system action]
2. User enters their full name in the shipping details form. [user action]
    i. System validates the full name field. [system action]
    ii. System displays an error message below the full name input field if validation fails. [system action]
3. User enters their street address. [user action]
    i. System validates the street address field. [system action]
    ii. System displays an error message below the street address input field if validation fails. [system action]
4. User enters their apartment/suite/unit (optional). [user action]
    i. System validates the apartment/suite/unit field. [system action]
    ii. System displays an error message below the apartment/suite/unit input field if validation fails. [system action]
5. User enters their city. [user action]
    i. System validates the city field. [system action]
    ii. System displays an error message below the city input field if validation fails. [system action]
6. User selects their state/province/region from a dropdown. [user action]
    i. System validates the state/province/region selection. [system action]
    ii. System displays an error message below the dropdown if validation fails. [system action]
7. User enters their postal/ZIP code. [user action]
    i. System validates the postal/ZIP code field. [system action]
    ii. System displays an error message below the postal/ZIP code input field if validation fails. [system action]
8. User selects their country from a dropdown. [user action]
    i. System validates the country selection. [system action]
    ii. System displays an error message below the dropdown if validation fails. [system action]

9. User enters their phone number. [user action]
    i. System validates the phone number field. [system action]
    ii. System displays an error message below the phone number input field if validation fails. [system action]
10. User submits the form. [user action]
    i. System performs server-side validation on the submitted data. [system action]
    ii. System provides suggestions or corrections if the address validation fails. [system action]
    iii. User corrects any errors and resubmits the form. [user action]
11. System stores the validated shipping information. [system action]
12. User proceeds to the next step. [user action]
    i. System displays the next step in the checkout process. [system action]

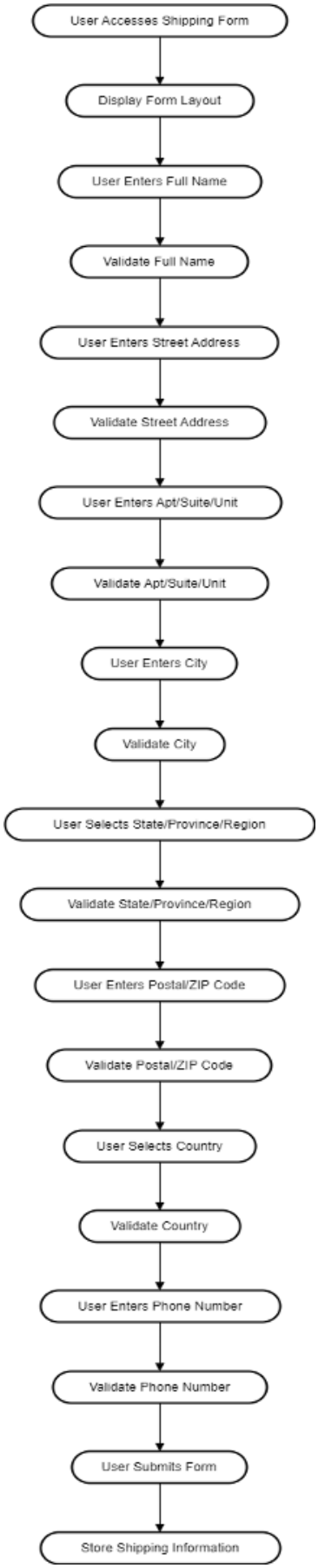

Figure 4: Task flow diagram to enter shipping information [3]

Task 2: Address Validation

Scenario: Sarah Thompson, after entering her shipping details, needs to ensure that the address is valid. The system must validate the address to avoid any delivery issues.

Application: EcoMart

User Persona: Sarah Thompson

Goal: To validate her shipping address to ensure successful delivery.

Task: Validate the entered shipping address.

Use Case: Address Validation

1. User submits the shipping details form. [user action]
    i. System performs client-side validation on the entered data. [system action]
2. Client-side validation checks:
    i. System checks for required fields, postal/ZIP code format, and phone number format. [system action]
    ii. System displays error messages below respective input fields if validation fails. [system action]
3. User corrects any errors and resubmits the form. [user action]
    i. System sends the data to the server for server-side validation. [system action]
4. Server-side validation:
    i. System uses a third-party API to validate the address. [system action]
    ii. If the address is invalid, the system provides suggestions or corrections. [system action]
5. User reviews and accepts suggestions (if any). [user action]
    i. System updates the form with corrected address (if applicable). [system action]
6. User resubmits the corrected form (if applicable). [user action]
    i. System stores the validated address information. [system action]
7. System confirms successful validation and displays the next step in the checkout process. [system action]

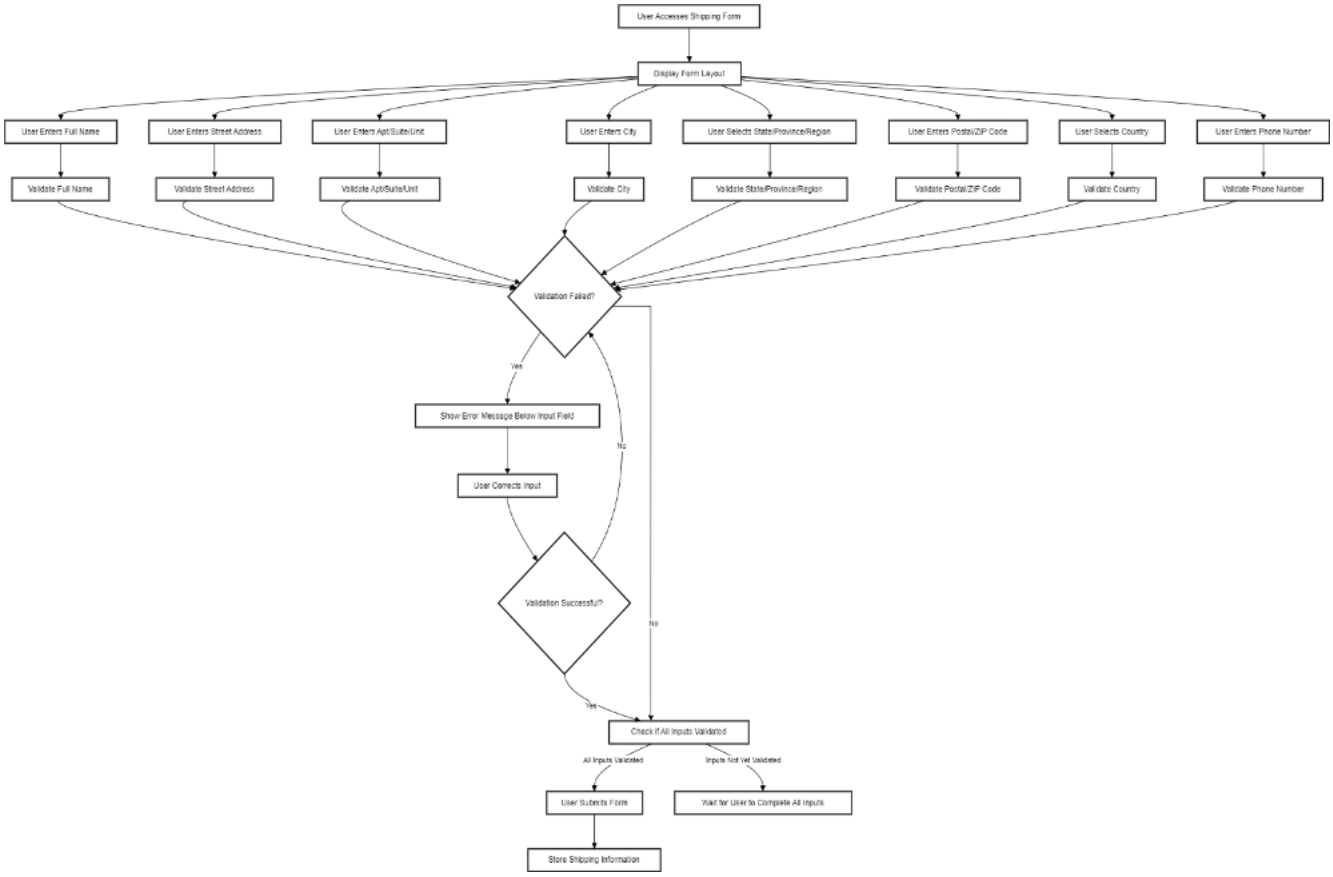

Figure 5: Task flow diagram for *form* validation [3]

Task 3: Is Billing Address Same as Shipping Address Validation

Scenario: Sarah Thompson needs to decide if her billing address is the same as her shipping address during the checkout process. If not, she will need to enter her billing address separately.

Application: EcoMart

User Persona: Sarah Thompson

Goal: To confirm if the billing address is the same as the shipping address or to enter a separate billing address.

Task: Validate if the billing address is the same as the shipping address or enter a separate billing address.

Use Case: Is Billing Address Same as Shipping Address Validation

1. User completes the shipping address form. [user action]
    i. System displays a checkbox asking if the billing address is the same as the shipping address. [system action]
2. User selects the checkbox if the billing address is the same. [user action]
    i. System auto-fills the billing address with the shipping address and hides the billing address section. [system action]
3. If the user does not select the checkbox, the system displays the billing address form. [system action]
4. User enters billing address details. [user action]
    i. System validates each field in the billing address form similar to the shipping address form. [system action]
    ii. System displays error messages below respective input fields if validation fails. [system action]
5. User corrects any errors and resubmits the billing address form. [user action]
    i. System performs server-side validation on the submitted billing address data. [system action]
    ii. System provides suggestions or corrections if the address validation fails. [system action]
    iii. User reviews and accepts suggestions (if any). [user action]
    iv. User resubmits the corrected billing address form (if applicable). [user action]
6. System stores the validated billing address information. [system action]
7. System confirms successful validation and displays the payment page. [system action]
8. User proceeds to the payment page. [user action]
9. System displays the payment options. [system action]

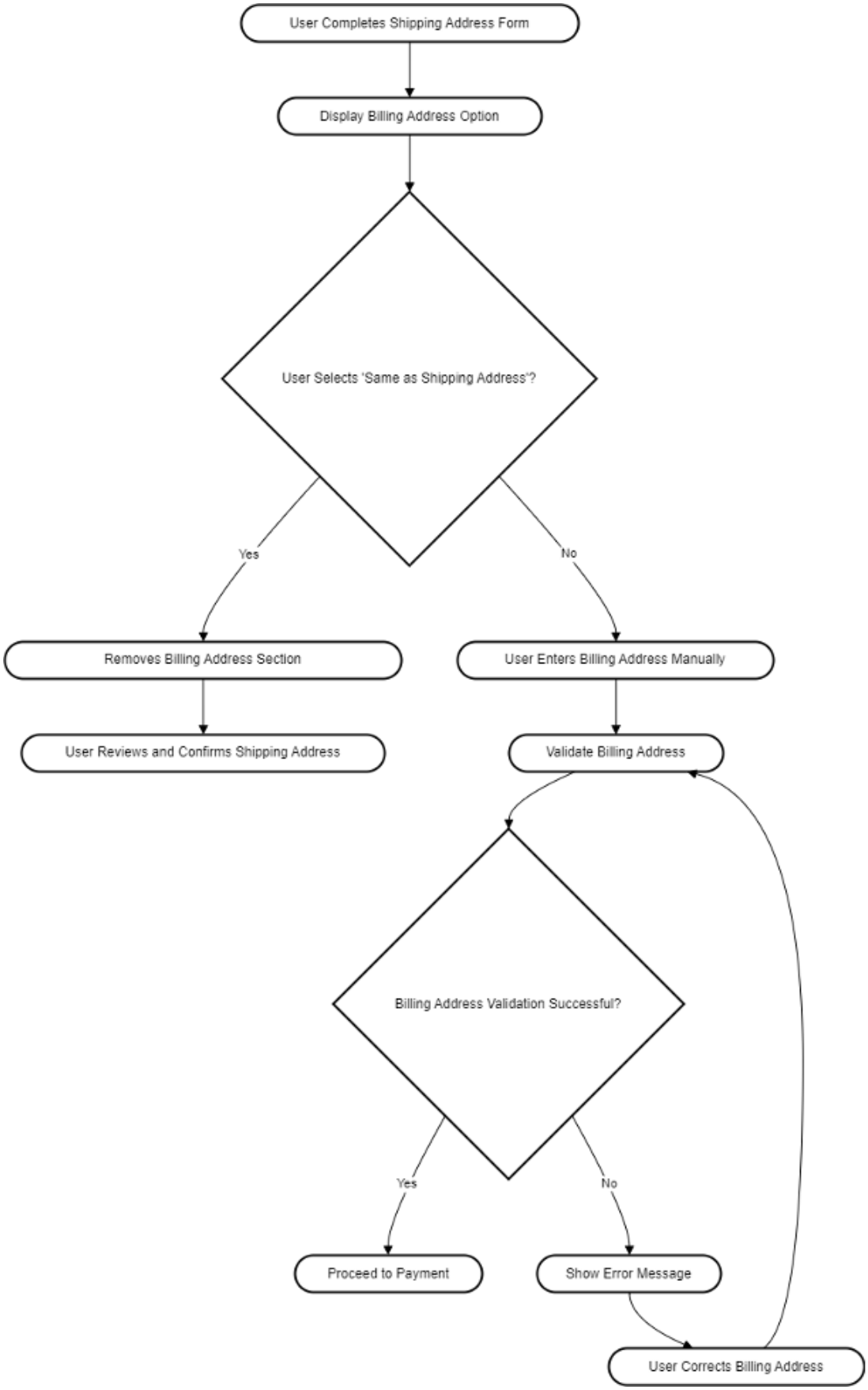

Figure 6: Task flow diagram for billing address same as shipping address. [3]

User-Centered Design:

The design and development of EcoMart, especially the Shipping Address feature, were guided by insights gathered from target users. A sitemap was created to organize the platform's structure and ensure a clear and intuitive user experience. Wireframes were also developed to visualize key pages, including the shipping address management page, and to optimize user flow and element placement such as address fields, validation messages, and action buttons. These user insights and design elements helped shape EcoMart into a seamless and user-friendly platform.

EcoMart Sitemap

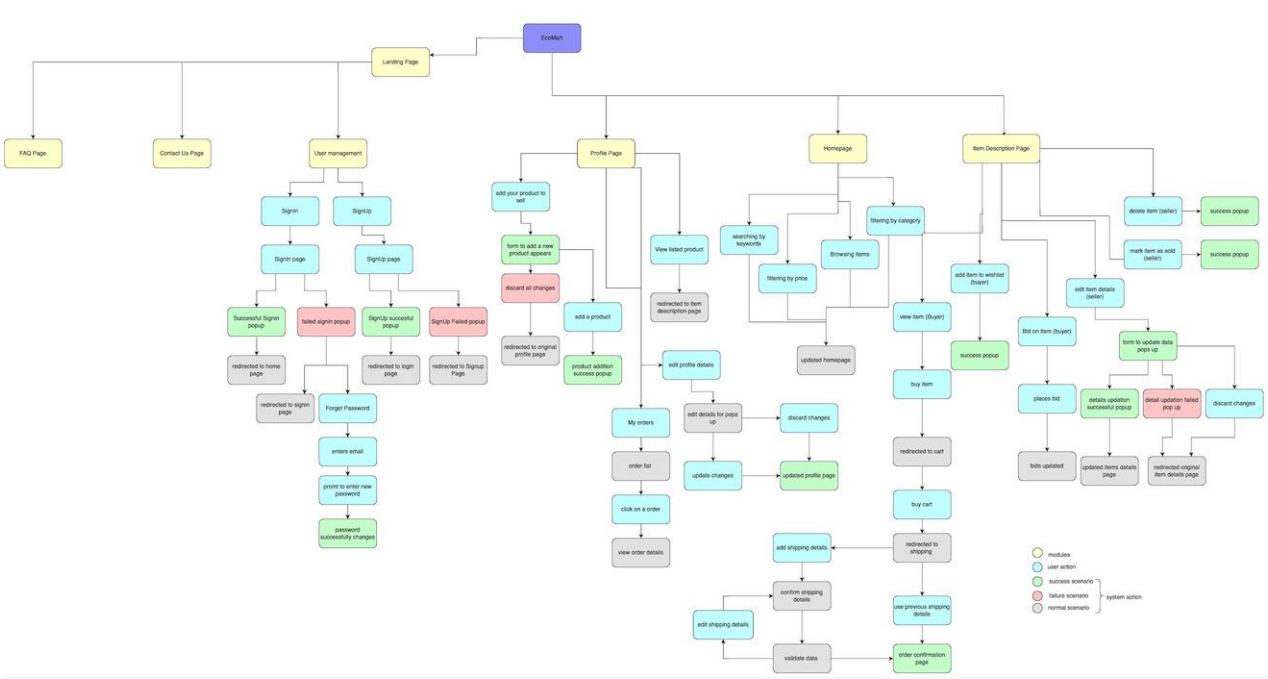

Figure 7: Proposed sitemap for EcoMart. created with draw.io [1][3]

EcoMart Shipping Address Page Wireframe

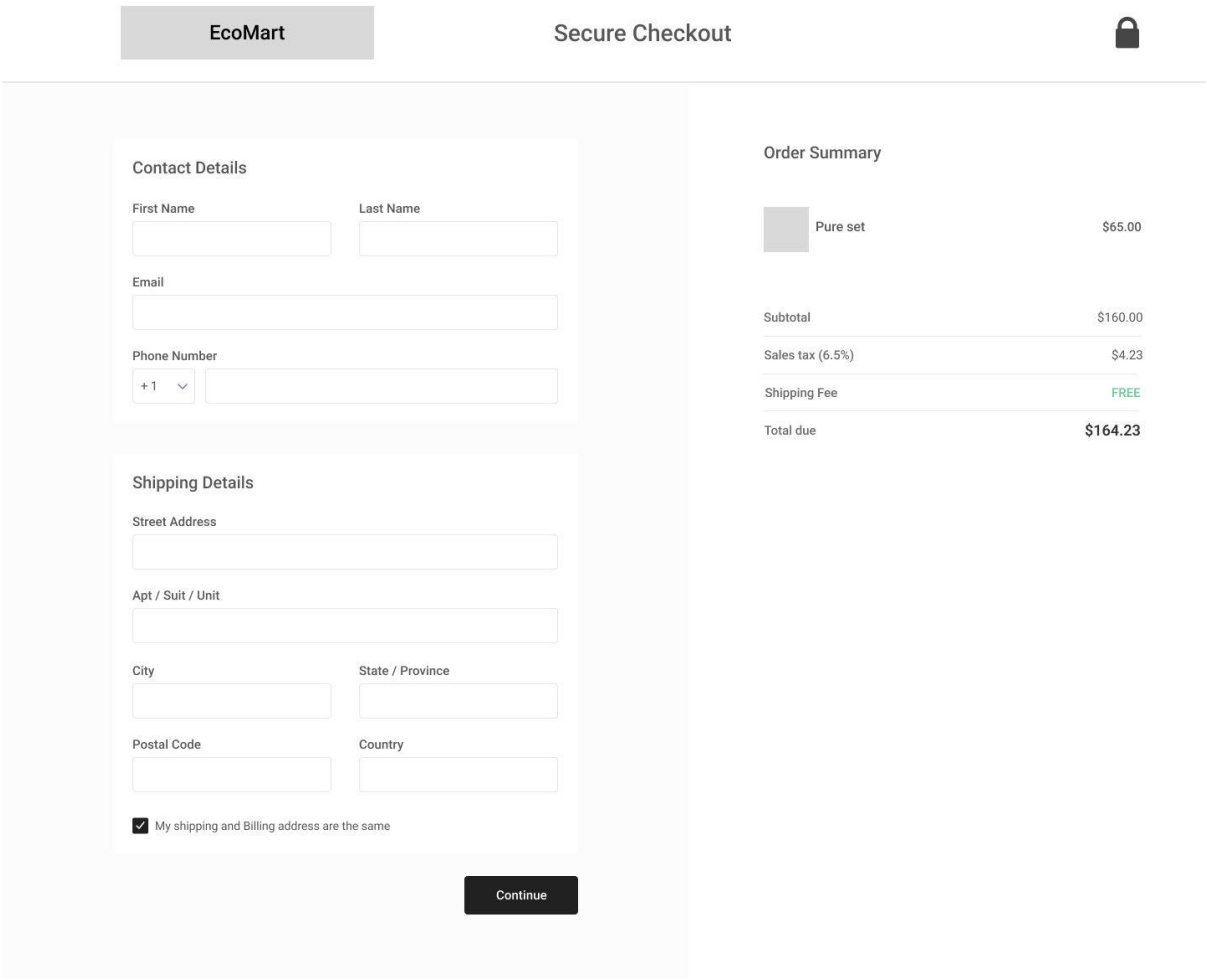

Figure 8: Wireframe of Shipping Details Page. Created with Figma [2][6].

Following the finalization of the sitemap and wireframes, the actual webpage design will be implemented. The shipping address page features a clean layout where users can easily add, edit, and delete addresses. The page prominently includes a form for entering new shipping information, address validation notifications, and options to set a billing address as the same as the shipping address. This design ensures that managing shipping addresses is straightforward and efficient.

This design approach, which includes sitemapping, wireframing, and webpage design, ensures that EcoMart meets the unique needs of its target users. The visually appealing, intuitive, and user-friendly platform enhances the shopping experience by providing seamless address management functionalities.

Information Architecture for EcoMart Shipping Address Page

The shipping address functionality on the EcoMart checkout page allows users to efficiently enter and manage their shipping information, ensuring a seamless checkout experience. Below is the structured information architecture for this feature:

Contact Details Section:

- **First Name**: Input field for the user's first name.
- **Last Name**: Input field for the user's last name.
- **Email**: Input field for the user's email address.
- **Phone Number**: Input field for the user's phone number with a country code selector.

Shipping Details Section:

- **Street Address**: Input field for the street address.
- **Apt / Suite / Unit**: Input field for additional address information such as apartment, suite, or unit number.
- **City**: Input field for the city.
- **State / Province**: Input field for the state or province.
- **Postal Code**: Input field for the postal code.
- **Country**: Input field for the country.

Checkbox:

- **My shipping and billing address are the same**: Checkbox to indicate if the shipping address is the same as the billing address.

Action Button:

- **Continue**: Button to proceed to the next step in the checkout process.

Order Summary Section:

- **Item Details**: Displays the description and price of the item(s) being purchased.
- **Subtotal**: Shows the subtotal amount for the items.
- **Sales Tax**: Displays the calculated sales tax.
- **Shipping Fee**: Displays the shipping fee, which may be free or a specified amount.
- **Total Due**: Displays the total amount due, including item prices, taxes, and any shipping fees.

This information architecture ensures that users can quickly and accurately provide their shipping information, facilitating a smooth and efficient checkout process on EcoMart

Application Architecture:

EcoMart's architecture is built using a tech stack comprising React for the frontend, and Python with Django and MySQL for the backend.

On the frontend, React is used to create a dynamic and responsive user interface. This component-based approach allows for the development of reusable UI elements, enabling a

smooth and interactive user experience. React efficiently handles state management and component rendering, ensuring the application remains fast and responsive.

On the backend, Python and Django form a robust server framework that facilitates the creation of RESTful APIs. Django's built-in features provide a powerful and scalable foundation, ensuring efficient handling of HTTP requests, clear separation of concerns, and scalability to support the application's growth. MySQL is used for the database, providing reliable data storage and efficient data retrieval.

This tech stack was chosen for its numerous benefits. React provides a rich ecosystem for building interactive UIs, while Python and Django offer a comprehensive framework for backend development. MySQL ensures robust and scalable data management. This combination enables consistent and efficient development across the application, supported by a vibrant community and extensive resources.

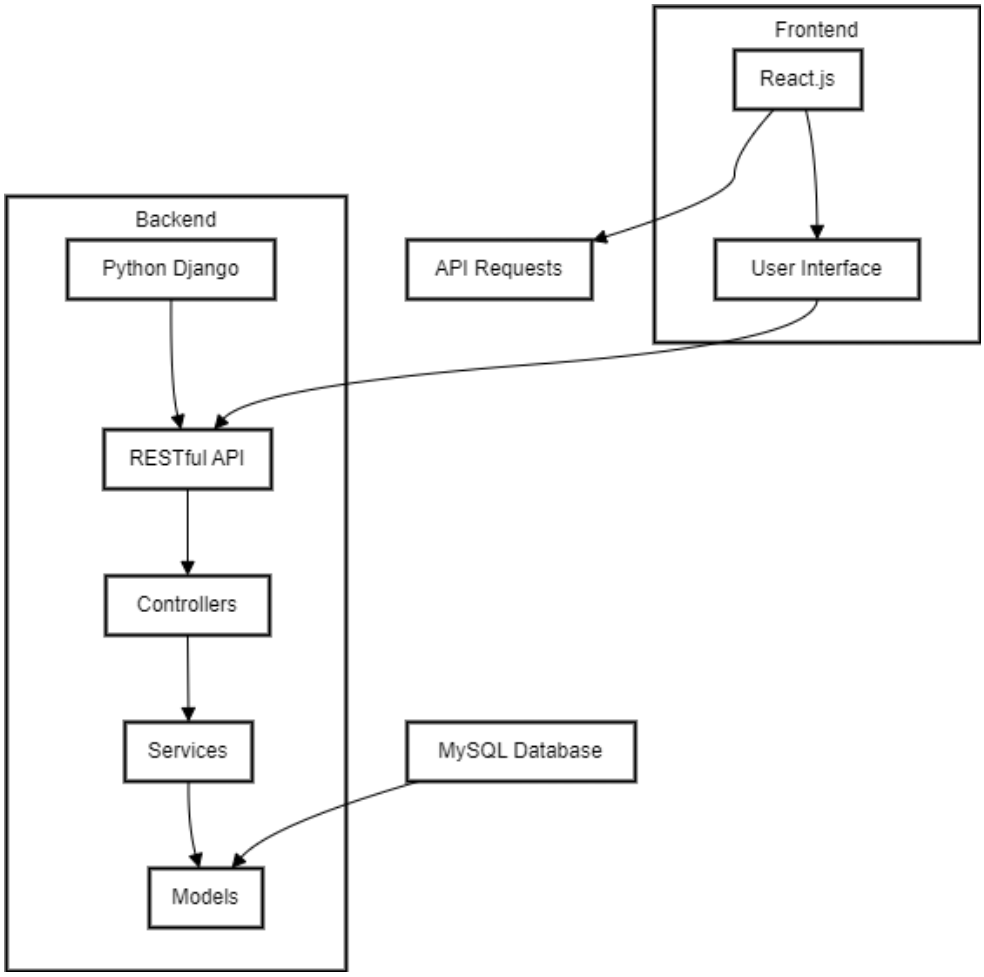

Figure 9: Application Architecture. Created with draw.io [3].

Interaction Design:

The click stream diagrams illustrate the user interactions involved in managing shipping addresses on EcoMart. These diagrams provide a visual representation of the sequence of user actions and their flow within the application's front-end.

In the click stream diagram for adding a new shipping address, the user starts from the account settings page, navigates to the shipping address management section, and fills out the address form. The system validates the address and, upon successful validation, saves the address to the user's profile.

Figure 10: Click Stream diagram of adding an address. Created with draw.io[3].

For the click stream diagram of editing an existing address, the user accesses the shipping address management section from their account settings, selects an address to edit, updates the necessary fields, and saves the changes. The system re-validates the updated address and confirms the changes.

Figure 11: Click Stream Diagram of editing an address. Created with draw.io[3].

These diagrams emphasize the importance of efficient address management functionality and provide a seamless flow from accessing account settings to managing shipping addresses.

Process Workflow:

Overall Workflow:

The overall application workflow for EcoMart, specifically focusing on the Shipping Address feature, demonstrates the seamless interaction between the frontend and backend components using React, Python, Django, and MySQL. Below are the detailed steps:

1. **User Interaction:**
    - The user interacts with the frontend UI, performing actions such as navigating to account settings, accessing the shipping address management section, and submitting forms to add or edit addresses.
2. **Frontend Request:**
    - The frontend sends requests to the backend API endpoints, including relevant data such as address details or user credentials.
3. **Backend Routing:**
    - The backend receives the requests and routes them to the appropriate API handlers based on the specified endpoints.
4. **Backend Processing:**
    - The backend processes the requests, which may involve validation, authentication, and data manipulation. For instance, validating the address format and ensuring the user is authenticated.
5. **Database Interaction:**

- If required, the backend interacts with the MySQL database to fetch or modify data based on the request. This includes operations like adding a new address, updating an existing address, or deleting an address.

6. **Business Logic:**
   - The backend applies business logic to the received data, such as checking for duplicate addresses, verifying data integrity, or implementing rules for setting a default address.
7. **Response Preparation:**
   - The backend prepares the response, including the necessary data, status codes, and any additional metadata. For example, a success message if the address is added successfully or an error message if validation fails.
8. **Response Sent:**
   - The backend sends the response back to the frontend, containing the requested data or indicating the status of the requested action.
9. **UI Update:**
   - The frontend receives the response and updates the user interface accordingly, displaying the retrieved data or providing appropriate feedback to the user. For instance, updating the address list or showing a confirmation message.
10. **User Feedback Loop:**
    - The user interacts with the updated UI, initiating further actions or continuing the workflow as necessary, such as adding another address or proceeding to checkout.

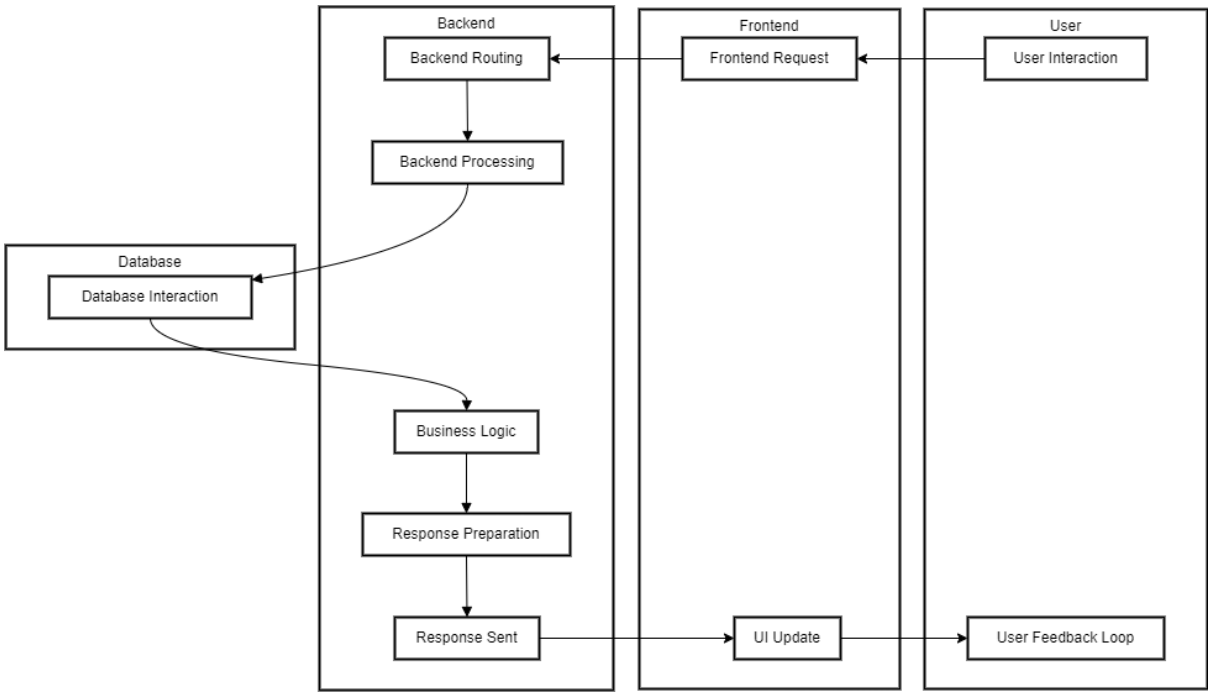

Figure 12: Process Workflow of Overall Application. Created with draw.io [3].

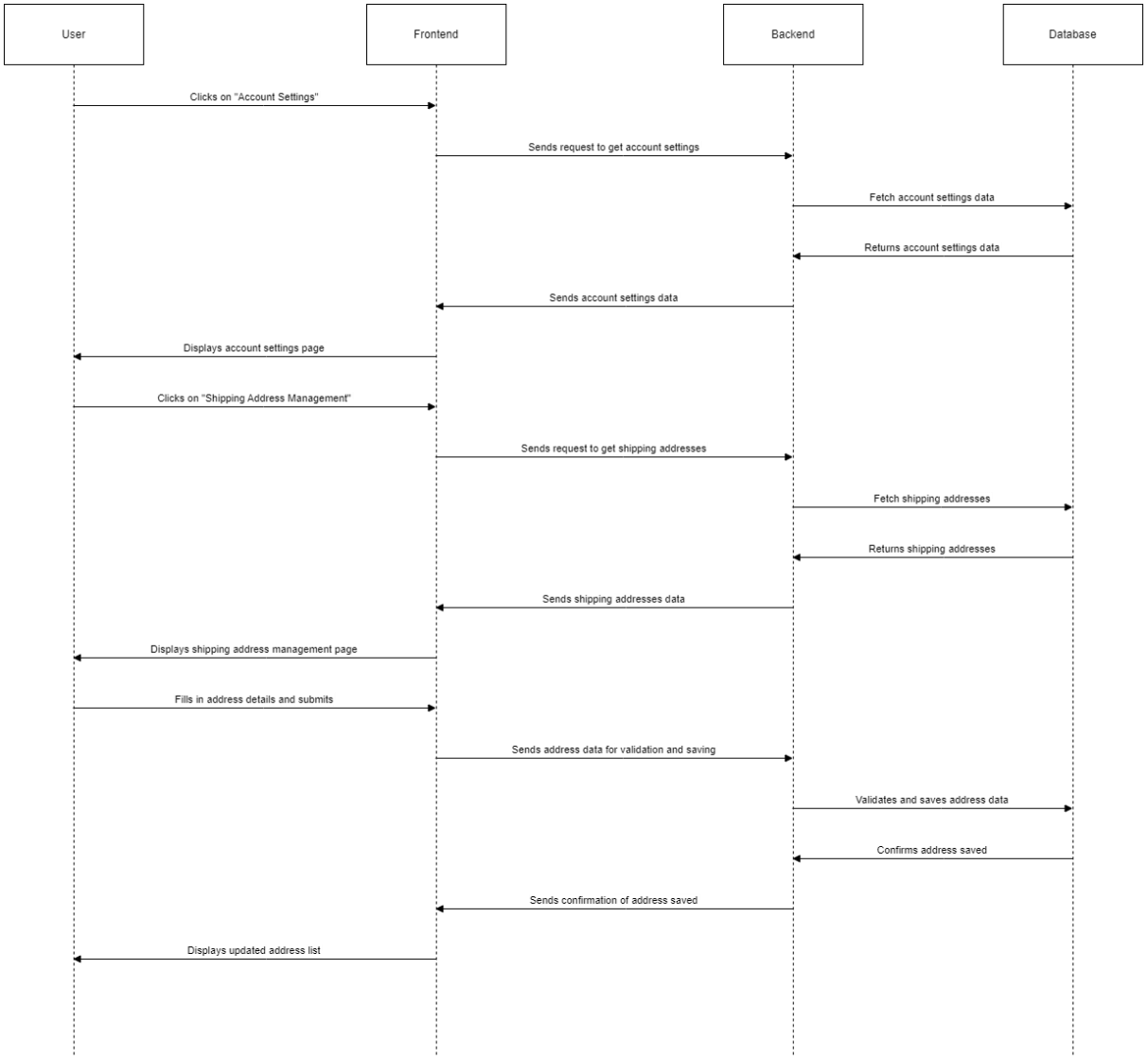

Figure 13: Process Workflow – Interaction Diagram of selected Feature. Created with draw.io [3].

Feature Specific Workflow:

The feature-specific workflow, as illustrated in Figure 11, outlines the process flow for managing shipping addresses on the EcoMart platform. Below are the steps for this workflow:

1. **User Interaction:**
    - The user interacts with the frontend UI, triggering actions such as adding, editing, or deleting a shipping address.
2. **Frontend Request:**
    - The frontend sends an API request to the backend, including parameters such as address details or action type (e.g., add, edit, delete).
3. **Backend Validation:**
    - The backend validates the received parameters to ensure they meet the required criteria (e.g., valid address format, authenticated user).
4. **Query Formation:**

- The backend forms a query using the validated parameters to perform the appropriate operation on the MySQL database (e.g., insert, update, delete).

5. **Database Query:**
   - The backend sends the query to the MySQL database for execution.
6. **Data Operation:**
   - The MySQL database executes the query and performs the required operation (e.g., adding a new address, updating an existing address, deleting an address).
7. **Response Preparation:**
   - The backend processes the result from the database, applying any necessary formatting or transformations.
8. **Response Sent:**
   - The backend sends the response back to the frontend, indicating the status of the operation (e.g., success, failure).
9. **UI Update:**
   - The frontend receives the response and updates the user interface to reflect the changes (e.g., updated address list, error messages).

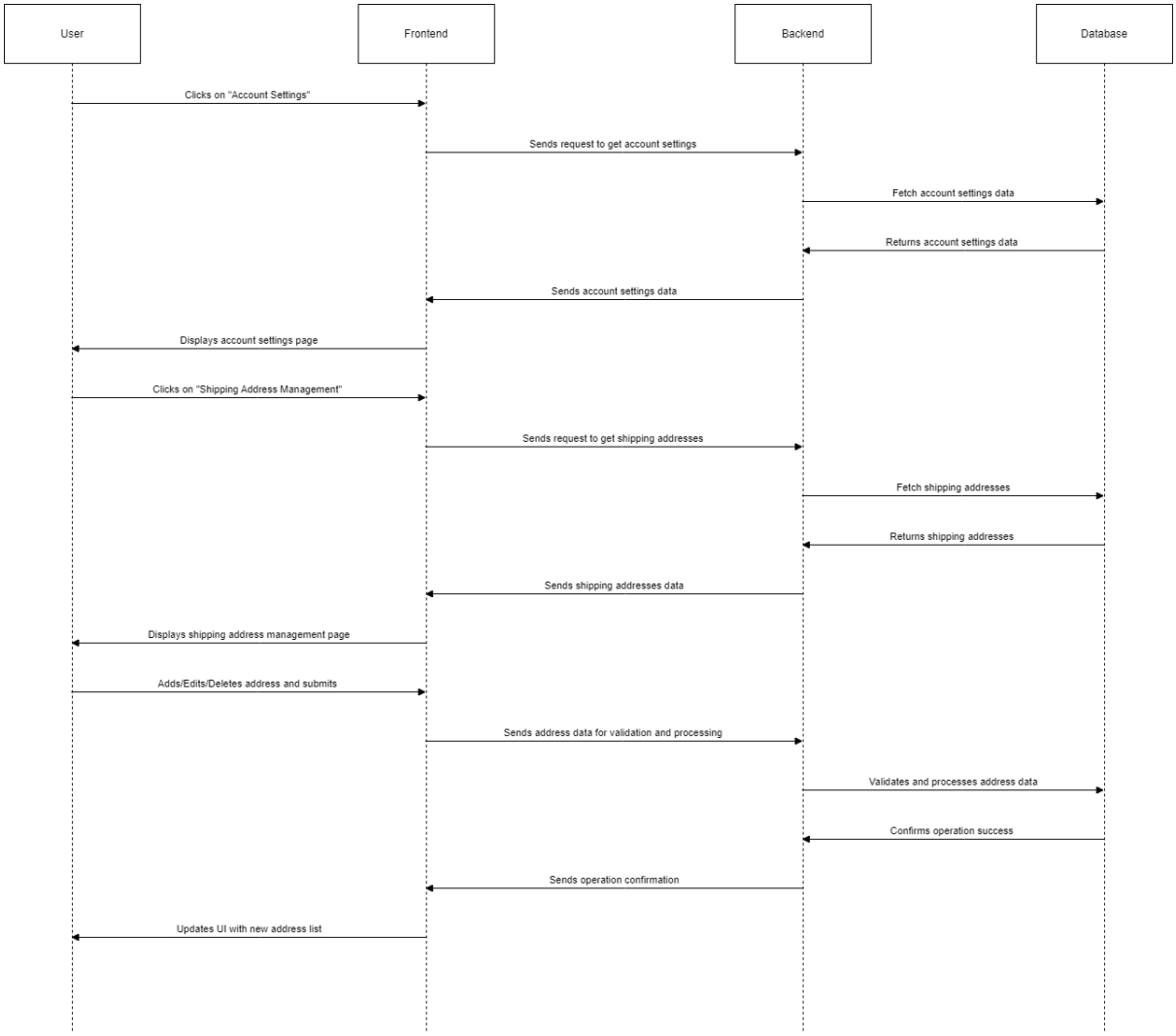

Figure 14: Process Workflow – Interaction Diagram of selected Feature. Created with draw.io [3].

Folder Structure:

Frontend Codebase Folder Structure:

The frontend folder structure for EcoMart is designed to ensure a clear and organized codebase, making it easier to maintain, scale, and develop new features. Here is a detailed explanation of each part of the folder structure:

Explanation of Key Directories and Files:

Public Folder

- **public/**: Contains static files that are directly served by the server.
    - **favicon.ico**: The favicon icon for the website, displayed in the browser tab.
    - **index.html**: The main HTML file that serves as the entry point for the React application. It contains the root `div` where the React app will be rendered.
    - **logo.png**: The logo image for the website, which can be used in various components like headers and footers.
    - **manifest.json**: The web app manifest for configuring the Progressive Web App (PWA) settings, such as the app's name, icons, and theme colors.

Source Folder

- **src/**: Contains all the source code for the React application.
    - **Assets/**: This folder is used to store static assets like images, fonts, and other media files that are used throughout the application.
    - **Components/**: Contains reusable React components that can be used across different parts of the application. This helps in maintaining a modular and organized codebase.
        - **LandingPage/**: A subfolder dedicated to components specific to the landing page of EcoMart. This includes various sections of the landing page, ensuring they are organized and easy to manage.
            - **FeaturesSection.js**: Component for the features section on the landing page, showcasing the key features of EcoMart.
            - **FooterSection.js**: Component for the footer section of the landing page, containing links, contact information, and other relevant details.
            - **HeroSection.js**: Component for the hero section of the landing page, usually the top section that includes a call-to-action and highlights the main value proposition.
            - **Landing.js**: The main component for the landing page that aggregates all other sections.
            - **LandingPageNav.js**: Component for the navigation bar on the landing page, providing links to different sections or pages.
            - **ContactUs.js**: Component for the contact us section, allowing users to get in touch with the company.
            - **FAQ.js**: Component for the frequently asked questions section, providing answers to common user queries.

        - **ShippingAddress/**: A subfolder that could be created for components specific to the shipping address functionality, ensuring all related components are grouped together for better maintainability.
            - **ShippingForm.js**: Component for the form where users can enter their shipping address details.
            - **AddressList.js**: Component for displaying a list of saved addresses.
            - **AddressItem.js**: Component for each individual address item, with options to edit or delete the address.
    - **index.css**: The main CSS file for global styles, applying base styles and themes to the entire application.
    - **index.js**: The main JavaScript file that bootstraps the React application and renders it into the DOM. This file typically includes the root component and any global providers, such as Redux or Context.
    - **reportWebVitals.js**: A file for measuring the performance of the React application. It provides a way to track the performance metrics and can be used to log results or send them to an analytics endpoint.

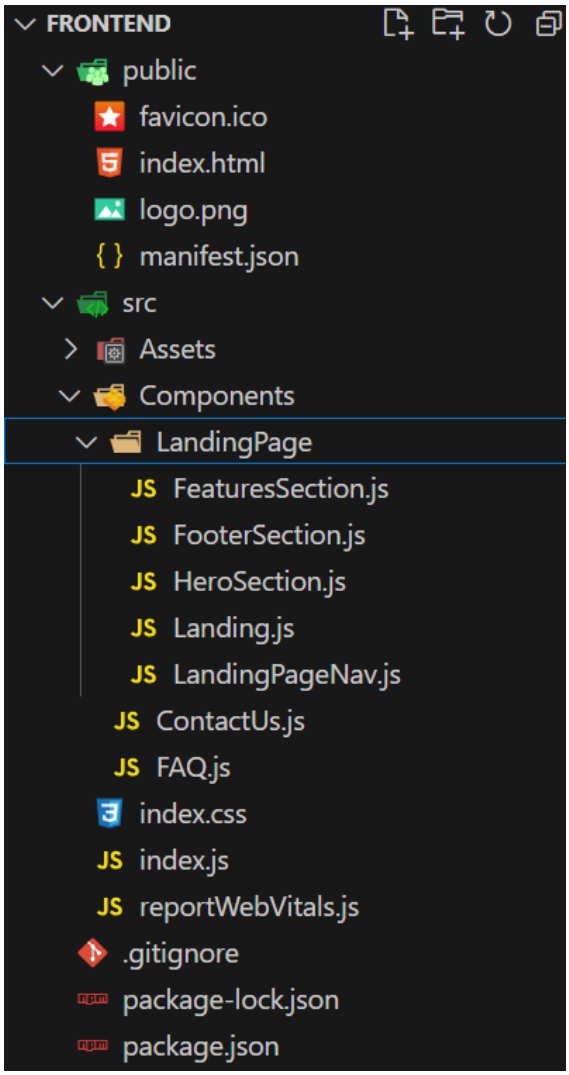

Figure 15: Frontend Folder Structure. Taken from [5]

Backend Codebase Folder Structure:

This folder structure is designed to keep the project organized and modular, making it easier to manage and extend the shipping details feature within the EcoMart backend.

Explanation of Key Directories and Files:

- **ecomart/**: This is the root directory of the Django project, containing project-wide settings, URLs, and configuration files.
- **shipping/**: This is the directory specific to the shipping details feature.
    - **static/**: Stores static files such as CSS, JavaScript, and images related to the shipping feature.
    - **views**: Contains views that handle HTTP requests related to shipping details.
    - **models**: Contains the database models for storing shipping details.
    - **urls**: Maps URLs to views specific to the shipping app.
    - **forms.py**: Defines forms for validating user input (if needed).
    - **serializers.py**: Defines serializers for converting complex data types to native Python datatypes for easy rendering into JSON or other content types.
    - **admin.py**: Configures the admin interface for shipping models.
    - **apps.py**: Configuration for the shipping app.
    - **permissions.py**: Custom permissions for the shipping app.
    - **services**: Contains business logic and services related to shipping.
    - **tasks.py**: Defines background tasks related to shipping.
- **requirements.txt**: Lists the Python dependencies required for the project.
- **manage.py**: Django's command-line utility for administrative tasks.
- **README.md**: Provides documentation and information about the project.

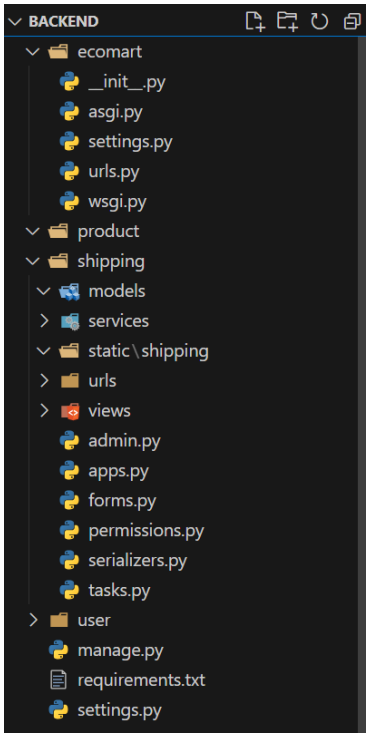

Figure 16: Backend Folder Structure. Taken from [5]

References:

[1] N. Khacharia, P. Kachhadiya, K. Patel, K. Parmar, C. Garg, and T. Patel, “D1_Group01 - Project Proposal." Dalhousie University, [online document], 2024. [Accessed July 9, 2024]

[2] C. Garg, “CSCI 5709 ASSIGNMENT 1.” Summer Term, Dalhousie University, [online document], 2024. [Accessed July 9, 2024]

[3] “Flowchart Maker and Online Diagram Software,” draw.io, [Online], Available: https://app.diagrams.net/ [Accessed July 9, 2024]

[4] Xtensio, “User Persona Template and Examples,” Xtensio, 2022, [Online], Available: https://xtensio.com/user-persona-template/ [Accessed July 9, 2024]

[5] C. Garg, “EcoMart” Visual Studio Code,[Tool], [Accessed July 9, 2024]

[6] Figma, "Figma: Web-based design tool," [Online], Available: https://www.figma.com/ [Accessed: July 9, 2024].