

Deliverable 4 (Due on Thursday 2/5/2024) (Team Submissions)

Shayan Ahmad, Matija Susic, Farzan Ali

Section 1: Introduction

1.1 Test Project Name: Abu Dhabi Eats

1.2 Summary of the Rest of the Test Plan: Provide a brief overview of the subsequent sections of the testing report.

This testing report comprehensively covers the strategies and methodologies employed to ensure the functionality, reliability, and usability of the "Abu Dhabi Eats" application. Each section of this report is dedicated to a detailed aspect of our testing efforts as follows:

- **Test Strategy Overview:** This section discusses our approach to validating each functionality of the application, including user profile creation, meal generation, meal plan scheduling, nutritional intake tracking, and health goal adjustments. We outline the combination of black box testing, system integration testing, and user acceptance testing to ensure comprehensive coverage.
- **Test Scenarios and Cases:** Detailed descriptions of test cases for all major features are provided, with each case linked to specific use cases to ensure functional correctness and user satisfaction. This section details the scenarios tested, expected results, and special attention to edge cases and alternative paths.
- **Test Environment and Tools:** Description of the software, hardware, and network environments used during the testing phases. We also list the tools and technologies employed for automated testing, issue tracking, and version control.
- **Test Results and Analysis:** Summarization of the test outcomes, including the number of tests passed, failed, and any discrepancies from expected results. This section includes an analysis of potential causes for failures and documentation of bugs with their severity levels.
- **Quality Assurance Metrics:** Presentation of quantitative metrics used to measure the quality of the application, such as test coverage percentage, bug density, and mean time to resolution for identified issues.

- **Mitigation and Recommendations:** Discussion on the mitigative strategies for any significant risks identified during the testing phases, along with recommendations for future testing cycles and project enhancements.

This structured approach ensures that every feature of "Abu Dhabi Eats" is rigorously tested and validated to meet our high standards of quality and performance, guaranteeing a reliable and user-friendly experience for all end-users.

Section 2: Feature Description

Describe the final use cases that you are testing in detail.

1. “Create profile”

Main Success Scenario:

1. User selects "Register."
2. User fills in personal details, including email and password.
3. User inputs dietary preferences and health goals.
4. System validates and saves the information.
5. System confirms account creation.

Alternative Scenarios:

2. a) If the email is already in use, the system prompts for another email.
3. a) If the user exits without completing the profile, the system saves the information that the user already entered.
3. b) If dietary preferences are incomplete, the system requests more information.
4. a) If there's an error in saving user preferences, an error message is displayed.

2. “Generate meal”

Main Success Scenario:

1. User initiates “Generate Meal.”
2. System recommends three distinct meal options to the user based on their profile and past dining history.
3. User selects one of the three recommended meals.
4. System offers a seamless transition to an existing food delivery service for ordering.

Alternative Scenarios:

3. a) If the recommended meals are not satisfactory, the user requests three new options.
4. a) If there's an issue with a particular food delivery service, the system provides alternative services.

3. “Schedule Meal Plan”

Main Success Scenario:

- 1) User navigates to the "Schedule Meal Plans".
- 2) User selects meal options for each day up to a week in advance.
- 3) System calls the recommendation algorithm to assist the user in selecting meals.
- 4) User confirms the meal plan schedule.
- 5) System sends reminder notifications to the user for meals based on their scheduled meal plans.

Alternative Scenarios:

5. a) If the user deviates from the schedule, the system prompts the user to remake the schedule (repeating steps 2-5).

4. “Track Nutritional Intake”

Main success scenario:

1. The system logs each meal consumed from the generated meal plan with complete nutritional information.
2. System updates the user's nutritional intake dashboard in real-time, showing calories, macros, and micronutrients.
3. Provides weekly summaries and insights into eating habits, with suggestions for Improvement.

Alternative Scenarios:

- 1a. When having a meal not recommended by the system, the user can manually log the meal information to accurately track nutrition intake.

5. “Adjust Health Goals and Preferences”

Main Success Scenario:

1. User accesses profile settings.
2. User selects the option to update dietary preferences or health goals.
3. System displays current settings and offers editable fields.
4. User submits changes.
5. System updates profile and confirms changes.

Extensions:

- 4a. If new preferences conflict with existing meal plans, offer to regenerate plans.

Section 3: Assumptions

3.1 Test Case Exclusions: List any test cases or scenarios that are not included in the testing.

Because we thought the following test cases are somewhat repetitive to what we've decided to include, we decided to omit them from our "official" test cases.

Omitted Test Case 1

Test Case Title: Validate Profile Creation with Valid Data

Use Case Tested by the Test Case: Create Profile

Technique Used: Equivalence Partitioning

Pre-condition: User is not logged in or registered

Input: Valid email, password, dietary preferences, and health goals

Steps to Execute:

1. Navigate to the registration page.
2. Input all required fields with valid data.
3. Submit the registration form.

Expected Results: User receives a confirmation message, and a new profile is created in the system.

Omitted Test Case 2

Test Case Title: Validate Profile Creation with Duplicate Email

Use Case Tested by the Test Case: Create Profile

Technique Used: Boundary Value Analysis

Pre-condition: A user with the email already exists in the system

Input: Duplicate email, valid password, dietary preferences, and health goals

Steps to Execute:

1. Navigate to the registration page.
2. Input all required fields using an email already in use.
3. Attempt to submit the registration form.

Expected Results: The system prevents registration and prompts the user to use a different email.

3.2 Test Tool: List the testing tools employed for the project.

Section 4: Test Approach

4.1 Test Strategy: Explain the testing techniques (e.g., Black Box, White Box, Exploratory Testing, System Testing) used to evaluate different system components.

Test Strategy For "Abu Dhabi Eats," : we employ a straightforward and effective testing strategy that combines Black Box Testing and Exploratory Testing methods. This approach is tailored to validate the functionalities of the system through user-driven scenarios and to ensure that the application behaves as expected in real-world usage conditions.

- **Black Box Testing:** This technique is primarily used to assess the functional aspects of the application without delving into the internal workings of the system. It focuses on testing the user interface and interactions with the system as perceived by end-users. Key areas include user registration, meal generation, and dietary preference updates. This approach helps in verifying that the system meets the specified requirements from a user's perspective.
- **Exploratory Testing:** Due to the time constraints and ongoing development, exploratory testing plays a critical role. It allows testers to use the application in ways that typical users might, which helps in identifying unexpected behaviors or bugs. It is particularly useful for assessing new features like meal planning and tracking nutritional intake that may not yet have comprehensive test cases defined.

4.2 Provide a rationale for your chosen approach. Include details on the tools, automation, and scripts used for testing each system part.

The decision to limit the testing approach to Black Box and Exploratory Testing methods stems from the need to efficiently validate user-facing features while accommodating the rapid pace of development and the project's deadline constraints. This strategy ensures that testing remains flexible and responsive to the project's evolving nature.

Tools:

- **Google Forms and Spreadsheets:** Used for organizing and executing Black Box tests. Test cases are documented in a spreadsheet, and Google Forms are used to collect feedback from testers during exploratory sessions.
- **Browser Developer Tools:** Utilized to quickly diagnose and address issues related to the user interface, such as CSS styling, JavaScript functionality, and responsive design.
- **Manual Testing:** Due to the project's scope and timeline, manual testing is a practical choice. It allows the development team to immediately verify fixes and changes without the overhead of complex automation tools.

The selected tools and techniques are conducive to a fast-paced development environment where features are being finalized and refined. They provide the necessary flexibility to adapt testing as the product evolves, ensuring that the team can focus on delivering a reliable and user-friendly application within the allotted timeframe.

By concentrating on practical and adaptable testing methods, "Abu Dhabi Eats" aims to maintain a high standard of quality, ensuring that the primary functionalities are robust and meet user expectations, even under the tight deadline.

Section 5: Test Cases

5.1 Test Cases: List all test cases. It is preferable if you group your test cases following any categorization scheme you may choose (E.g. test cases related to features, security, performance, etc.)

5.1.1 Category - Test cases related to Profile Creation

1.

Test Case Title: Password Length Less Than 8

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "abc123!". Password length less than 8 characters.

Steps to Execute:

- User selects "Register."
- User fills in personal details, including email and password with length less than 8 characters.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System successfully creates the account.

2.

Test Case Title: Password Length Between 8 and 16

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "abcd1234567!". Password length between 8 and 16 characters.

Steps to Execute:

- User selects "Register."
- User fills in personal details, including email and password with length between 8 and 16 characters.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should successfully create the account.

3.

Test Case Title: Password Length Greater Than 16

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "abcdefgh123457890!". Password length greater than 16 characters.

Steps to Execute:

- User selects "Register."

- User fills in personal details, including email and password with length greater than 16 characters.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should successfully create the account.

4.

Test Case Title: Nominal Height, Minimum Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 165cm (nominal value), Weight = 30kg (minimum value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length nominal and minimum values.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle minimum weight input with nominal height and successfully create the profile.

5.

Test Case Title: Nominal Height, Almost Minimum Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 165cm (nominal value), Weight = 31kg (min+ or almost minimum value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle almost minimum weight input with nominal height.

6.

Test Case Title: Nominal Height, Nominal Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 165cm (nominal value), Weight = 62kg (nom or average value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.

- System validates and saves the information.

Expected Results: System should handle nominal weight input with nominal height.

7.

Test Case Title: Nominal Height, Almost Maximum Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 165cm (nominal value), Weight = (400 - 1)kg (almost maximum value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.

System validates and saves the information.

Expected Results: System should handle almost maximum weight input with nominal height.

8.

Test Case Title: Nominal Height, Maximum Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 165cm (nominal value), Weight = 400kg (max)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle maximum weight input with nominal height.

9.

Test Case Title: Minimum Height, Nominal Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 60cm (minimum value), Weight = 62kg (nom or average value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle minimum height input with nominal weight.

10.

Test Case Title: Almost Minimum Height, Nominal Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 61cm (almost minimum value), Weight = 62kg (nom or average value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle almost minimum height input with nominal weight.

11.

Test Case Title: Almost Maximum Height, Nominal Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = (270-1cm (almost maximum value), Weight = 62kg (nom or average value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System should handle almost maximum height input with nominal weight.

12.

Test Case Title: Maximum Height, Nominal Weight

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Boundary Value Testing

Pre-condition: N/A

Input: Height = 270cm (maximum value), Weight = 62kg (nom or average value)

Steps to Execute:

- User selects "Create Profile."
- User inputs personal details, including height and weight with length specified values above.
- User inputs dietary preferences and health goals.

System validates and saves the information.

Expected Results: System should handle maximum height input with nominal weight.

13.

Test Case Title: Weight Less Than 30kg

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "29". Weight value of less than 30kgs.

Steps to Execute:

- User selects "Register."
- User fills in personal details, including weight with value less than 30kg.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System successfully creates the account.

14.

Test Case Title: Weight Between 30 and 400kg

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "100". Weight value is between 30 and 400kg.

Steps to Execute:

- User selects "Register."
- User fills in personal details, including weight with value between 30 and 400kg.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System successfully creates the account.

15.

Test Case Title: Weight More Than 400kg

Use Case Tested by the Test Case: "Create Profile"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: "401". Weight value of more than 400kgs.

Steps to Execute:

- User selects "Register."
- User fills in personal details, including weight with value more than 400kg.
- User inputs dietary preferences and health goals.
- System validates and saves the information.

Expected Results: System successfully creates the account.

16.

Test Case Title: Generate Meal with Dietary Restriction

Use Case Tested by the Test Case: Generate Meal

Technique Used: Equivalence Partitioning

Pre-condition: User profile is set with a dietary restriction (e.g., Gluten-Free)

Input: User requests a meal generation

Steps to Execute:

1. Navigate to the "Generate Meal" feature.
2. Request meal generation.

3. Review the generated meal options.

Expected Results: All generated meal options comply with the Gluten-Free dietary Restriction.

17.

Test Case Title: Meal Generation for Vegetarian Diet

Use Case Tested by the Test Case: Generate Meal

Technique Used: Equivalence Partitioning

Pre-condition: User's dietary preference is set to "Vegetarian" in their profile.

Input: Request to generate a meal.

Steps to Execute:

1. User selects the "Generate Meal" option.
2. System retrieves the user's dietary preferences.
3. System generates meal options based on the "Vegetarian" diet.

Expected Results: The system presents only vegetarian meal options.

18.

Test Case Title: Scheduling Meal Plan with Existing Allergies

Use Case Tested by the Test Case: Schedule Meal Plan

Technique Used: Boundary Value Analysis

Pre-condition: User profile includes a known allergy (e.g., peanuts).

Input: User schedules a meal plan for the upcoming week.

Steps to Execute:

1. User navigates to "Schedule Meal Plans."
2. User selects meal preferences, excluding any dishes with peanuts.
3. System schedules meals for the week, avoiding the allergen.

Expected Results: The scheduled meal plan for the week contains zero meals with peanuts.

19.

Test Case Title: Tracking Nutritional Intake Post-Meal Update

Use Case Tested by the Test Case: Track Nutritional Intake

Technique Used: Pairwise testing

Pre-condition: User has a meal plan scheduled for the current day.

Input: User manually logs a meal not in the meal plan.

Steps to Execute:

1. User accesses the "Nutritional Intake" feature.
2. User adds a meal that was consumed outside the scheduled meal plan.
3. System updates the nutritional dashboard with the new meal's info.

Expected Results: The dashboard accurately reflects the nutritional information of the manually logged meal, alongside the scheduled meals.

20.

Test Case Title: Validate Meal Option Regeneration on Dissatisfaction

Use Case Tested by the Test Case: Generate Meal

Technique Used: Equivalence Partitioning

Pre-condition: User has expressed dissatisfaction with current meal options.

Input: User requests new meal options.

Steps to Execute:

1. User indicates dissatisfaction with presented meal options.
2. User requests new meal options.
3. System generates a new set of meal options.

Expected Results: The system provides three alternative meal options different from the initial set

21.

Test Case Title: Meal Plan Rescheduling Upon User Deviation

Use Case Tested by the Test Case: Schedule Meal Plan

Technique Used: Equivalence Partitioning

Pre-condition: User deviates from the previously scheduled meal plan.

Input: User initiates a rescheduling request.

Steps to Execute:

1. User accesses the "Meal Plan" feature.
2. User selects the option to reschedule their meal plan.
3. User reselects meal options for the schedule.
4. System confirms the new meal plan.

Expected Results: The system updates the user's meal schedule and confirms the changes without system errors or data persistence issues.

22.

Test Case Title: Handling Conflicting Dietary Changes

Use Case Tested by the Test Case: Adjust Health Goals and Preferences

Technique Used: Boundary Value Analysis

Pre-condition: User's current meal plan is set based on previous dietary preferences.

Input: User updates dietary preferences which conflict with the existing meal plan.

Steps to Execute:

1. User opens "Health Goals and Preferences" settings.
2. User updates dietary preferences to a conflicting choice.
3. System detects conflict with existing meal plans.
4. System prompts the user to accept meal plan regeneration.
5. User accepts.

Expected Results: The system offers to regenerate the meal plan, and upon user acceptance, a new meal plan that aligns with updated preferences is created.

23.

Test Case Title: Profile Update with Invalid Email Format

Use Case Tested by the Test Case: Create Profile

Technique Used: Boundary Value Analysis

Pre-condition: User is on the registration page.

Input: Invalid email format (e.g., "user@domain"), valid password, dietary preferences, and health goals.

Steps to Execute:

1. Input an invalid email format in the email field.
2. Complete the remaining fields with valid information.
3. Submit the registration form.

Expected Results: System displays an error message indicating the email format is incorrect.

24.

Test Case Title: Generate Meal with Caloric Restriction Boundary

Use Case Tested by the Test Case: Generate Meal

Technique Used: Boundary Value Analysis

Pre-condition: User has a caloric goal set near a boundary value (e.g., 2000 calories).

Input: Request to generate a meal with specific caloric requirements.

Steps to Execute:

1. User selects the "Generate Meal" option with a set caloric restriction.
2. System retrieves the user's dietary preferences and caloric goal.
3. System generates meal options near the caloric limit.

Expected Results: Generated meal options are within the caloric boundary, neither exceeding nor falling short by a significant amount.

25.

Test Case Title: Scheduling Meal Plan with Maximum Meal Entries

Use Case Tested by the Test Case: Schedule Meal Plan

Technique Used: Boundary Value Analysis

Pre-condition: User attempts to schedule the maximum number of meals for a day.

Input: User schedules meals up to the maximum allowed per day.

Steps to Execute:

1. User navigates to "Schedule Meal Plans."
2. User adds the maximum number of meals for each day, up to a week in advance.
3. System processes and saves the meal entries.

Expected Results: The system allows scheduling up to the maximum limit and provides a confirmation message once the plan is successfully saved.

26.

Test Case Title: Manual Meal Log with Incomplete Nutritional Data

Use Case Tested by the Test Case: Track Nutritional Intake

Technique Used: Equivalence Partitioning

Pre-condition: User is logged in and has consumed a meal not part of the generated meal plan.

Input: Partial nutritional information for a manually logged meal.

Steps to Execute:

1. Access the "Nutritional Intake" tracking feature.
2. Manually log a new meal with incomplete nutritional data.
3. Submit the meal log entry.

Expected Results: The system alerts the user to provide complete nutritional information or fills in estimated values based on similar meals.

27.

Test Case Title: Health Goal Update with Valid New Goals

Use Case Tested by the Test Case: Adjust Health Goals and Preferences

Technique Used: Equivalence Partitioning

Pre-condition: User has existing health goals set.

Input: Valid new health goals that do not conflict with dietary preferences.

Steps to Execute:

1. Open "Health Goals and Preferences" in the profile settings.
2. Enter new health goals within the valid range.
3. Submit the updated goals.

Expected Results: The system updates the health goals in the user profile and confirms the update to the user without offering meal plan regeneration.

28.

Test Case Title: Regenerate Meal Plan after Dietary Preference Update

Use Case Tested by the Test Case: Adjust Health Goals and Preferences

Technique Used: Truth Table

Pre-condition: User has existing meal plans based on old dietary preferences.

Input: Updated dietary preferences that conflict with existing meal plans.

Steps to Execute:

1. Open "Health Goals and Preferences" settings.
2. Update dietary preferences to conflicting choices.
3. System detects conflicts with existing meal plans.
4. User chooses to regenerate meal plans based on new preferences.
5. Submit the changes and accept the meal plan regeneration.

Expected Results: The system provides the user with new meal plan options that match the updated dietary preferences and schedules them accordingly.

29.

Test Case Title: Schedule Meal Plan for Negative Days

Use Case Tested by the Test Case: "Schedule Meal Plan"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: Number of days = -1

Steps to Execute:

- User navigates to the "Schedule Meal Plans."
- User attempts to select meal options for -1 days.

Expected Results: System should display an error message indicating that the number of days cannot be negative.

30.

Test Case Title: Schedule Meal Plan for 7 Days

Use Case Tested by the Test Case: "Schedule Meal Plan"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: Number of days = 7

Steps to Execute:

- User navigates to the "Schedule Meal Plans."
- User selects meal options for 7 days.

Expected Results: System should successfully allow the user to schedule meal options for the next 7 days.

31.

Test Case Title: Schedule Meal Plan for 10000 Days

Use Case Tested by the Test Case: "Schedule Meal Plan"

Technique Used to generate test case with a proof: Equivalence Partitioning

Pre-condition: N/A

Input: Number of days = 10000

Steps to Execute:

- User navigates to the "Schedule Meal Plans."
- User attempts to select meal options for 10000 days.

Expected Results: System should display an error message indicating that scheduling a meal plan for such a larger number of days than 7 is not supported.

32.

Test Case Title: Adjusting Conflicting Health Goals

Use Case Tested by the Test Case: "Adjust Health Goals and Preferences"

Technique Used to generate test case with a proof: Exploratory Testing

Pre-condition: User accesses profile settings.

Input: User selects conflicting health goals (weight loss, maintaining, and muscle gain).

Steps to Execute:

- User navigates to the profile settings.
- User selects the option to update health goals.
- User chooses conflicting health goals weight loss, maintaining, and muscle gain.
- User submits the changes.

Expected Results:

System should detect conflicting health goals and provide a warning message to the user.

The warning message should inform the user about the conflicting nature of the selected goals and suggest reevaluating them.

User should be given the option to revise the goals.

33. Test Case Title: Email Format Verification

- **Use Case Tested by the Test Case:** "Create Profile"
- **Technique Used to generate test case with a proof:** Equivalence Partitioning
- **Pre-condition:** User is on the registration page.
- **Input:** "user@domain.com". Correct email format.
- **Steps to Execute:**
 1. User selects "Register."
 2. User inputs "user@domain.com" as the email address.
 3. User completes the rest of the registration form.
 4. System processes and verifies the information.
- **Expected Results:** System accepts the email and successfully processes the registration.

34. Test Case Title: Invalid Email Format

- **Use Case Tested by the Test Case:** "Create Profile"
- **Technique Used to generate test case with a proof:** Equivalence Partitioning
- **Pre-condition:** User is on the registration page.
- **Input:** "useratdomaincom". Missing '@' and '.' in the email address.
- **Steps to Execute:**
 1. User selects "Register."
 2. User inputs "useratdomaincom" as the email address.
 3. User completes the rest of the registration form.
 4. System processes and verifies the information.
- **Expected Results:** System identifies the email format as incorrect and prompts the user to correct it.

35.

Test Case Title: Meal Preference Update Confirmation

- **Use Case Tested by the Test Case:** "Adjust Health Goals and Preferences"
- **Technique Used to generate test case with a proof:** State Transition Testing
- **Pre-condition:** User is logged in and navigates to the profile settings.
- **Input:** User updates their meal preference from "Vegetarian" to "Vegan".
- **Steps to Execute:**
 1. User accesses the "Health Goals and Preferences" settings.
 2. User selects the option to change dietary preferences from "Vegetarian" to "Vegan".
 3. User submits the changes.
 4. System updates the preferences and provides a confirmation.
- **Expected Results:** The system should confirm the update and display the new preference as "Vegan".

36.

Test Case Title: Schedule Conflict Resolution

- **Use Case Tested by the Test Case:** "Schedule Meal Plan"
- **Technique Used to generate test case with a proof:** Exploratory Testing
- **Pre-condition:** User has already scheduled meals for the week but decides to change one meal due to an unexpected schedule change.
- **Input:** User tries to schedule a new meal that conflicts with an existing meal time.
- **Steps to Execute:**
 1. User navigates to the "Schedule Meal Plans" section.
 2. User selects a new meal for a time slot that already has a scheduled meal.
 3. System detects the conflict and prompts the user to resolve it.
 4. User chooses to replace the existing meal with the new selection.
 5. System updates the meal plan and confirms the change.
- **Expected Results:** The system should handle the scheduling conflict gracefully, allowing the user to resolve it and confirm the updated schedule without errors.

37.

Test Case Title: Invalid Dietary Restriction Submission

- **Use Case Tested by the Test Case:** "Create Profile"
- **Technique Used to generate test case with a proof:** Negative Testing
- **Pre-condition:** User is on the profile creation page.
- **Input:** User selects a dietary restriction that conflicts with other selected options (e.g., selects both 'Vegan' and 'Seafood').
- **Steps to Execute:**
 1. User selects "Register."
 2. User fills in personal details, including conflicting dietary restrictions.
 3. User attempts to submit the profile.
 4. System validates the information and identifies the conflict.
- **Expected Results:** System displays an error message regarding the dietary restriction conflict and asks the user to correct their selections.

38.

Test Case Title: Seamless Transition to Updated Meal Plan

- **Use Case Tested by the Test Case:** "Adjust Health Goals and Preferences"
- **Technique Used to generate test case with a proof:** Exploratory Testing
- **Pre-condition:** User has an active meal plan and decides to update their health goals which affect the meal plan.
- **Input:** User updates health goals from 'maintenance' to 'weight loss'.
- **Steps to Execute:**
 1. User navigates to "Health Goals and Preferences."

2. User updates their health goal to 'weight loss'.
 3. System prompts the user to update their meal plan based on new goals.
 4. User accepts the update.
 5. System seamlessly updates the meal plan and confirms the changes.
- **Expected Results:** System should automatically update the meal plan to reflect the new health goals and confirm the update without errors, ensuring user satisfaction

39.

Test Case Title: Real-Time Notification for Meal Plan Reminders

- **Use Case Tested by the Test Case:** "Schedule Meal Plan"
- **Technique Used to generate test case with a proof:** Exploratory Testing
- **Pre-condition:** User has scheduled a meal plan with reminders enabled.
- **Input:** Time reaches the scheduled meal reminder time.
- **Steps to Execute:**
 1. User enables notifications for the meal plan.
 2. User continues normal application usage or locks the device.
 3. System checks the current time and triggers a notification when it matches a scheduled meal time.
 4. User receives a notification reminding them of the meal.
- **Expected Results:** The system should send timely notifications at the correct scheduled times, ensuring users are reminded of their meals as planned.

40.

Test Case Title: Data Encryption Validation for Sensitive Information

- **Use Case Tested by the Test Case:** "Create Profile"
- **Technique Used to generate test case with a proof:** Security Testing
- **Pre-condition:** The application requires encryption for storing and transmitting sensitive user information.
- **Input:** User registration with sensitive data (e.g., password, health information).
- **Steps to Execute:**
 1. User registers with sensitive information.
 2. Inspect the method of data storage and transmission via network tools.
- **Expected Results:** All sensitive user information should be encrypted using industry-standard encryption techniques both at rest and in transit. Verification via network tools should not reveal any plaintext sensitive data.

41.

Test Case Title: Time Zone Sensitivity for Meal Scheduling

- **Use Case Tested by the Test Case:** "Schedule Meal Plan"
- **Technique Used to generate test case with a proof:** Functional Testing
- **Pre-condition:** User is located in a different time zone from the server.
- **Input:** User schedules a meal plan while traveling across time zones.

- **Steps to Execute:**
 1. User adjusts their device time zone to a new location.
 2. User navigates to "Schedule Meal Plans" and schedules meals for the day.
 3. Check if the meal times adjust according to the new time zone.
- **Expected Results:** The system should accurately adjust meal times based on the user's current time zone, ensuring that meal reminders are relevant and timely regardless of geographical changes.

42.

Test Case Title: Meal Recommendation Refresh on Preference Change

- **Use Case Tested by the Test Case:** "Adjust Health Goals and Preferences"
- **Technique Used to generate test case with a proof:** Functional Testing
- **Pre-condition:** User has previously set preferences and received meal recommendations.
- **Input:** User updates their dietary preferences to exclude dairy.
- **Steps to Execute:**
 1. User navigates to profile settings.
 2. User updates dietary preferences to exclude dairy products.
 3. User navigates back to the meal recommendation page.
- **Expected Results:** The meal recommendations should refresh to exclude any meals containing dairy, reflecting the updated preferences immediately.

43.

Test Case Title: Invalid Login Attempt Monitoring

- **Use Case Tested by the Test Case:** "Create Profile"
- **Technique Used to generate test case with a proof:** Security Testing
- **Pre-condition:** User attempts to log in with incorrect credentials multiple times.
- **Input:** Incorrect password entered five times consecutively.
- **Steps to Execute:**
 1. User enters email and incorrect password.
 2. User attempts to log in five times.
 3. Observe system response after multiple failed attempts.
- **Expected Results:** The system should lock the user account temporarily or require additional verification to prevent unauthorized access.

44.

Test Case Title: Dynamic Nutritional Information Update

- **Use Case Tested by the Test Case:** "Track Nutritional Intake"
- **Technique Used to generate test case with a proof:** Integration Testing
- **Pre-condition:** User logs meals that include ingredients with variable nutritional values.
- **Input:** User logs a homemade sandwich with varying ingredient amounts.

- **Steps to Execute:**
 1. User enters the meal details including individual ingredients and their quantities.
 2. User submits the meal information.
 3. User checks the nutritional information dashboard.
- **Expected Results:** The nutritional dashboard updates in real-time to reflect the exact caloric and nutritional values based on the entered ingredients and their quantities.

45.

Test Case Title: Simultaneous Meal Planning by Multiple Users

- **Use Case Tested by the Test Case:** "Schedule Meal Plan"
- **Technique Used to generate test case with a proof:** Concurrency Testing
- **Pre-condition:** Multiple users are logged in and scheduling meal plans at the same time.
- **Input:** Each user schedules a weekly meal plan simultaneously.
- **Steps to Execute:**
 1. Multiple users access the "Schedule Meal Plans" feature.
 2. Each user selects meals for a week and submits their plans at the same time.
 3. System processes all requests without delay or error.
- **Expected Results:** The system should handle multiple simultaneous meal plan submissions efficiently, ensuring there are no delays, data losses, or errors.

5.2 Traceability Matrix: Develop a traceability matrix to correlate requirements (use cases) with test cases.

Test Case Number	Test Case Name	Use Case	Testing Technique Used	Passed/Failed/Inconclusive
1	Password Length Less Than 8	Create Profile	Equivalence Partitioning	p
2	Password Length Between 8 and 16	Create Profile	Equivalence Partitioning	p
3	Password Length Greater Than 16	Create Profile	Equivalence Partitioning	p
4	Nominal Height, Minimum Weight	Create Profile	Boundary Value Testing	p
5	Nominal Height, Almost Minimum Weight	Create Profile	Boundary Value Testing	p
6	Nominal Height, Nominal Weight	Create Profile	Boundary Value Testing	p

7	Nominal Height, Almost Maximum Weight	Create Profile	Boundary Value Testing	p
8	Nominal Height, Maximum Weight	Create Profile	Boundary Value Testing	p
9	Minimum Height, Nominal Weight	Create Profile	Boundary Value Testing	p
10	Almost Minimum Height, Nominal Weight	Create Profile	Boundary Value Testing	p
11	Almost Maximum Height, Nominal Weight	Create Profile	Boundary Value Testing	p
12	Maximum Height, Nominal Weight	Create Profile	Boundary Value Testing	p
13	Weight Less Than 30kg	Create Profile	Equivalence Partitioning	p
14	Weight Between 30 and 400kg	Create Profile	Equivalence Partitioning	p
15	Weight More Than 400kg	Create Profile	Equivalence Partitioning	p
16	Generate Meal with Dietary Restriction	Generate Meal	Equivalence Partitioning	p
17	Meal Generation for Vegetarian Diet	Generate Meal	Equivalence Partitioning	p
18	Scheduling Meal Plan with Existing Allergies	Schedule Meal Plan	Boundary Value Analysis	p
19	Tracking Nutritional Intake Post-Meal Update	Track Nutritional Intake	Pairwise testing	p
20	Validate Meal Option Regeneration on Dissatisfaction	Generate Meal	Equivalence Partitioning	p
21	Meal Plan Rescheduling Upon User Deviation	Schedule Meal Plan	Equivalence Partitioning	p
22	Handling Conflicting Dietary Changes	Adjust Health Goals and Preferences	Boundary Value Analysis	p
23	Profile Update with Invalid Email Format	Create Profile	Boundary Value Analysis	p
24	Generate Meal with Caloric Restriction Boundary	Generate Meal	Boundary Value Analysis	p
25	Scheduling Meal Plan with Maximum Meal Entries	Schedule Meal Plan	Boundary Value Analysis	p
26	Manual Meal Log with Incomplete Nutritional Data	Track Nutritional Intake	Equivalence Partitioning	p
27	Health Goal Update with Valid New Goals	Adjust Health Goals and Preferences	Equivalence Partitioning	p
28	Regenerate Meal Plan after Dietary Preference Update	Adjust Health Goals and	Truth Table	p

		Preferences		
29	Schedule Meal Plan for Negative Days	Schedule Meal Plan	Equivalence Partitioning	p
30	Schedule Meal Plan for 7 Days	Schedule Meal Plan	Equivalence Partitioning	p
31	Schedule Meal Plan for 10000 Days	Schedule Meal Plan	Equivalence Partitioning	p
32	Adjusting Conflicting Health Goals	Adjust Health Goals and Preferences	Exploratory Testing	f
33	Email Format Verification	Create Profile	Equivalence Partitioning	p
34	Invalid Email Format	Create Profile	Equivalence Partitioning	p
35	Meal Preference Update Confirmation	Adjust Health Goals and Preferences	State Transition Testing	p
36	Schedule Conflict Resolution	Schedule Meal Plan	Exploratory Testing	p
37	Invalid Dietary Restriction Submission	Create Profile	Negative Testing	p
38	Seamless Transition to Updated Meal Plan	Adjust Health Goals and Preferences	Exploratory Testing	p
39	Real-Time Notification for Meal Plan Reminders	Schedule Meal Plan	Exploratory Testing	p
40	Data Encryption Validation for Sensitive Information	Create Profile	Security Testing	p
41	Time Zone Sensitivity for Meal Scheduling	Schedule Meal Plan	Functional Testing	p
42	Meal Recommendation Refresh on Preference Change	Adjust Health Goals and Preferences	Functional Testing	p
43	Invalid Login Attempt Monitoring	Create Profile	Security Testing	p
44	Dynamic Nutritional Information Update	Track Nutritional Intake	Integration Testing	p
45	Simultaneous Meal Planning by Multiple Users	Schedule Meal Plan	Concurrency Testing	p

Section 6: Testing Results

Offer a summary of the testing outcomes, highlighting what test cases passed, failed, and unresolved issues.

Test cases 1-3 passed. When the password was not of sufficient length, an error message was

displayed indicating that the password must be of length at least 8 (as shown in the screenshot bellow), when the password was too long, the system warned the user in a similar fashion, and when the password was of acceptable length the system managed to create the profile.

Abu Dhabi Eats

Almost there!
Create your account.

ms14012@nyu.edu

☒ I agree to the [Terms and Conditions](#) and [Privacy Policy](#).

Continue

Activate Windows
Go to Settings to activate Windows.

Test cases 4-12 passed. When the heights or weights of valid value, the system successfully created the profile.

Abu Dhabi Eats

Congratulations!

Your daily net calorie goal is:

1,620
calories

With this plan, you should:
Lose 10 lbs by July 11

☒ [Sign up for emails](#)

Get nutrition from Abu Dhabi Eats. Plus, a first look at new features!

EXPLORE ABU DHABI EATS

Activate Windows
Go to Settings to activate Windows.

Test cases 13-15 passed. When the weight was of invalid value, the system displayed an error message (in the screenshot bellow), and when the weight was of a valid value, the system

successfully created the profile.

Enter Your Info

How tall are you?

170



How much do you weigh?

401



It's OK to es



Must be between 30 and 400 kg

What's your goal weight?

100

Don't worry. This doesn't affect your daily calorie goal and you can always change it later.

Please select which sex we should use to calculate your calorie needs.

☒ Male ☐ Female

When were you born?

03 - Jun - 2005



BACK

NEXT

Daily plan for gluten free successfully generated (Test case 16):

Breakfast	Classic Eggs Benedict (Tawa Gluten Free Eatery) (400)
	Protein: 20g, Fat: 20g, Carbs: 20g
	Tawa Gluten Free Eatery
Lunch	Chicken Tikka Biryani (Behrouz Biryani - Reem Island) (550)
	Protein: 34g, Fat: 10g, Carbs: 83g
	Behrouz Biryani - Reem Island
Dinner	Mixed Souvlaki Bowl (Go! Greek) (590)
	Protein: 35g, Fat: 32g, Carbs: 30g
	Go! Greek
Total	1540 calories, 89g protein, 62g fat, 133g carbs

Daily plan for vegetarian successfully generated (Test Case 17):

Breakfast	Veg Chowmein (Faasos - Wraps & Rolls - Reem Island) (400)
	Protein: 10g, Fat: 10g, Carbs: 70g
	Faasos - Wraps & Rolls - Reem Island
Lunch	2- Veggie Shack Burger Box(Shake Shack - Galleria Mall) (650)
	Protein: 25g, Fat: 35g, Carbs: 65g
	Shake Shack - Galleria Mall
Dinner	3- Baked Falafel Platter (Alkalime Restaurant - Saadiyat) (500)
	Protein: 20g, Fat: 25g, Carbs: 18g
	Alkalime Restaurant - Saadiyat
Total	1550 calories, 55g protein, 70g fat, 153g carbs

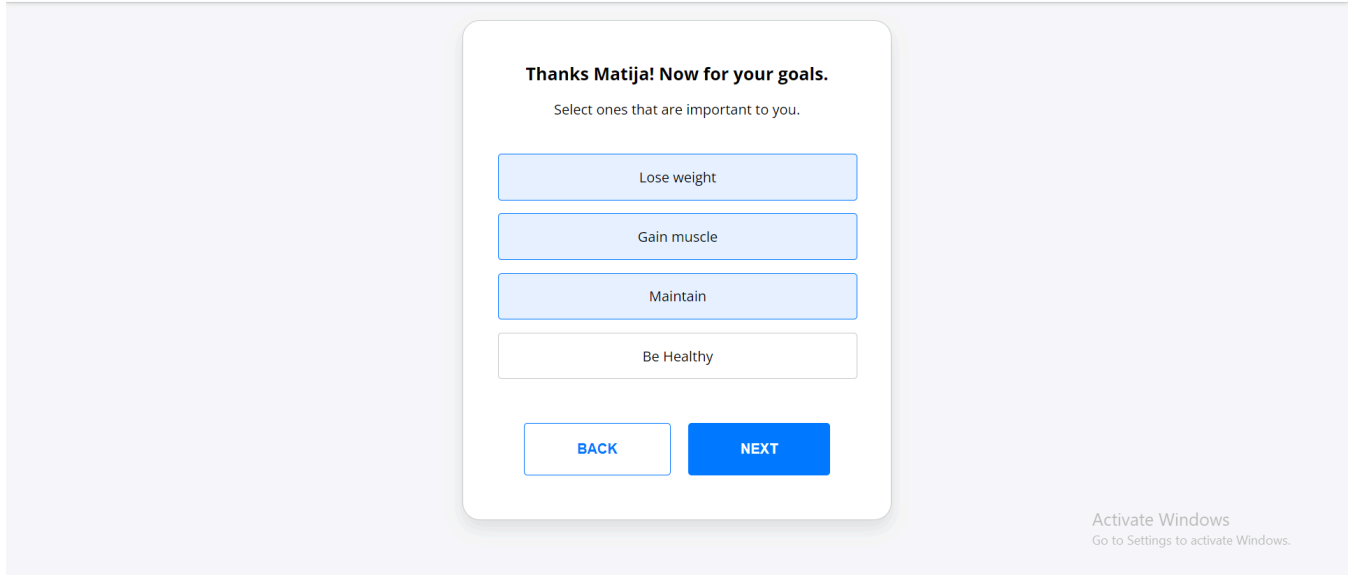
A weekly meal plan:

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Breakfast	Holy Guacamole Sourdough! (Alkalime Restaurant - Saadiyat) (350)	Wild Mushrooms & Chicken Pasta (leen's casual to gourmet - al markaziya) (550)	Keep Calm and Curry On (Alkalime Restaurant - Saadiyat) (300)	Curried Lentils (asha's - galleria mall) (250)	Fettuccine Alfredo with Shrimp (The cheesecake factory - The galleria mall) (650)	Nasi Goreng (chin chin - al zahiyah) (550)	Bhindi Masala (asha's - galleria mall) (200)
	Protein: 8g, Fat: 20g, Carbs: 35g	Protein: 35g, Fat: 22g, Carbs: 15g	Protein: 25g, Fat: 15g, Carbs: 15g	Protein: 13g, Fat: 5g, Carbs: 38g	Protein: 35g, Fat: 35g, Carbs: 55g	Protein: 35g, Fat: 30g, Carbs: 50g	Protein: 3g, Fat: 10g, Carbs: 24g
	Alkalime Restaurant - Saadiyat	leen's casual to gourmet - al markaziya	Alkalime Restaurant - Saadiyat	asha's - galleria mall	The cheesecake factory - The galleria mall	chin chin - al zahiyah	asha's - galleria mall
Lunch	Chicken Kabsa (Kababji Grill) (500)	What's Your Beef (Alkalime Restaurant - Saadiyat) (500)	Chicken & Shiitake Gyoza (leen's casual to gourmet - al markaziya) (350)	Oodles of Noodles - Chicken (Alkalime Restaurant - Saadiyat) (550)	The Real Burger (Alkalime Restaurant - Saadiyat) (600)	Baked Falafel Platter (Alkalime Restaurant - Saadiyat) (400)	The Big Supreme 6 Inch Meal (Subway Reem Island) (700)
	Protein: 35g, Fat: 15g, Carbs: 50g	Protein: 40g, Fat: 20g, Carbs: 40g	Protein: 15g, Fat: 10g, Carbs: 35g	Protein: 35g, Fat: 18g, Carbs: 50g	Protein: 35g, Fat: 30g, Carbs: 45g	Protein: 15g, Fat: 20g, Carbs: 45g	Protein: 30g, Fat: 35g, Carbs: 60g
	Kababji Grill	Alkalime Restaurant - Saadiyat	leen's casual to gourmet - al markaziya	Alkalime Restaurant - Saadiyat	Alkalime Restaurant - Saadiyat	Alkalime Restaurant - Saadiyat	Subway Reem Island
Dinner	Blushing BBQ Teriyaki Salmon (Alkalime Restaurant - Saadiyat) (450)	Spinach Cottage Cheese (asha's - galleria mall) (300)	Penne Alfredo with Chicken (The cheesecake factory - The galleria mall) (650)	Almond Crusted Chicken (Alkalime Restaurant - Saadiyat) (500)	Lamb Curry (asha's - galleria mall) (600)	Louisiana Chicken Pasta (The cheesecake factory - The galleria mall) (700)	Zoodles Alfredo (Alkalime Restaurant - Saadiyat) (350)
	Protein: 35g, Fat: 18g, Carbs: 35g	Protein: 20g, Fat: 15g, Carbs: 10g	Protein: 35g, Fat: 35g, Carbs: 55g	Protein: 30g, Fat: 22g, Carbs: 30g	Protein: 40g, Fat: 40g, Carbs: 15g	Protein: 35g, Fat: 30g, Carbs: 65g	Protein: 10g, Fat: 20g, Carbs: 10g
	Alkalime Restaurant - Saadiyat	asha's - galleria mall	The cheesecake factory - The galleria mall	Alkalime Restaurant - Saadiyat	asha's - galleria mall	The cheesecake factory - The galleria mall	Alkalime Restaurant - Saadiyat

Test cases 28-31 passed. When the number of days in a schedule was an invalid value, the system displayed an error message, and when the number of days in a schedule was a valid value, the system successfully created the schedule.

Test case 32 failed. When the user selects conflicting goals, the system does not display an error message, instead it successfully updates goals.

Abu Dhabi Eats



The screenshot shows a mobile application interface for 'Abu Dhabi Eats'. A central white card with rounded corners and a subtle shadow contains the following elements:

- Title:** 'Thanks Matija! Now for your goals.' in bold black text.
- Instruction:** 'Select ones that are important to you.' in a smaller black font.
- Goal Selection:** Four rectangular buttons stacked vertically:
 - 'Lose weight' (light blue background, blue border)
 - 'Gain muscle' (light blue background, blue border)
 - 'Maintain' (light blue background, blue border)
 - 'Be Healthy' (white background, light blue border)
- Navigation:** Two buttons at the bottom of the card:
 - 'BACK' (white background, blue border, blue text)
 - 'NEXT' (solid blue background, white text)

In the bottom right corner of the app's light purple background, there is a small watermark that reads: 'Activate Windows Go to Settings to activate Windows.'

Test cases 33-34 passed. When the user entered a valid email address, the system created the profile successfully and when the user entered an invalid email address, the system displayed an error message.

Almost there!
Create your account.

useratdomaincom

Please include an '@' in the email address. 'useratdomaincom' is missing an '@'.

☒ I agree to the Terms & Conditions and Privacy Policy.

Continue

Activate Windows
Go to Settings to activate Windows.

Section 7: Recommendations on Software Quality

Provide suggestions to enhance the software's quality based on the testing results.

Considering the testing results and the current stage of the "Abu Dhabi Eats" project, the following recommendations can enhance the software's quality:

1. **Continuous Integration and Continuous Deployment (CI/CD):** Implement CI/CD pipelines to ensure that code changes are systematically validated by automated tests and deployments. This approach helps in identifying integration issues early and reduces the risk of deployment failures.
2. **Increase Test Coverage:** Expand both unit and integration test coverage to encompass all critical paths and edge cases in the application. This would include more comprehensive tests for user profile management, meal recommendation logic, and nutritional tracking to ensure robustness across various scenarios.
3. **Automated Regression Testing:** Develop a suite of automated regression tests that can be run with each release. This will help in ensuring that new changes do not break existing functionalities, particularly for complex features like meal planning and dietary tracking.
4. **User Acceptance Testing (UAT):** Conduct structured User Acceptance Testing with real users to gather feedback on the usability and functionality of the system. This feedback can be invaluable in refining user interfaces and interactions, especially in ensuring that meal recommendations and nutritional advice are user-friendly and actionable.
5. **Performance Optimization:** As the application scales, performance testing should be a continuous part of the development cycle to ensure that the application can handle high user loads, especially during peak usage times. Optimizing database queries and server responses can significantly enhance user experience.
6. **Security Audits:** Regular security audits and updates to ensure that user data, particularly

sensitive information like health data and personal preferences, is protected against unauthorized access and breaches.

7. **Accessibility Enhancements:** Improve accessibility features to ensure that the app is usable by people with disabilities. This could include text-to-speech for meal descriptions, high-contrast display options, and easier navigation.
8. **Feedback Loops:** Establish mechanisms for users to provide feedback easily within the app. Use this feedback to continuously improve the system, focusing on user-reported issues and feature requests.
9. **Quality Assurance (QA) Team Expansion:** Consider expanding the QA team to include specialists in performance and security testing to address these critical areas more thoroughly.
10. **Documentation and Training:** Enhance documentation for both end-users and system administrators. Provide training materials to help users fully utilize all features of the app, thereby improving satisfaction and engagement.

Implementing these recommendations will require careful planning and resources but will significantly enhance the overall quality and reliability of the "Abu Dhabi Eats" software.