American International University-Bangladesh (AIUB)

Department of Computer Science

Faculty of Science & Technology (FST)

Fuel Finder App

A Software Quality and Testing Project Submitted
By

Semester: Fall_23_24				Section: C	Group No: 03
SL	**SN**	**Student Name**	**Student ID**	Individual Contribution (in %)	Total Marks: 50 Earned Marks:
A	**26**	**BHOWMIK, MOUMITA**	**21-45616-3**	33%	
B	**28**	**NIHAL, SADMAN TAHMID**	**21-45629-3**	33%	
C	**32**	**ISLAM, MD. NAHIDUL**	**22-46332-1**	34%	

The project will be Evaluated for the following Course Outcomes

EVALUATION CRITERIA	**Total Marks (50)**
System Features	[10 Marks] **A: B:** **C: D:**
Quality Attributes	[10 Marks] **A: B:** **C: D:**
Plan for Testing Levels	[10 Marks] **A: B:** **C: D:**
Test cases, Format, Submission	[20 Marks] **A: B:** **C: D:**

Software Test Plan

for

<Fuel Finder App>

Version 1.0 approved

Prepared by < **BHOWMIK,MOUMITA** ; **NIHAL, SADMAN TAHMID** ; **ISLAM, MD. NAHIDUL** >

<AIUB>

<22nd September,2024>

Table of Contents

System Features

1. Login

Functional Requirements

1.1 The user shall be granted access to the system through his/her login credentials - username and password.
1.2 Information filled on the Login page will be crossed with the database entry.
1.3 Users shall have an option to reset their password via an email link.
1.4 If the login information matches, the application will forward the page to the home page according to the account logged into the system. In case, if the system detects that the user's role is a user then if the user's role is a management, the system logs into the system management panel.
1.5 In case of incorrect username or password, the system will ask the user to log in again by asking for a username and password once more.
1.6 If fails to enter the username and password exceed the limit of continues 3 times, random verification code is created by system for retry login, verify whether a user is a human or not with a CAPTCHA.
1.7 The system shall lock the user account login for half hour when the number of login attempts exceed their limit, i.e. 10 times.

Priority Level: High
Precondition: User have valid user id and password.
Cross Reference: 3.4, 7.1, 7.5, 8.1

2. Signup

Functional Requirements

2.1 The system shall allow new users to sign up providing name, email, phone number, and password.
2.2 The system shall generate an acknowledgement email/message with verification link/code for validation of account.
2.3 If all the information of data is provided on the page then it will forward the user to Another page for providing the username and password for his account.
2.4 The system shall impose a password strength that shall include at least 8 characters in Length, letters, numbers and special symbols.
2.5 In a case when the username and password field is filed with appropriate data then user's data and sign-in credentials to the system database
2.6 In a case when the username and password field is not filled with appropriate data then the system will ask for re-writing of data again.
2.7 In case of all of the information of the data is not given in the system then the system will ask to enter all the information by highlighting which one is empty.
2.8 The system shall provide an option to register through third-party systems i.e Google or Facebook, etc.

Priority Level: High
Precondition: Users shall provide valid email or phone number for account verification, different icons or tags to draw users' attention.
Cross Reference: 7.1, 7.2, 7.4, 7.5

3.Search And View Category

Functional Requirements

3.1 The User shall be able to view what search criterions bring suitable fuel.
3.2 The system shall allow the user to search for various types of fuel details available in the area.
3.3 List items matching the entered search by the user.
3.4 The system shall allow users to view specific fuel type availability in real time.
3.5 Able to choose fuel type based on users' need.
3.6 System to show search results as lists, and map views, with distance and approximate time of arrival.

Priority Level: High
Precondition: Users must grant permission for the location access to be used while searching for nearby.
Cross Reference: 4.2, 4.6, 5.1, 5.2, 6.2, 6.5

4. Location Session

Functional Requirements

4.1 The application shall access the GPS of the device to show real-time location and nearby options.
4.2 Display the nearest fuel stations based on GPS or manually fed locations
4.3 The system should provide the user with the capability to choose preferred navigation applications that will be used to open directions in.
4.4 There should be a map indicating the station location along with information.
4.5 The user should be able to find the nearest fueling stations based on his/her current location or by entering specific address or location.
4.6 Application should return a list of closest stations w/ distance from current user's position.

Priority Level : High
Precondition: Users to allow GPS/ location services.
Cross Reference: 3.2, 6.2, 6.4, 7.2

5.Display Fuel Price:

Functional Requirements

5.1 The system shall display the current fuel price for various types of fuel, such as gasoline and diesel, for the stations in its neighborhood.
5.2 The system shall deliver a compare view of the fuel prices of some of the filling stations in its vicinity.
5.3 The system shall enable checking the trends history in fuel prices of selected Gas Stations or places.
5.4 The system should be able to enable the user to sort fuel prices from nearest to farthest area, price range, and/or specific preferred brands.

5.5 The system shall enable creating price alerts, user-defined, which on an event of fuel price drop below the user threshold sends automatically an e-mail to the user.
5.6 The system shall provide information about any filling station with details about its facilities, time of operation, and services offered.
5.7 The system shall allow the user to rate and add comments to filling stations about their service quality, cleanliness, and how easy it is to access.

Priority Level: High
Precondition: User has granted location access or manually entered location and
Is at location screen along with current price.
Cross Reference: 3.4, 3.6, 7.6, 8.3, 8.4, 8.5

6. On-Road Features
Functional Requirements

6.1 The system shall provide an estimate of the range on remaining fuel given a user's current fuel level and his or her vehicle's fuel efficiency coupled with his or her intended route, if any.
6.2 System will identify nearest gas stations based on either the user's intended route or current location of the user.
6.3 P Rentals will enter the threshold value of the warning for low fuel level. It should, in turn, return a warning if the range of remaining fuel is below the threshold value set.
6.4 The system will also be able to provide turn-by-turn navigation to selected gas stations based on user preference.
6.5 The system should also show in real time the traffic situation and should indicate, if delays are detected, alternative detours to reach the chosen gas station.
6.6 If supported, the system integrates into the user's car dashboard and automatically updates fuel level and consumption records.
6.7 The system will provide a facility to the user to filter the gas stations according to price, amenities, or fuel types in a swift way with mobility.

Priority Level: Medium
Precondition: User has typed in current fuel level and possibly a route planned.
Cross Reference:4.1, 4.3, 4.6, 5.6

7. Safety and Privacy
Functional Requirements

7.1 The system shall securely store user credentials using latest industry criteria for encryption mechanisms so that data is protected.
7.2 The system shall record the location of a user only when explicit permission is granted by the user.
7.3 The system shall ensure, in a transparent manner, the implementation of a privacy policy indicating what kind of user data is being collected and why.
7.4 The system shall not distribute or share any personal location information of the users with third-party organizations unless requested explicitly by the users.
7.5 Multi-Factor Authentication shall be optionally available with the aim of ensuring

improved security of the user's account.
7.6The system should allow viewing and revoking location access, among other permissions included in settings applications, at will.
7.7 The system should notify users each time their data is being accessed or used, for instance, when offering personalized recommendations or analytics.
7.8 The system should automatically run periodic audits of security protocols of the system and update the same to keep it consistent with updated data protection regulations.
7.9 The system should be able to delete any accounts and hence all information provided by a particular user upon the request of that user with specific guidelines on how this can be done.

Priority Level: High
Precondition: The user has successfully registered an account and/or signed in with valid credentials. Application request user to grant location access. User accepts/denies the application access location.
Cross Reference: 1.3, 2.8, 4.1, 5.5, 6.5, 11.2

8.Feedback and Support:
Functional Requirements
8.1 Feedback functionality, problem-reporting through the app.
8.2 The system will provide him with various options to reach support for any questions he may have and any problems he might encounter regarding the use of the application. As an example, this can be achieved by providing him with FAQs, internal chat within the app, or just e-mail possibility.
8.3 Rating and commenting about the gas station: The user should be able to provide feedback within the system.
8.4 The system shall provide a facility to view and sort the gas station based on average of users' rating/reviews.
8.5 The system shall provide provision for users to report any wrong fuel price.
8.6.5ther information at a station to help in the accuracy of the data on the app.

Priority Level: Medium
Precondition: The user has successfully registered an account and/or signed in with valid credentials.
Cross Reference: 5.5, 5.7, 9.4, 9.5

9. User Reviews and Ratings:
Functional Requirements:

9.1 The system shall provide a facility to leave reviews and ratings about the features or services of the app.
9.2 The user shall have an option to rate a maximum of 5-star and minimum 1-star rating for this application.
9.3 The system shall show the average rating, based on all users' reviews and ratings visible to the future users.
9.4 The system shall provide facility to filter reviews like 5-star, 4-star etc and recent

feedback.

9.5 The system shall provide facility to report improper or spam reviews. Those reviews need to be moderated.

9.6 Users can allow review editing or deletion.

Priority Level: Medium
Precondition: The user needs a valid account to put up a review or rating.
Cross Reference: 5.7, 8.1, 8.3, 8.4, 8.5

10. Filter and Sort Option:

Functional Requirements:

10.1 The software shall provide an option to filter the search results which shall include but not be limited to the options of: price range, distance, fuel type and its availability etc.

10.2 The software shall provide options to sort them out: according to Price- in ascending/descending order, according to Distance- nearest first, according to Fuel type- diesel, gasoline, etc.

10.3 The system should not favor any one of the listed options by default, available in a list, showing all in case filtering is off.

10.4 Multi-filtering: Any user should be able to apply more than one filter in combination with each other to get further filtered results.

10.5 Refresh of the result on same page: No full loading of the page on filters and sorting options applied to the system.

Priority Level: Medium
Precondition: The user has access to the app and the location services are available to him.
Cross Reference: 3.4, 3.6, 5.1, 5.4, 6.7

11. Payment Option

Functional Requirements

11.1 The system shall provide an option to the user for selection of his preferred mode of payment, that is, credit card, debit card, or any digital wallet for purchasing fuel.

11.2 The system shall store the user's payment information in such a way that it is secured and may be used in subsequent transactions if the user so prefers to save payment information.

11.3 The software shall provide the users with the capability to proceed with a payment once they have confirmed the quantity of fuel and the total price for this.

11.4 The system should check whether the mode of payment is valid and whether it has sufficient funds available in it to carry out the transaction.

11.5 If the payment fails, then the failed status should be presented to him; he should again give other modes of paying or retry in the meantime.

11.6 If the payment continuously fails over three times consecutively, then the system

shall generate a random verification code to let him try again for security purposes.
11.7 In case of more than five tries, the system shall freeze the user's options to make the payment, temporarily for an hour period.

Priority Level: High
Precondition: User should have valid payment information with sufficient balance on his selected method of payment.
Cross Reference: 1.6, 5.1, 5.4, 7.1, 7.5

12. Booking Serial:
Functional Requirements:

12.1 The software shall provide an interface to a user to book a serial number, showing the available time slots.
12.2 A user shall be allowed to select his/her date and his/her desired time slot from the given time slots.
12.3 A unique serial number shall be generated for the user after confirmation and shall be dispatched to the user via email or SMS.
12.4 In case of any user trying to make a booking with more than one serial number in less than a 24 hour period, it shall flash a warning to the user and not allow him/her to extra book.
12.5 Any user shall be allowed to, at any time before his chosen date and time slot, cancel or reschedule his booking in the system.

Priority Level: High
Precondition: User must have an active account and be logged in.
Cross Reference: 3.1, 5.1, 7.4, 8.1, 8.3

2. System Quality Attributes

QA1 - Usability

The ability of users to access an app and use it successfully to achieve their goals.The Fuel Finder App should be usable in such a way that the user can intuitively navigate through available fuel stations around them and their price. A very usable design would smoothly allow the user to do the following:

1. Easy in of location or smooth access to location.
2. See list/map of stations around with price information.
3. Filter/sort the stations by fuel type/distance/price.
4. Regular search request : perform in average 2s, maximum 5s under high load.

Example A user opens the app, inputs the city, and sees in a few seconds a list of fuel stations with current prices.

QA2 - Performance

This is the time that it takes to deliver the system's response from an interaction done by the user to carry on with the task at hand. The Fuel Finder App shall process any request by a user-for example, fetching the nearest fuel station list to the user and/or showing the price of the fuel-in minimum time. Poor performance is an annoyance to users. Desirable - Optimal: The app shall return results either from its database or from external services, like a fuel station API, with an average of 3 seconds. For instance, during rush hour or poor network conditions, a response should not take more than 7 seconds. Smoothen the loading of the map; there would be no lag in scrolling.

Example: In case any location is searched by the user, the map and list of fuel stations should load fast, and the user will smoothly be able to scroll through their results.

QA3 - Reliability

The dependability of performance of an application under normal conditions. The Fuel Finder App should always be up and available for those moments when users need to use it. Thus, every minute it is down or crashes-it degrades the experience to the users. It shall make sure that it is available to the users for 99.9%, which means it may go down during the year for no more than 8.76 hours, can handle an error in a graceful manner, such as a failed API request or a network connectivity problem, by retrying the operation or allowing the user to become informed that there is a problem.

Example: The application should always remain online and ready at all times for any end user to search for fuel stations, even at peak usage times.

QA4 - Scalability

The application should be able to scale up and support exponential growth in users without degradation in performance. As the application gains in popularity, it will be expected to handle an ever-increasing number of concurrent users without slowing down or crashing. It will realize no degradation in performance up to 100000 concurrent users. Resources auto-scale up in order dynamically to take full use of the cloud infrastructure when the user demands are high.

Example: Holiday weekends, since users travel more, cause spikes in the number of users' queries; responses by the application stay under 7 seconds.

QA5 - Security

Security ensures that sensitive user data is protected against unauthorized access. Since Fuel Finder App contains personal information related to location history and details of payment, these need to be secure. Related to the requirement of security in the app, all sensitive information should be encrypted by AES-256 encryption so that upon interception, data would not be readable. This would allow the user to implement MFA in such a way that an added layer of security can be introduced at the time of attempting to access the account.

Example: Every time users refuel or store any modes of payment, it encrypts information such as that; hence, MFA would be needed to access them.

QA6 - Maintainability

Maintainability defines how easily an application is brought up to date, or fixed without causing any downtimes or affecting the users whatsoever. A successful application would see periodic updates in the form of bug fixes, performance enhancements, and feature additions. To provide updates gracefully, the application's maintenance tasks-that is, deployment of bug fixes, or performance patches-must be done without taking the application offline using rolling updates, or similar techniques. Critical fixes should be deployed in a couple of minutes with least disruption of user services.

Example: There is a bug in the pricing algorithm. The team pushes the fix, and it does not require the app to go offline, plus users don't even notice the update.

QA7 - Availability

It refers to how frequently the application is accessible and usable for the users. The highest availability shall be during the travel to and from work. The site shall always be up. Planned maintenance shall not exceed more than 2 hours of outage per month. The app should have redundancy and failovers such that when one server goes down, another one kicks in with minimum disruption.

Example: During the time the server is brought down for maintenance, the users can still make use of the App, though some functionalities faced small disruption.

QA8 – Interoperability

Interoperability means how well an application interacts with other systems and third-party services. The service depends on a lot of third-party services. Among them are map and fuel price databases, to name a few, as well as payment systems. For faultless operations diverse external APIs temporary failures should not bring the app down. For example, if some fuel price API is down, at least the application could present cached data or at least be informative to the user about this instead of failing completely.

Example: If the map service goes down it lists the fuel stations instead of showing a map - horrible user experience.

QA9 - Efficiency

This defines the degree to which an application is well equipped with resources such as data usage and battery life. The app should not have a heavy impact on users' devices and networks. Data Usages: optimization to less than 10 MB per hour of continuous use. The application should not consume more than 5% of the device's battery life during usage within a window of 30 minutes, especially with such use involving GPS and maps.

Example: A user can comfortably use the app to find fuel prices for as long as he or she may want without experiencing severe battery drain and high data consumption.

QA10 - Testability

It is the case of running tests on an application for performing as expected. The app is to be designed in such a way that its code allows tests to run upon it and simulate bugs efficiently and in an early manner. That means core features-finding fuel station and processing payment-should be covered by at least 90% unit and integration tests. Automation tests run after every update-newly added code shouldn't break any features that were previously working.

Example: Tests run before releasing a new feature make sure searching for stations and paying for fuels still work exactly as expected without having to individually manually test each feature.

Testing Levels

1.Unit Testing

This is to ensure that the features of the application are tested component by component. Ensuring each and every component functions well individually hence identifying problems in their formative stages.

Following are the components to be subjected for tests:

Users' registration module: Thorough testing of the input on name, email, phone number, password. That all of them are in proper format, for instance, correct format of email, minimum password length.

Fuel Price Calculation: The different algorithms will be passed the predicted fuel range remaining for differing levels of fuel and other measurements pertinent to the vehicle's efficiency.

Location Services: Emphasis will be placed on the accuracy of location provided but also on the reliability of gas station identifications given the known locations as output.

Submission of Feedback: Feedback submitted should be properly processed and stored in the database.

Test Cases:

- It should take an edge case like empty field, special character, very long input.

- Expected behavior on valid input: Confirm users have been registered successfully with correct fuel calculation.

2. **Integration Testing**

The goal is to ensure that integration testing will have various components of an application work together for seamless functioning from one module to another. Integration tests can be done at the following places:

Price Comparison Feature: Fetching price feature shall be integrated with UI to show correct effective real-time data.

Registration and Notification: One has to test whether user registration must trigger notifications, which in turn would lead to an email/ SMS and thus is effective for proper communication between the modules of registration and these notification services.

Navigation and Location Services: The navigation features shall feed appropriate and correct data from the location services to the user for finding a gas station.

Test Cases:

• A confirmation email/SMS has been generated appropriately.

• Integrate feeds from various APIs at respective fuel stations and check whether the fuel price coming in the UI is correct.

• The test shall switches all the components on and see that they talk to each other from Main screen to selected gasoline station.

3. **System Testing**

The aim will be to confirm the integrated software product against the requirements specified with a view to assuring its overall functionality and proper performance in a controlled environment.

Testing Aspects:

Functional Testing: Extended functionality testing of user registration, listing of fuel prices, route navigation will be done to ensure that mentioned features work as intended.

Performance Testing: Apply performance tests using simulation techniques under variable loads so that at least a set of users are on the system at one time; measure the response times of the system with stability.

Security Testing: Perform adequate testing mechanisms for various types of security-related problems like SQL injections, data leaks, and other major security issues.

Test Cases

• A very good example would be user registration, subsequently looking for fuel prices, and navigating to the selected station in order to make sure everything interacts.

• These are tests executed to understand how the application performs when heavy loads are thrown at the system. Define what the maximum number of concurrent users can be handled with no deterioration in performance.

4. **Acceptance Testing**

The objective of this acceptance testing will be to ensure that the application is meeting the business requirements and is ready to Go-Live. The motive would be to provide the final product that would meet user expectations.

Acceptance Criteria:

There is feedback from Beta testing users of the application regarding overall satisfaction and possible improvement points. Examples of key features to be reviewed for all requirements to be met include but are not limited to the following: price alerts, fuel range estimation, and options of signing up using third-party accounts.

Test Cases :

- Conduct User Acceptance Testing on a focus group. Ensure feedback addresses real use: usability, functionality, and any potential problems arising in ordinary use.
- The application should affirm its functionality and usability on the completion of all acceptance criteria mentioned in the requirement specification.

Table 1: Test Case for **Login**

<table>
<tr><td colspan="3">Project Name: <b>Fuel Finder App</b></td><td colspan="2">Test Designed by: Sadman Tahmid Nihal</td></tr>
<tr><td colspan="3">Test Case ID: FR_1</td><td colspan="2">Test Designed date: 19/09/2024</td></tr>
<tr><td colspan="3">Test Priority (Low, Medium, High): High</td><td colspan="2">Test Executed by: Sadman Tahmid Nihal</td></tr>
<tr><td colspan="3">Module Name: Login</td><td colspan="2">Test Execution date:19/09/2024</td></tr>
<tr><td colspan="5">Test Title: Verify user can log in with valid username and password.</td></tr>
<tr><td colspan="5">Description: Test the login functionality with valid and invalid username and password</td></tr>
<tr><td colspan="5">Precondition (If any): The user must be registered with a valid email/username and password.</td></tr>
<tr><td><b>Test Steps</b></td><td><b>Test Data</b></td><td><b>Expected Results</b></td><td><b>Actual Results</b></td><td><b>Status (Pass/ Fail)</b></td></tr>
<tr><td>1.Open the app.
2.Click on the login button.
3.Enter valid username/email and password.
4.Enter invalid username/email or password.
5.Enter valid username and passwords but fail multiple times</td><td>Username: Sadman Tahmid Nihal
Password: 2314nih@l
Username: Sadman Tahmid
Password: 2314 nih@l
Username: Sadman Tahmid Nihal,
Pass: 12(tried 5 times)</td><td>The user is logged in successfully and redirected to the dashboard.
Error message is displayed: "Invalid username or password."
Account is temporarily after five unsuccessful tries.</td><td>As expected.</td><td>Pass</td></tr>
<tr><td colspan="5">Post Condition: The user is either logged in or shown an error message for invalid login attempts.</td></tr>
</table>

Table 2: Test Case for **Signup Session**

<table>
<tr><td colspan="3">Project Name: <b>Fuel Finder App</b></td><td colspan="2">Test Designed by: Sadman Tahmid Nihal</td></tr>
<tr><td colspan="3">Test Case ID: FR_2</td><td colspan="2">Test Designed date: 19/09/2024</td></tr>
<tr><td colspan="3">Test Priority (Low, Medium, High): High</td><td colspan="2">Test Executed by: Sadman Tahmid Nihal</td></tr>
<tr><td colspan="3">Module Name: Signup Session</td><td colspan="2">Test Execution date:19/09/2024</td></tr>
<tr><td colspan="5">Test Title: Verify user can register with valid details.</td></tr>
<tr><td colspan="5">Description: Test the signup functionality using valid and invalid user inputs.</td></tr>
<tr><td colspan="5">Precondition (If any): The user should not already have an account with the same email or phone number.</td></tr>
<tr><th>Test Steps</th><th>Test Data</th><th>Expected Results</th><th>Actual Results</th><th>Status (Pass/ Fail)</th></tr>
<tr><td>1. Open the app.
2. Click on the sign-up button.
3. Enter valid user information (name, email, phone number etc).
4 Enter an already registered email/phone number.
5. Enter a password that is weak.</td><td>Username: Sadman Tahmid Nihal Password: 2314nih@l
Username: Sadman Tahmid Nihal Password: 2314nih@l
Username: Sadman Tahmid Nihal, Pass: 2314</td><td>Registration is successful, and the user receives a confirmation email or SMS.
Error message: "This email/phone number is already in use."
Error message: "Password must be at least 8 characters, include a number, and a special character."</td><td>As expected.</td><td>Pass.</td></tr>
<tr><td colspan="5">Post Condition: User is either successfully registered or shown an appropriate error message.</td></tr>
</table>

Table 3: Test Case for **Search and View Category**

Project Name: **Fuel Finder App**	Test Designed by: Sadman Tahmid Nihal
Test Case ID: FR_3	Test Designed date: 19/09/2024
Test Priority (Low, Medium, High): High	Test Executed by: Sadman Tahmid Nihal
Module Name: Search and View Category	Test Execution date:19/09/2024
Test Title: Verify user can search for and view nearby fuel stations.	
Description: Test the functionality of searching and viewing fuel stations based on location and fuel type.	
Precondition (If any): The user must have GPS/location services enabled.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/ Fail)**
1. Open the app. 2. Click on the Search button. 3.Allow location access. 4.Filter search by fuel type (e.g., petrol, diesel). 5.Select a fuel station from the list. 6.View station on the map.	Location: Enabled Fuel Type: Diesel Station ID: 124	Map or list of nearby fuel stations is displayed. The filtered list of stations with the specific fuel is displayed. Station details (address, fuel price, availability) are shown. Map is updated with directions and estimated time of arrival.	Not expected.	Fail.
Post Condition: User is either successfully registered or shown an appropriate error message.				

Table 4: Test Case for **Location Session**

Project Name: **Fuel Finder App**	Test Designed by: Sadman Tahmid Nihal
Test Case ID: FR_4	Test Designed date: 19/09/2024
Test Priority (Low, Medium, High): High	Test Executed by: Sadman Tahmid Nihal
Module Name: Location Session	Test Execution date:19/09/2024
Test Title: Verify user can set and change preferred location	
Description: Test the ability of the system to set and save the preferred location of the user.	
Precondition (If any): The user needs to have a valid location that they want to set as their preferred location and must on the GPS of their devices.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/ Fail)**
1. Go to the settings page. 2. Click on the location button. 3. Give access to the open device's location. 4. Set a preferred location. 5. Search for Nearby Fuel Stations. 6. Filter the list of nearby fuel stations by selecting a specific fuel type (e.g., diesel, gasoline, electric). 7. Select the required fuel station. 8. Click on Get directions to get the directions of that fuel station. 9. Click the "Save" button.	Location: Mirpur, Dhaka, Bangladesh Fuel Type: Diesel Location: Enabled Type: Diesel	The preferred location is saved and displayed correctly. A list of nearby stations with diesel is displayed. Preferred location is updated.	Not expected.	Fail
Post Condition: The new preferred location is saved and displayed correctly.				

Table 5: Test Case for **Display Fuel Price**

<table>
<tr><td colspan="3">Project Name: <b>Fuel Finder App</b></td><td colspan="2">Test Designed by: MD. Nahidul Islam</td></tr>
<tr><td colspan="3">Test Case ID: FR_5</td><td colspan="2">Test Designed date:20/09/2024</td></tr>
<tr><td colspan="3">Test Priority (Low, Medium, High): High</td><td colspan="2">Test Executed by: MD. Nahidul Islam</td></tr>
<tr><td colspan="3">Module Name: Display Fuel Price</td><td colspan="2">Test Execution date:20/09/2024</td></tr>
<tr><td colspan="3">Test Title: Find the price of fuel</td><td colspan="2"></td></tr>
<tr><td colspan="3">Description: Fuel price with other details.</td><td colspan="2"></td></tr>
<tr><td colspan="5">Precondition (If any): User has granted location access or manually entered a specific location and viewing location with the current price</td></tr>
<tr><td><b>Test Steps</b></td><td><b>Test Data</b></td><td><b>Expected Results</b></td><td><b>Actual Results</b></td><td><b>Status (Pass/Fail)</b></td></tr>
<tr><td>1. Log into the app.
2. Click on the option bar and find Fuel Price.
3. See the prices.
4. Chose the liked one.</td><td>Click on the showing details like name of the pump and price or search by name of the pump.</td><td>Show the full details.</td><td>As expected,</td><td>Pass</td></tr>
<tr><td colspan="5">Post Condition: User is validated with database and successfully show the details. The searched data details are saved in the database.</td></tr>
</table>

Table 6: Test Case for **On-Road Features**

Project Name: **Fuel Finder App**	Test Designed by: MD. NAHIDUL ISLAM
Test Case ID: FR_6	Test Designed date:20/09/2024
Test Priority (Low, Medium, High): Medium	Test Executed by: MD. Nahidul Islam
Module Name: On-Road Features	Test Execution date:20/09/2024
Test Title: Find the best route for pump.	
Description: Give access the app to find the nearest pump.	
Precondition (If any): User has entered current fuel level and optionally, a planned route.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Enter into the app 2. Click on the option bar and find nearest pump. 3. Give access for location. 4. Select the pump and starting to go.	Location: Badda fuel pump Access: Allow	User would find the nearest fuel pump	Not expected	Fail

Post Condition: After find the desire location user can deny access of his phone for location.

Table 7: Test Case for **Security and Privacy**

Project Name: **Fuel Finder App**	Test Designed by: MD. NAHIDUL ISLAM
Test Case ID: FR_7	Test Designed date:20/09/2024
Test Priority (Low, Medium, High): High	Test Executed by: MD. Nahidul Islam
Module Name: Security and Privacy	Test Execution date:20/09/2024
Test Title: Check security cotext	
Description: App will not share anything with third party	
Precondition (If any): User creates an account or log in with existing credentials. User is prompted to allow location access. User can grant or denies location access.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Enter into the app 2. Do searching or viewing 3. Access the location while using the apps. Also can do messaging on the support system . Can save personal choisable fuel pumps.	Messaging on support system, allow location access and save something.	After closing the app the location access will deny, the save details will securely handled. User name or password never be shared with anyone. Everything will be secure.	Not expected	Fail
Post Condition: User is secured with their data privacy.				

Table 8: Test Case for **Feedback and Support**

<table>
<tr><td colspan="3">Project Name: <b>Fuel Finder App</b></td><td colspan="2">Test Designed by: MD. NAHIDUL ISLAM</td></tr>
<tr><td colspan="3">Test Case ID: FR_8</td><td colspan="2">Test Designed date:20/09/2024</td></tr>
<tr><td colspan="3">Test Priority (Low, Medium, High): Medium</td><td colspan="2">Test Executed by: MD. Nahidul Islam</td></tr>
<tr><td colspan="3">Module Name: <b>Feedback and Support</b></td><td colspan="2">Test Execution date:20/09/2024</td></tr>
<tr><td colspan="3">Test Title: verify login with valid username and password</td><td colspan="2"></td></tr>
<tr><td colspan="3">Description: Test website login page</td><td colspan="2"></td></tr>
<tr><td colspan="5">Precondition (If any): User must have valid username and password</td></tr>
<tr><th>Test Steps</th><th>Test Data</th><th>Expected Results</th><th>Actual Results</th><th>Status (Pass/Fail)</th></tr>
<tr><td>1. Open Rating and Review: Go to Feedback & Support, then click "Rating and Review."
2. Rate and Review: Choose a star rating and write a detailed review.
3. Submit: Click "Post" to submit your review</td><td>User selects a 4-star rating.
User enters the review text: "The app is user-friendly and efficient."</td><td>The user should be able to successfully submit the rating and review.
The rating and review are recorded in the app's database.</td><td>As expected</td><td>Pass</td></tr>
<tr><td colspan="5">Post Condition: Despite the failure, verify that the rating and review are recorded in the app's database. Log any error messages encountered during the test for further investigation.</td></tr>
</table>

Table 9: Test Case for **Verify Fuel Booking Process**

<table>
<tr><td colspan="3">Project Name: <b>Fuel Finder App</b></td><td colspan="2">Test Designed by: Moumita Bhowmik</td></tr>
<tr><td colspan="3">Test Case ID: FR_9</td><td colspan="2">Test Designed date: 20/09/2024</td></tr>
<tr><td colspan="3">Test Priority (Low, Medium, High): High</td><td colspan="2">Test Executed by: Moumita Bhowmik</td></tr>
<tr><td colspan="3">Module Name: Booking</td><td colspan="2">Test Execution date: 21 /09/2024</td></tr>
<tr><td colspan="5">Test Title: Verify Fuel Booking Process</td></tr>
<tr><td colspan="5">Description: Test the process of booking fuel .</td></tr>
<tr><td colspan="5">Precondition (If any): User must have an registered account.</td></tr>
<tr><td><b>Test Steps</b></td><td><b>Test Data</b></td><td><b>Expected Results</b></td><td><b>Actual Results</b></td><td><b>Status (Pass/Fail)</b></td></tr>
<tr><td>1. Enter valid name,username, email, and password.
2. After successfully logging in, navigate to the "Booking" section.
3. Select the type of fuel you want to book (e.g., gasoline, diesel, natural gas).
4. Enter the quantity of fuel you want to order (e.g., 10 gallons).
5. Choose the delivery location or fuel station.
6. Select the payment method (e.g., credit card, PayPal).
7. Click on the “Confirm Booking" button.</td><td>Username: abid
Password: Abc12345</td><td>1. The app should confirm the fuel booking.
2. A confirmation message should be displayed.
3. The user should receive an email or notification with the booking details.</td><td>Not expected</td><td>Fail</td></tr>
<tr><td colspan="5">Post Condition: The user's fuel booking is successfully processed, and the booking details are saved in the FuelFinder database.</td></tr>
</table>

Table 10: Test Case for **Payment Method**

Project Name: **Fuel Finder App**	Test Designed by: Moumita Bhowmik
Test Case ID: FR-10	Test Designed date: 20/09/2024
Test Priority (Low, Medium, High): High	Test Executed by: Moumita Bhowmik
Module Name: Payment Method	Test Execution date: 21/09/2024
Test Title: Verify payment method with phone number and password	
Description: Test payment method process	
Precondition (If any): The user is on the payment page and must be logged in.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to the payment option. 2. Click on the select button. 3. Choose a payment method. 4. Enter the amount. 5. Enter valid phone number and password. 6. Click submit.	Phone number: 0123456789 Password: 123	Payment is successful.	Not expected,	Fail
Post Condition: Payment must be recorded.				

Table 11: Test Case for **Reviews and Rating**

Project Name: **Fuel Finder App**	Test Designed by: Moumita Bhowmik
Test Case ID: FR_11	Test Designed date: 21/09/24
Test Priority (Low, Medium, High): Medium	Test Executed by: Moumita Bhowmik
Module Name: Reviews and Ratings	Test Execution date: 21/09/24
Test Title: Verify user can submit a review and rating for a fuel station	
Description: Test the user review and rating functionality on the FuelFinder app.	
Precondition (If any): The user needs a valid account to put up a review or rating.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to the stations's detail page. 2. Click on the "add a review" button. 3. Enter a review text. 4. Enter Select a star rating (1 to 5). 5. Click submit.	Station ID:1234 "Great service!" Rating: 4 stars.	User is able to view details of the fuel station. User is directed to the review submission form. Review text is input correctly in the text area. User's selected star rating is registered. Review and rating are submitted successfully.	As expected,	Pass
Post Condition: The user's review and rating are stored in the database, the review is displayed on the station's page, the rating is included in the station's average rating calculation.				

Table 12: Test Case for **Filter and Sort Options**

Project Name: **Fuel Finder App**	Test Designed by: Moumita Bhowmik
Test Case ID: FR_12	Test Designed date:21/09/2024
Test Priority (Low, Medium, High): Medium	Test Executed by: Moumita Bhowmik
Module Name: Filter and Sort Options	Test Execution date: 21/09/2024
Test Title: Verify filter and sort functionality for fuel stations	
Description: Test the filter and sort options provided for searching fuel stations.	
Precondition (If any): The user must have access to the app and the location services are available to him.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to the FuelFinder app. 2. Access the search bar and search for fuel stations. 3. Apply filter: Set Price Range (e.g., $2.5 - $3.5) 4. Apply filter: Set Fuel Type (e.g., Diesel) 5. Sort the results by Price (Ascending)	Price Range = $2.5 - $3.5 Fuel Type = Diesel	The app home screen should load successfully. The app should display search results based on the user's location or general criteria. The results should be filtered to show only stations within the selected price range. The app should filter results to show stations that have Diesel available. Results should be sorted with the lowest price fuel stations appearing first. The results should reflect both the selected price range and distance. The page should not reload fully. Results should refresh dynamically The system should show all search results without any filters applied.	As expected,	Pass

6. Apply multiple filters (Price Range + Distance) 7. Check for page refresh after applying filter 8. Clear all filters	Sort by = Price (Ascending) Price Range = $2.5 - $3.5; Distance = < 5 miles Apply any filter (e.g., Fuel Type = Gasoline)			
Post Condition: The user successfully applies filters and sorting options without any issues, the filtered and sorted results are displayed as per the applied options without full page reloads.				