TEST PLAN

I. Overall Test Plan

Our approach to testing our application will focus mostly on checking the functionality ò the app and making sure each component is working as intended. Since we are dividing the workload into three main parts, our test plan will also follow this division of labor. Once both the frontend and backend components are functional, we will move on to integration and testing for integration issues. We will also address some abnormal test cases to ensure that our authorization based functionality is implemented correctly.

II. Test Case Descriptions

- **BE1:**

1. Backend Test 1
2. This test will ensure that the backend framework registers a new user correctly.
3. The user will input the required information on the temporary signup page to register a new user.
4. Inputs: Required user information.
5. Outputs: Created user is shown in the admin management tool/database.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **BE2:**

1. Backend Test 2
2. This test will ensure that the backend framework returns recommended nutritional data.
3. We will test to see if the backend can calculate and register the correct nutritional data for an user using the information obtained during their registration.
4. Inputs: User health information.
5. Outputs: User's recommended nutritional data is shown in the admin management tool/database.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **BE3:**

1. Backend Test 3
2. This test will ensure that the backend framework returns the correct API response.
3. We will test to see if the backend can send the correct API response with a response code to the frontend when information is requested.
4. Inputs: Website specific information formatted in JSON.
5. Outputs: API response with expected response code.
6. Normal

7. Blackbox
8. Functional
9. Unit Test

- **FE1:**

1. Frontend Test 1
2. This test will ensure that the frontend homepage displays all relevant components correctly.
3. We will test to see if the frontend homepage includes all relevant components and dropdown components functions correctly.
4. Inputs: No input.
5. Outputs: Functional homepage layout.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **FE2:**

1. Frontend Test 2
2. This test will ensure that the frontend web pages redirect to correct pages upon certain actions/requests.
3. We will test to see if each web page will redirect to the correct page when the user chooses a redirecting component.
4. Inputs: User clicks on redirecting components on the web page (buttons/links).
5. Outputs: The correct website is loaded and rendered.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **FE3:**

1. Frontend Test 3
2. This test will ensure that the meal recommender displays new meals when the page is refreshed.
3. We will test to see if the meal recommender will give out 3 new meals for the user to choose from upon clicking “New meals” or refreshing the page.
4. Inputs: User clicks on “New meals” or the reload button of the browser.
5. Outputs: 3 new meals are displayed to the user.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **FS1:**

1. Fullstack Test 1
2. This test will ensure that the backend APIs are received successfully by the frontend components and the content is displayed correctly when necessary.
3. We will test to see if the frontend can send a request to the backend, the backend can respond with an appropriate API response, and the frontend can handle the API response according to the status code received.
4. Inputs: Enter the URLs to different web pages that are part of the app package.
5. Outputs: The page loads successfully and displays relevant information, which is included in the API response.
6. Normal
7. Blackbox
8. Functional
9. Integration Test

- **FS2:**

1. Fullstack Test 2
2. This test will ensure that the frontend login and signup pages reload upon sending a request with missing/incorrect information (empty fields).
3. We will test to see if the pages for logging in and registering a user will only reload and not redirect to another page when the user submit an empty field.
4. Inputs: User presses login/sign up with an empty required field.
5. Outputs: The page refreshes and does not redirect to another page.
6. Normal
7. Blackbox
8. Functional
9. Unit Test

- **FS3:**

1. Fullstack Test 2
2. This test will ensure that the user is redirected to the login page upon manually entering any other app page to the URL if they are not logged in.
3. We will test to see if the user is redirected to the login page if they try to access other parts of the web app unauthorized.
4. Inputs: User enters a valid URL to a main page of the web app (Home, User Info, etc.)
5. Outputs: The app redirects to the login page.
6. Abnormal
7. Blackbox
8. Functional
9. Unit Test

- **FS4:**

1. Fullstack Test 3

2. This test will ensure that if the user is logged in and tries to navigate to the login or signup page, they will get redirected back to the page they were previously on.
3. We will test to see if the user is logged in and tries to navigate to the login or signup page, they will get redirected back to the page they were previously on. This is to make sure the user cannot login or sign up when they are already authorized to be in the application.
4. Inputs: User enters the URL to the login or signup page of the web app
5. Outputs: The app redirects the user back to whichever page they were on before trying to go to the login/signup page.
6. Abnormal
7. Blackbox
8. Functional
9. Unit Test

III. Test Case Matrix

	Normal/ Abnormal	**Blackbox/ Whitebox**	**Functional/ Performance**	**Unit/ Integration**
BE1	Normal	Blackbox	Functional	Unit
BE2	Normal	Blackbox	Functional	Unit
BE3	Normal	Blackbox	Functional	Unit
FE1	Normal	Blackbox	Functional	Unit
FE2	Normal	Blackbox	Functional	Unit
FE3	Normal	Blackbox	Functional	Unit
FS1	Normal	Blackbox	Functional	Integration
FS2	Abnormal	Blackbox	Functional	Unit
FS3	Abnormal	Blackbox	Functional	Unit
FS4	Abnormal	Blackbox	Functional	Unit