Brockport Eats (BPEats)

By

Aschby Gurpersaud

Dominic Adu Agyei

And

Yusuf Unsal

12/6/2023

CIS – 317 Analysis and Design

Professor Eric Owusu

**Introduction:**

* The main problem that students are experiencing is the absence of a food delivery service. With the Brockport Eats system, on-campus students can order ahead of time and have food delivered to them in their corresponding residence halls. They wouldn’t need to wait in line and potentially be late for class.
* Our user interface will be available for mobile devices.
* This system will not only help students but will provide business opportunities for SUNY Brockport as well as help small businesses around the area.
* In addition, students can take the opportunity to get work study or an on-campus job from our system.
* Our inspiration came from other food delivery services such as Uber Eats and Door Dash. The bits and pieces we will incorporate from them are the navigation menu and login panel.
* We investigated into the existing campus dining services for any potential food delivery options and found that they do not have such accommodations.
* However, upon further investigation, we found out that Brockport previously had a food delivery system named Tracks. Taking that into consideration, we will take notes into why their previous system was terminated, and how to combat those obstacles.

**Inputs:**

* For our Brockport Eats System, the user inputs will undergo numerous procedures for processing.
* The accessibility input data will be processed depending on the user status. The database will scan the user data and determine if the user is existing or new.
* The navigational inputs will be processed based what the user selected. The app will process the input to determine where to redirect the user within the app.
* The selection inputs will be processed based on the availability and price of their choices. The application will authenticate and process the user selections, checking and verifying their choices.

**Outputs:**

* After the accessibility input data is processed, the app will output the main home screen, or the account creation page.
* The navigational inputs when processed, will direct the user within the app accordingly.

**Outputs (Cont.):**

* After processing the selection input, the app will generate detailed order confirmations with estimated times of arrival, payment confirmation, and information of what was ordered.
* In addition, order fulfillment receipts will be followed by feedback requests for our services.

**Processes:**

* For our Brockport Eats System, the user inputs will undergo numerous procedures for processing.
* The accessibility input data will be processed depending on the user status. The database will scan the user data and determine if the user is existing or new.
* The navigational inputs will be processed based what the user selected. The app will process the input to determine where to redirect the user within the app.
* The selection inputs will be processed based on the availability and price of their choices. The application will authenticate and process the user selections, checking and verifying their choices.

**Feasibility Assessment:**

1. **Economic:**

* **Cost Estimation**: The development costs, including software development($12,000), design($5,000), testing, server infrastructure, marketing($500), and ongoing maintenance($500).
* **Revenue Projection**: Potential revenue streams, such as delivery fees($1,000), commission from restaurants(%5 - %15), and advertising($250).
* **Return on Investment (ROI)**: The expected ROI by comparing development costs to projected revenues over time. (ROI = [($25,000 - $6,000) / $17,500] \* 100 = 108.6%)
* **Market Research**: There is sufficient demand and not a big competition for local deliveries for Brockport.

1. **Technical Feasibility:**

* **Technology Stack**: Determine the required technologies and infrastructure for the app, including mobile platforms (iOS and Android), backend servers, databases, and APIs.
* **Scalability**: Ability to handle growth in terms of users, orders, and concurrent requests.
* **Security**: Identify potential security risks and outline measures to protect user data and payment information, 2FA.
* **Technical Expertise**: Team having technical skills or if external expertise is required.

**Feasibility Assessment (Cont.):**

1. **Operational-Business Feasibility:**

* **Business Model:** The food delivery app operates on a commission-based, platform-as-a-service (PaaS) model, connecting college students, faculty, and local residents with a wide selection of restaurants and eateries in the vicinity of SUNY Brockport.
* **Operational Processes**: Supply Chain: The operational process for the food delivery app serving SUNY Brockport College involves several key steps to ensure efficient and reliable service. Such as, restaurant onboarding, user registration, orders, processing, delivery, payment, ratings & feedback, marketing & promotions, customer support, scalability, and data analysis.
* **Marketing and User Acquisition**: Target audience, branding, social media marketing, college partnerships, referral program, and influencer marketing.

1. **Legal & Contractual Feasibility:**

* **Regulatory Compliance**: There are not any regulations related to food delivery services, protecting users &businesses data, and increasing local employment.
* **Contracts:** 5% to 15% commission depending on the order size, yearly contract for the first year, if both parties agree then every two years.
* **Intellectual Property**: We will make sure that our app does not infringe on any existing patents or trademarks.

1. **Political Feasibility**

* **Local Policies**: There are not any specific regulations related to food delivery services for our area.
* **Community Engagement**: Most people we interviewed from the local community, including college students and businesses said they will support this idea.

1. **Schedule-Timeline Feasibility:**

* **Member Roles:**

Yusuf – Business Analyst

Aschby – Operations

Dominic –Code/Development

* **Project Timeline**: Meeting Times: 5pm – 8pm Monday, Wednesday, Friday
* **Meeting Place:** Virtual and/or Campus
* **Launch date:** 12/11/23
* **Resource Availability:** We have access to the necessary human and financial resources to meet the project's timeline.
* **Risk Assessment:** Potential risks that could cause delays, such as if our funding is cut, we will create the main parts of our project, then add special features over time.

**Planning Phase – Baseline Project Plan:**

* **Business Value:** BrockportEats aims to enhance the campus community's dining experience by providing a convenient and efficient platform for ordering meals from local restaurants. The app seeks to promote local businesses and offer students a diverse range of food options.
* **Staff:** The project team includes developers, UI/UX designers, and project managers with experience in app development and local business partnerships.

**Analysis Phase - System Requirements:**

* **Strategy:** The app strategy involves creating a seamless, user-friendly platform that connects students with local restaurants, offering a diverse range of cuisines.
* **Use Cases Analysis:** Use cases involve students placing orders, restaurants receiving and fulfilling orders, and drivers delivering orders to users.

**Design Phase - System Specification:**

* **Design Selection:** The chosen design focuses on a mobile-first approach with a clean and visually appealing interface, emphasizing ease of use for both students and restaurant owners.
* **Program Design:** The app's architecture is designed to facilitate efficient communication between the user interface, database, and external APIs for menu integration and payment processing.

**Implementation Phase - New System and Maintenance Plan:**

* **User Interface Design:** The UI is designed to provide an intuitive ordering experience, with features such as a personalized user dashboard, interactive menus, and a seamless checkout process.

**Business Requirements:**

* The business will implement a on campus food delivery system for students.
* The business will generate revenue based on delivery fees and sponsorships.
* The business will create business opportunities for partnerships with small business around the local Brockport area.

**User Requirements:**

* The user must sign up with their student email and Student ID.
* The user must select the appropriate residential hall address.
* The user must choose an appropriate method of card payment.
* The drivers must be within a 20-mile radius of the campus.
* Restaurant must sign up with the appropriate location and within 10 -mile radius.

**Functional Requirements:**

* **User Management**
* 1.1 The system shall enable user registration and account creation.
* 1.2 The system shall allow users to update their personal information, delivery addresses, and payment methods.
* 1.3 The system shall provide users with access to order history and real-time order tracking.
* 1.4 The system shall support a loyalty program for users to earn and redeem rewards.
* **Order Management**
* 2.1 The system shall enable users to browse restaurant listings, select menu items, and customize their orders.
* 2.2 The system shall calculate the total order cost, including delivery fees and taxes.
* 2.3 Users shall have the option to provide feedback and ratings for delivery experiences and food quality.
* 2.4 Orders shall be transmitted to the chosen restaurant for preparation.
* 2.5 The system shall send push notifications to users when their orders are on their way.
* **Restaurant Management**
* 2.6 The system shall display a list of partnered restaurants, including restaurant names, cuisine types, menu items, prices, and ratings.
* 2.7 Restaurant must be able to customize their menu options and prices.
* **Driver Management:**
* 2.7 Driver must be able to view/accept orders.
* 2.8 Driver must be able to cancel orders.
* **Administration**
* 2.9 All user registrations shall be verified by the database administrator.

**Non-functional Requirements:**

* **Operational**
* 1.1 The app should be compatible with tablet devices used by college students and residents.
* 1.2 The app should be web-based and accessible on major web browsers.
* 1.3 The app should facilitate wireless connectivity to printers for order receipts and notifications.
* 1.3 Users can save their login information  for quicker login.
* **Performance**
* 2.1 The app should provide quick response times, with actions completed in 3 seconds or less.
* 2.2 The app's data, including new orders and restaurant information, should be updated every 5 minutes.
* **Security**
* 3.1 User accounts and personal information should be stored securely and in compliance with data protection regulations.
* 3.2 Only authorized personnel, such as the customization shop supervisor, may approve non-standard customizing options.
* 3.3 Access to the app on tablet devices should be restricted to the assigned salesperson.
* **Cultural and Political**
* 4.1 Compliance with company policies regarding technology procurement, such as purchasing computer equipment from specific vendors, should be maintained.

**Overall Use Case Diagram:**

A diagram of a user registration

Description automatically generated with medium confidence

**User Registration:**

* **Use case diagram: U-1**
  + User will enter their student information.
  + Admin will verify the student information in the database.
  + After verification, user account is created.
  + User can now login.
  + User is redirected to the BPeats homepage.

A diagram of a person's figure

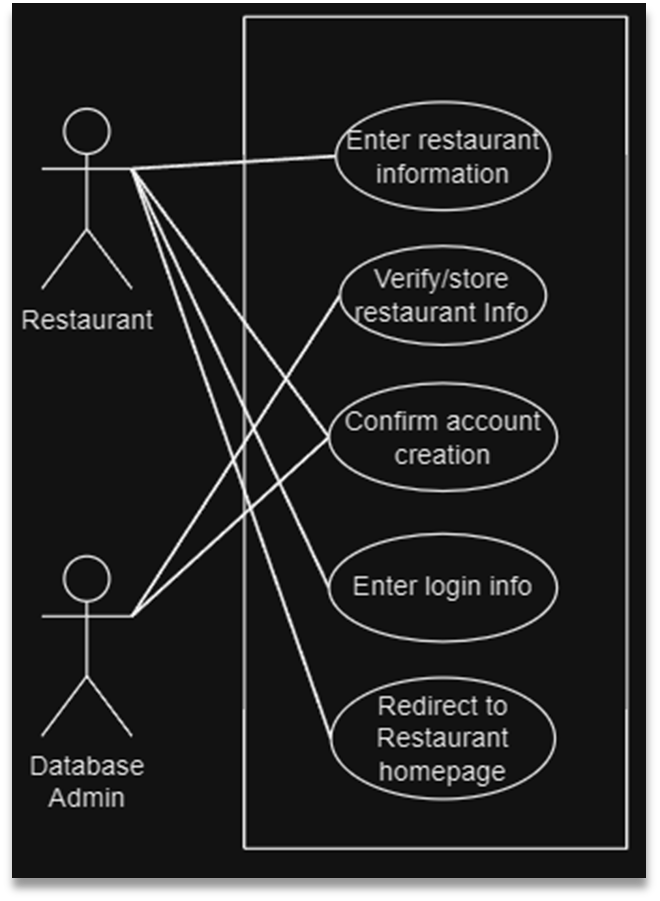
Description automatically generated

**User Registration: U-1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Name: User Registration**Priority: High | | | | **ID: U-1** |
| Short Description: New users register on the Brockport Eats app, providing personal information, creating a username and password, and adding delivery addresses. | | | | |
| Trigger: User wants to create an account.  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  User first last name, Student email, phone, student ID username and password  Delivery address Verification email | Source:  User | Description:  Confirmation email with a verification link  User registration and profile creation | Destination:  System application  System Database | |
| Major Steps Performed:   * The user enters their info. * The system database admin validates the registration information * The system sends a confirmation email for the user to verify their account creation * The system securely stores user information in a database for future use, ensuring data privacy and security. * The user enters their login info * The system database validates their credentials. * System will redirect user to the BPeats homepage after verified login | | Information for steps:  User information  System user Database admin  User email  System database, user info    User login information  System user database  BPeats Homepage | | |

**Restaurant Registration:**

* **Use Case Diagram: U-2**
  + Restaurant will enter their student information.
  + BP Eats database admin will verify the restaurant information.
  + After verification, restaurant account is created.
  + The newly created restaurant account is now visible and customizable.



**Restaurant Registration: U-2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Name: restaurant registration**Priority: High | | | | **ID: U-2** |
| Short Description: Restaurants create register themselves in the app so users can order from them and they can customize their menus. | | | | |
| Trigger: a restaurant wants to register themselves in the app  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  Restaurant name  Menu items  Menu prices  Hours  Location | Source:  Restaurant | Description:  Restaurant enters their info, then gets a profile created in their account which allows them to sell their menu items and customize their menu options. | Destination:  System application ,  System database | |
| Major Steps Performed:   * The restaurant enters their information * The system database admin validates their information by checking their location ensuring that the restaurant is legitimate and within a 10-mile radius of the campus * System will create a restaurant profile for the restaurant * The restaurant can now login to their account * The restaurant can customize their menu items and prices to their liking | | Information for steps:  Restaurant information  System Database, admin, restaurant info  Restaurant and menu items.  Restaurant login info  Restaurant menu items, prices, hours | | |

**Driver Registration:**

* **Use Case Diagram: U-3**
  + Driver will enter their information.
  + Database admin will verify and store the driver information.
  + The driver account is created.
  + The driver can now login to the Bpeats app.

A diagram of a driver

Description automatically generated

**Driver Registration: U-3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Name: Driver registration**Priority: High | | | | **ID: U-3** |
| Short Description: Drivers can register themselves in the BPEats app to conduct deliveries. | | | | |
| Trigger: a driver wants to create an account  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  Driver first last name  Driver vehicle model  Driver age and license status  Direct deposit | Source:  Driver | Description:  Driver enters their information which gets verified, then a newly created driver account is created for them in the BPeats app. | Destination:  System application,  System database | |
| Major Steps Performed:   * The driver enters their information * The system database admin validates their information by checking their driver’s license and ensuring that they own a vehicle. * System then creates the driver account * The driver can now login to their account * Driver can then see their delivery jobs to do in their account and cancel jobs if needed | | Information for steps:  Driver information  System Database, admin, driver info  Driver info  System application  Driver info  Drive login info  Driver account, driver info | | |

**Order Placement:**

* **Use case diagram: U-4**
  + Student enters the order information.
  + Student enters their payment info and checks out the order.
  + The BPeats system generates order receipt.
  + The Bpeats system forwards order details.
  + The BPeats system assigns a driver for the order delivery.

A diagram of a person's figure

Description automatically generated

**Order Placement: U-4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Name: Order Placement                                                                    Priority: High | | | | ID : U-4 |
| Short Description: Users browse restaurant listings, select menu items, customize orders, and place them. | | | | |
| Trigger: User wants to place an order  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  Restaurant name, menu items, prices  User’s information, menu selections, customization payment details and payment method  Driver Name, vehicle model | Source:  User  Restaurant  Driver | Description:  Order confirmation with estimated delivery time  Generated order receipts | Destination:  User  System | |
| Major Steps Performed:   * The user selects their restaurant choice, menu items, and customizations * The user goes to the checkout process and their information is auto filled * Payment processing is initiated, and the user securely provides payment method and information, such as credit/debit card details and choice of payment. * After successful payment processing, the system confirms the order and generates an order receipt with an estimated delivery time. * The system forwards order details, including the selected items and delivery address, to the chosen restaurant for preparation. * The system then assigns a driver to deliver the user’s order and sends the user their descriptive info. | | Information for steps:    Restaurant, menu items, customizations, requests  User information  User payment details and methods  Hidden payment info and method, receipt, user order info, order receipt  Restaurant info, user delivery address  Driver Name, Driver vehicle model | | |

**Order Cancel:**

* Use case diagram: U-5
  + Student enters the cancellation request. (i.e., their reason for canceling)
  + The restaurant will verify the cancellation with the restaurant.
  + After verification, the order will be cancelled, and payment refunded.
  + The driver will be notified and cancel the delivery.

A diagram of a person's figure

Description automatically generated

**Order Cancel: U-5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Name:  Order cancel Priority: High | | | | ID: U-5 |
| Short Description: User orders food in the BP Eats app, but decides they want to cancel their order. | | | | |
| Trigger: User wants to cancel their order  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  Order info  reason for cancellation payment info  Driver info | Source:  User  Driver | Description:  Order cancelation receipt  Email that shows order canceled, and money refunded to the account | Destination:  System application  System Database  User | |
| Major Steps Performed:  - The user cancels their order before their order is picked up for delivery  - The system asks the user the reason for cancelling  - The system verifies the cancellation and checks the current status of the order  - The system generates an order cancellation receipt and email, then notifies the driver | | Information for steps:  User order cancellation request  Reason for order cancellation  User student email, cancellation confirmation receipt, order info  Driver info, student payment info, cancellation receipt | | |

**Order Status:**

* **Use case diagram: U-6**
  + Student checks the status of their order.
  + Student can track their order via the BPeats app.
  + The driver confirms the order delivery with the student, restaurant, and the system.

A diagram of a person's structure

Description automatically generated

**Order Status: U-6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Name: order status** Priority: High | | | | **ID: U-1** |
| Short Description: Users can check the order status of their order and see delivery tracking in real time | | | | |
| Trigger: User wants to check their order status  Type: External / Temporal | | | | |
| Major Inputs: | | Major Outputs: | | |
| Description:  Order info  Driver info  Restaurant info | Source:  User  Driver  Restaurant | Description:  Order is checked with the restaurant and the driver and relays the status | Destination:  System application | |
| Major Steps Performed:   * The user checks their order status. * The system verifies if the order is still in preparation, or if the order is on its way. * The system sends a notification if the order is done with preparation, has been picked up, or is on route via delivery * The user can track their order in the system in real time * The order delivery status is confirmed by the driver via photo between the user and the restaurant * The User account and restaurant account gets a notification of successful delivery after the driver confirms it. * The Driver gets paid via tip and the job is marked as complete | | Information for steps:  Order info  Restaurant, driver info, driver tracking  Order info, restaurant, driver info, driver tracking  Driver info, Driver tracking  Driver info, user info, order info, restaurant info  Driver info, user info, order info, restaurant info  Driver info | | |

**Context Diagram:**

A diagram of a restaurant

Description automatically generated

**Main Data Flow Diagram (Level 0):**

A black screen with white lines and text

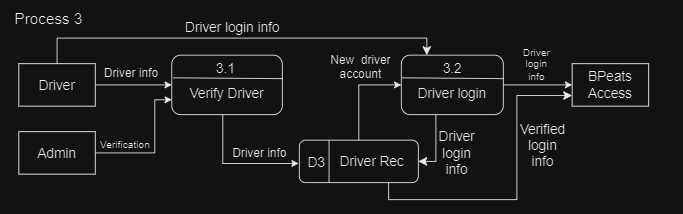
Description automatically generated

A diagram of a student login

Description automatically generated**Process 1 DFD:**

A computer screen shot of a restaurant login

Description automatically generated**Process 2 DFD:**

**Process 3 DFD:**

A diagram of a payment process

Description automatically generated**Process 4 DFD:**

A diagram of a process flow

Description automatically generated**Process 5 DFD:**

**Process 6 DFD:**

A diagram of a delivery process

Description automatically generated

A diagram of a student registration

Description automatically generated**Structure Chart 1:**

A diagram of a restaurant registration

Description automatically generated**Structure Chart 2:**

**A diagram of a driver registration

Description automatically generatedStructure Chart 3:**

A diagram of a payment method

Description automatically generated**Structure Chart 4:**

**A diagram of a flowchart

Description automatically generatedStructure Chart 5:**

A diagram of a delivery process

Description automatically generated**Structure Chart 6:**

**Interface Structure Design:**

A computer screen shot of a diagram

Description automatically generated

**A screenshot of a phone

Description automatically generated**A screenshot of a menu

Description automatically generated**A screenshot of a menu

Description automatically generatedA screenshot of a phone

Description automatically generatedApp Design:**

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 001**

* **Requirement addressed:** User Registration
  + **Objective:** To ensure that users can successfully register on the BrockportEats app.

**Test Case 1.1:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| User Interface | Name | John Doe |
| User Interface | Email | [john.doe@example.com](mailto:john.doe@example.com) |
| User Interface | Password | test@1234 |
| User Interface | Confirm Password | test@1234 |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the registration page.
3. Enter the user's name, email, password, and confirm password.
4. Click the "Register" button.

**Expected Results/Notes:**

* User is successfully registered, and a confirmation message is displayed.

**Actual Results/Notes:**

* The registration process completed successfully. A confirmation email was sent to the provided email address, and the user is able to log in with the registered credentials.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 002**

* **Requirement addressed:** User Login
  + **Objective:** To ensure that registered users can log in successfully.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| User Interface | Email | [john.doe@example.com](mailto:john.doe@example.com) |
| User Interface | Password | test@1234 |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the login page.
3. Enter the user's email and password.
4. Click the "Login" button.

**Expected Results/Notes:**

* User is successfully logged in, and the home screen is displayed.

**Actual Results/Notes:**

* The login process completed successfully. The user was redirected to the home screen, and their account information is accurately displayed. No error messages or unexpected behaviors were observed.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 003**

* **Requirement addressed:** Driver Registration
  + **Objective:** To validate the successful registration process for a new driver.

Test Cases:

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Driver Registration | Driver Information | Fill in Valid Driver Information |

Script:

1. Open the BrockportEats app.
2. Navigate to the driver registration section.
3. Fill in the required driver information fields.
4. Submit the registration form.

Expected Results/Notes:

* The app should successfully register the new driver and provide a confirmation message or notification indicating the successful registration. The driver should now be able to log in with the provided credentials.

Actual Results/Notes:

* After filling in valid driver information and submitting the registration form, the app displayed a confirmation message: "Driver registration successful! You can now log in to start accepting delivery requests." The driver was able to log in with the provided credentials, confirming that the registration process worked as expected. Additionally, a confirmation email was sent to the registered email address.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID:** 004

* **Requirement addressed:** Driver Login
  + **Objective:** To verify the successful login process for a registered driver.

**Test Cases:**

| Interface ID | Data Field | Value Entered |
| --- | --- | --- |
| Driver Login Screen | Email Address | Driver's Valid Email |
| Driver Login Screen | Password | Driver's Valid Password |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the driver login screen.
3. Enter the driver's valid email address.
4. Enter the driver's valid password.
5. Click the "Login" button.

**Expected Results/Notes:**

* The app should validate the entered credentials and log the driver into the app, providing access to the driver's dashboard.

**Actual Results/Notes:**

* After entering the valid email address and password and clicking the "Login" button, the app successfully logged in the driver. The driver was redirected to the driver's dashboard, confirming that the login process worked as expected. Additionally, a welcome message was displayed, and the driver's profile information was accurately loaded.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 005**

* **Requirement addressed:** Placing an Order
  + **Objective:** To confirm that users can successfully place an order for a Chicken Burrito.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Interface | Select Menu Item | Chicken Burrito |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the menu.
3. Select "Chicken Burrito" from the menu.
4. Add the item to the cart and proceed to checkout.

**Expected Results/Notes:**

* The order is successfully placed, and a confirmation is displayed.

**Actual Results/Notes:**

* The order for the Chicken Burrito was successfully placed. The app displayed a confirmation message, and the item appeared in the user's order history. The total amount was correctly calculated, and the expected delivery time was provided. No errors or issues were encountered during the process.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 006**

* **Requirement addressed:** Restaurant Search
  + **Objective:** To verify that users can search for restaurants offering Mexican cuisine.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Search Interface | Search Bar | "Mexican" |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the search bar.
3. Enter "Mexican" in the search bar.
4. Press the Enter key.

**Expected Results/Notes:**

* A list of restaurants offering Mexican cuisine is displayed.

**Actual Results/Notes:**

* The search results displayed a variety of Mexican restaurants. The listings included relevant details such as restaurant name, cuisine type, and average ratings. The search results matched the expected outcome.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 007**

* **Requirement addressed:** App Navigation
  + **Objective:** To confirm that users can navigate between screens seamlessly.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Navigation Interface | Home Screen | N/A |
| Navigation Interface | Menu Screen | N/A |
| Navigation Interface | User Profile Screen | N/A |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the home screen.
3. Access the menu screen.
4. Navigate to the user profile screen.

**Expected Results/Notes:**

* Users can navigate between the home screen, menu screen, and user profile screen without encountering errors.

**Actual Results/Notes:**

* The navigation between screens was smooth and without any delays. Each screen displayed the expected content, and the transition between screens was seamless. No errors or unexpected behaviors were observed during the navigation process.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 008**

* **Requirement addressed:** Order Tracking
  + **Objective:** To verify that users can track an in-progress order.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Tracking Interface | Order ID | 68 |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the order tracking section.
3. Enter the valid order ID for an in-progress order (e.g., 123456789).
4. Click the "Track Order" button.

**Expected Results/Notes:**

* Users should be able to view real-time updates and tracking information for their in-progress order.

**Actual Results/Notes:**

* The order tracking interface displayed real-time updates for the order with ID 68. The status changed from "Preparing" to "Out for Delivery," and the estimated delivery time was accurate. The tracking information matched the actual movement of the delivery driver on the map. The tracking feature worked as expected without any issues.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 009**

* **Requirement addressed:** Delivery Confirmation
  + **Objective:** To confirm that users can successfully confirm the delivery of their order.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Confirmation Interface | Order ID | 68 |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the order confirmation section.
3. Enter the valid order ID for a delivered order.
4. Click the "Confirm Delivery" button.

**Expected Results/Notes:**

* Users should receive confirmation that their order has been successfully delivered.

**Actual Results/Notes:**

* The order confirmation interface displayed a confirmation message stating that order 68 has been successfully delivered. The status of the order changed from "Out for Delivery" to "Delivered," and the order details were updated accordingly. The confirmation process worked as expected without any issues.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 010**

* **Requirement addressed:** Restricted Delivery Areas
  + **Objective:** To test orders placed to areas with specified delivery restrictions.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Placement Screen | Delivery Address | Holmes |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the order placement screen.
3. Enter a delivery address known to be within a restricted area.
4. Proceed to place the order.

**Expected Results/Notes:**

* Users should receive a notification or error message indicating that delivery is not available to the specified address due to restrictions.

**Actual Results/Notes:**

* The app displayed an error message stating, "Delivery to the entered address is restricted. Please choose another delivery address." The user was prevented from proceeding with the order, confirming that the app correctly enforces delivery restrictions. This ensures that orders are not accepted for addresses in restricted areas.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 011**

* **Requirement addressed:** Order Cancellation
  + **Objective:** To test the cancellation of a pending order before final confirmation.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Review Screen | Cancel Order Button | Select Cancel Option |

**Script:**

1. Open the BrockportEats app.
2. Place an order and navigate to the order review screen.
3. Before final confirmation, click the "Cancel Order" button.
4. Confirm the cancellation action.

**Expected Results/Notes:**

* The app should promptly cancel the pending order, and the user should receive a confirmation notification.

**Actual Results/Notes:**

* After clicking the "Cancel Order" button, the app displayed a confirmation message stating, "Your order has been successfully canceled." The order was removed from the user's pending orders list, confirming that the cancellation process worked as expected. Additionally, a notification was received on the user's device, providing immediate feedback about the canceled order.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 012**

* **Requirement addressed:** Order Cancellation
  + **Objective:** To test the cancellation of a pending order after final confirmation.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Order Details Screen | Cancel Order Button | Select Cancel Option |

**Script:**

1. Open the BrockportEats app.
2. Place an order and navigate to the order details screen.
3. After final confirmation, click the "Cancel Order" button.
4. Confirm the cancellation action.

**Expected Results/Notes:**

* The app should promptly cancel the pending order, and the user should receive a confirmation notification. The cancellation should reflect accurately in the user's order history.

**Actual Results/Notes:**

* After clicking the "Cancel Order" button on the order details screen, the app displayed a confirmation message stating, "Your order has been successfully canceled." The order was removed from the user's pending orders list, and the cancellation was reflected in the user's order history. Additionally, a notification was received on the user's device, providing immediate feedback about the canceled order.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 013**

* **Category:** Driver Interactions
* **Requirement addressed:** Driver Availability
  + **Objective:** To test the ability of a driver to set and update their availability status.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Driver Availability Screen | Availability Status | Toggle Between Available/Unavailable |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the driver availability screen.
3. Toggle the availability status between "Available" and "Unavailable."
4. Confirm the status update.

**Expected Results/Notes:**

* The app should allow the driver to set and update their availability status, and the updated status should be reflected in real-time.

**Actual Results/Notes:**

* After navigating to the driver availability screen and toggling the status between "Available" and "Unavailable," the app successfully updated the driver's availability status. The updated status was reflected immediately in the app's user interface. Additionally, a confirmation message was displayed, confirming the status update. The driver's availability status accurately changed as expected.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 014**

* **Requirement addressed:** Order Acceptance
  + **Objective:** To test the driver's ability to accept an in-progress delivery request.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Driver Dashboard | In-Progress Order List | Select an In-Progress Order |
|  | Accept Order Button | Click Accept Button |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the driver dashboard.
3. Locate an in-progress delivery request in the list.
4. Click the "Accept Order" button.
5. Confirm the order acceptance.

**Expected Results/Notes:**

* The app should allow the driver to accept an in-progress delivery request, and the order should be assigned to the driver for completion.

**Actual Results/Notes:**

* After navigating to the driver dashboard and selecting an in-progress delivery request, clicking the "Accept Order" button successfully assigned the order to the driver. The app displayed a confirmation message, indicating that the order was accepted. The order details were updated in the driver's active orders list, confirming that the order acceptance process worked as expected.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 015**

* **Requirement addressed:** Order Completion
  + **Objective:** To test the driver's ability to confirm the completion of a delivered order.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Driver Dashboard | Active Orders List | Select a Delivered Order |
|  | Complete Order Button | Click Complete Button |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the driver dashboard.
3. Locate a delivered order in the active orders list.
4. Click the "Complete Order" button.
5. Confirm the order completion.

**Expected Results/Notes:**

* The app should allow the driver to confirm the completion of a delivered order, and the order status should be updated accordingly.

**Actual Results/Notes:**

* After navigating to the driver dashboard and selecting a delivered order, clicking the "Complete Order" button successfully marked the order as completed. The app displayed a confirmation message, indicating that the order completion was successful. The order status was updated in the driver's order history, and the completed order was removed from the active orders list. The completion process worked as expected.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 016**

* **Requirement addressed:** Restaurant Onboarding
  + **Objective:** To test the process of adding a new restaurant to the BrockportEats app.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Restaurant Onboarding Form | Restaurant Information | Fill in Valid Restaurant Information |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the restaurant onboarding section.
3. Fill in the required restaurant information fields.
4. Submit the onboarding form.

**Expected Results/Notes:**

* The app should successfully onboard a new restaurant and provide confirmation to the restaurant owner.

**Actual Results/Notes:**

* After navigating to the restaurant onboarding section and filling in valid restaurant information, submitting the onboarding form successfully added the new restaurant to the app. The app displayed a confirmation message, indicating that the restaurant onboarding was successful. The new restaurant details were visible in the restaurant dashboard, confirming that the onboarding process worked as expected.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 017**

* **Requirement addressed:** Menu Availability
  + **Objective:** To test the real-time update of menu availability in the BrockportEats app.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Restaurant Dashboard | Menu Availability | Update Availability for Items |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the restaurant dashboard.
3. Locate the menu availability section.
4. Update the availability status for selected menu items.
5. Confirm the menu availability update.

**Expected Results/Notes:**

* The app should allow the restaurant owner to update the availability status of menu items in real-time, and the changes should be reflected immediately.

**Actual Results/Notes:**

* After navigating to the restaurant dashboard and updating the availability status for selected menu items, the changes were immediately reflected in the app. The app displayed a confirmation message, indicating that the menu availability update was successful. The updated availability status was visible in the restaurant's live menu, confirming that the real-time update worked as expected.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 018**

* **Requirement addressed:** Order Notification
  + **Objective:** To test the restaurant's ability to receive and confirm new order notifications in real-time.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Restaurant Dashboard | New Order Notification | Receive and Confirm New Order |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the restaurant dashboard.
3. Wait for a new order notification.
4. Confirm the receipt of the new order.
5. Accept and confirm the new order.

**Expected Results/Notes:**

* The app should promptly notify the restaurant about new orders in real-time, and the restaurant should be able to confirm and accept the new order.

**Actual Results/Notes:**

* After navigating to the restaurant dashboard and waiting for a new order notification, the app successfully notified the restaurant about a new order. The notification was clear and displayed relevant details, such as order items and customer information. Confirming the receipt of the new order and accepting it worked as expected. The order details were immediately updated in the restaurant's active orders list, confirming that the order notification and confirmation process worked in real-time.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 019**

* **Requirement addressed:** Order Preparation
  + **Objective:** To test the restaurant's ability to update the order preparation status in the BrockportEats app.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Restaurant Dashboard | Order Preparation Status | Update Preparation Status for Selected Order |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the restaurant dashboard.
3. Locate an order in the active orders list.
4. Update the order preparation status.
5. Confirm the order preparation update.

**Expected Results/Notes:**

* The app should allow the restaurant to update the order preparation status for selected orders, and the changes should be reflected immediately.

**Actual Results/Notes:**

* After navigating to the restaurant dashboard and updating the order preparation status for a selected order, the changes were immediately reflected in the app. The app displayed a confirmation message, indicating that the order preparation status update was successful. The updated status was visible in the restaurant's active orders list, confirming that the order preparation status could be updated as expected. The real-time update worked seamlessly.

**Test Plan Page 1 of 1**

* **Program ID:** BrockportEats
* **Version Number:** 1.0
* **Tester:**
* **Date Designed:**
* **Date Conducted:**
* **Results:** Passed

**Test ID: 020**

* **Requirement addressed:** Order Completion
  + **Objective:** To test the restaurant's ability to confirm the delivery of an order to the assigned driver.

**Test Cases:**

| **Interface ID** | **Data Field** | **Value Entered** |
| --- | --- | --- |
| Restaurant Dashboard | Completed Orders List | Select a Delivered Order |
|  | Confirm Delivery to Driver | Click Confirm Delivery Button |

**Script:**

1. Open the BrockportEats app.
2. Navigate to the restaurant dashboard.
3. Locate a delivered order in the completed orders list.
4. Confirm the delivery of the order to the assigned driver.

**Expected Results/Notes:**

* The app should allow the restaurant to confirm the delivery of an order to the assigned driver, and the order status should be updated accordingly.

**Actual Results/Notes:**

* After navigating to the restaurant dashboard and selecting a delivered order, clicking the "Confirm Delivery to Driver" button successfully confirmed the delivery of the order to the assigned driver. The app displayed a confirmation message, indicating that the delivery confirmation was successful. The order status was updated in the restaurant's order history, and the order was removed from the completed orders list, confirming that the delivery confirmation process worked as expected.

**Pseudocode:**

**Student Registration:**

PROMPT Student to Enter their Brcokport Email and information

GET their Brockport email and information

STORE their information in the database

**Restaurant Registration:**

PROMPT Restaurant to Enter their information

GET their restaurant information

STORE their information in the database

**Driver Registration:**

PROMPT Driver to Enter their vehicle and license information

GET their driver information

STORE their information in the database

**Student login:**

PROMPT Student to enter their login information

GET their login information

VERIFYtheir information in the database

**Restaurant login:**

PROMPT Restaurant to enter their login information

GET their restaurant login information

VERIFY their information in the database

**Driver login:**

PROMPT driver to enter their login information

GET their driver login information

VERIFY their information in the database

**Student Order:**

PROMPT Student to choose their order

GET their order information

VERIFY order inventory

COMPUTE order total

**Payment:**

PROMPT Student to enter payment information

READ their payment information

VERIFY payment information

GENERATE order receipt

**Student Order:**

PROMPT Student to choose their order

GET their order information

VERIFY order inventory

COMPUTE order total

**Driver delivery confirmed:**

PROMPT driver to confirm delivery

READ their input from confirm delivery

GENERATE delivery confirmation

**References:**

* Hillman, Tricia. BP Eats Interview. Conducted by Dominic Adu Agyei, Yusuf Unsal, Aschby Gurpersaud. 12 October 2023.
* Coworkers from local businesses. BP Eats Interview. Conducted by Dominic Adu Agyei, Yusuf Unsal, Aschby Gurpersaud. 11 September 2023.
* Classmates/students (future users). BP Eats Interview. Conducted by Dominic Adu Agyei, Yusuf Unsal, Aschby Gurpersaud. 8 September 2023.
* Multiple RA's. BP Eats Interview. Conducted by Dominic Adu Agyei, Yusuf Unsal, Aschby Gurpersaud. 8 September 2023.
* “Manage Operations with the Uber Eats Orders App | Uber Eats.” *Uber Eats*, Uber Eats, merchants.ubereats.com/us/en/technology/manage-orders/uber-eats-orders-app/. Accessed 14 Sept. 2023.
* “How DoorDash’s Strategy Achieved Food Delivery Domination.” *Latana.com*, resources.latana.com/post/doordash-success-story/. Accessed 29 Sept. 2023.