A picture containing text, sign

Description automatically generated

**LiftRight**

Ali salah 221000295

Alzahraa Gwyly 221000968

Jana Mokhtar 221000

Mohamed Nasr 221000089

Yomna Awad Khairy 221001896

**Under the supervision of**

Dr. Mohamed Elgazzar

Eng. Ahmed Jamal

Table of Contents

[Project Idea 3](#_Toc185721186)

[Problem Statement 3](#_Toc185721187)

[User Personas 4](#_Toc185721188)

[System Specifications 5](#_Toc185721189)

[1. Introduction 5](#_Toc185721190)

[2. System Objectives 5](#_Toc185721191)

[3. Functional Specifications 5](#_Toc185721192)

[**3.1. User Management** 5](#_Toc185721193)

[**3.2. Profile Management** 6](#_Toc185721194)

[**3.3. Workout Plan Generation** 6](#_Toc185721195)

[**3.4. Diet Recommendations** 6](#_Toc185721196)

[**3.5. Calorie Calculator Using BMR For Unlogged In Users** 6](#_Toc185721197)

[**3.6. Calorie Calculator Using BMR For Unlogged In Users** 6](#_Toc185721198)

[4. Non-Functional Specifications 6](#_Toc185721199)

[**4.1. Performance** 6](#_Toc185721200)

[**4.2. Scalability** 6](#_Toc185721201)

[**4.3. Security** 6](#_Toc185721202)

[**4.4. Compatibility** - Support major modern browsers: Chrome, Firefox, Safari, Edge. 6](#_Toc185721203)

[**4.5. Usability** 6](#_Toc185721204)

[**4.6. Maintainability** 7](#_Toc185721205)

[5. System Architecture 7](#_Toc185721206)

[**5.1. Client-Side** 7](#_Toc185721207)

[**5.2. Server-Side** 7](#_Toc185721208)

[**5.3. Database** 7](#_Toc185721209)

[6. Use Case Examples ( More Comprehensive in the Diagrams Section For Documentation ) 8](#_Toc185721210)

[**6.1. User Registration** 8](#_Toc185721211)

[**6.2. Generate Workout Plan** 8](#_Toc185721212)

[**6.3. Generate Diet Recommendation** 8](#_Toc185721213)

[Deployment Plan 8](#_Toc185721214)

[1. Objective 8](#_Toc185721215)

[2. Pre-Deployment Checklist 9](#_Toc185721216)

[3. Environment Details 9](#_Toc185721217)

[4. Deployment Process 9](#_Toc185721218)

[5. Rollback Plan 9](#_Toc185721219)

[6. Testing and Validation 9](#_Toc185721220)

[7. Communication Plan 10](#_Toc185721221)

[8. Timeline 10](#_Toc185721222)

[9. Resources 10](#_Toc185721223)

[10. Post-Deployment Activities 10](#_Toc185721224)

[1. Acceptance Criteria 10](#_Toc185721225)

[2. Automated Tests 11](#_Toc185721226)

[API 11](#_Toc185721227)

[1. Overview 11](#_Toc185721228)

[2. Authentication 12](#_Toc185721229)

[3. Endpoints 12](#_Toc185721230)

[**3.1 User Management** 12](#_Toc185721231)

[**3.2 Profile Management** 13](#_Toc185721232)

[**3.3 Workout and Diet Plans** 14](#_Toc185721233)

[**3.4 Utility Endpoints** 14](#_Toc185721234)

[4. Error Handling 15](#_Toc185721235)

[5. Example Request and Response Formats 15](#_Toc185721236)

[DIAGRAMS: INCLUDED IN PROJECT DOCUMENTATION FOLDER SEPERATELY. 16](#_Toc185721237)

Project Idea

The Workout Planner App is a Django-based application designed to provide users with personalized workout plans and diet recommendations. It tailors plans based on user inputs such as height, weight, age, body fat percentage, activity level, and fitness goals. The app aims to simplify fitness planning by offering easily accessible and actionable advice for users with diverse fitness levels and objectives.

**Problem Statement**

Many individuals struggle with planning effective workouts and diets tailored to their specific needs and goals. Existing solutions are often overly complex, generic, or require significant expertise to use effectively. The Workout Planner App addresses this gap by providing personalized fitness and nutrition plans in a user-friendly platform, enabling users to focus on their fitness journey without unnecessary barriers.

**User Personas**

1. Ali – The Beginner

- Age: 25  
- Occupation: Student  
- Goals: Lose weight and get fit.  
- Pain Points: Doesn’t know how to start with workouts or diet planning.  
- Needs: Easy-to-follow workout plans and diet suggestions that fit a student budget.

2. Sara – The Fitness Enthusiast

- Age: 30  
- Occupation: Marketing Professional  
- Goals: Build muscle and track progress.  
- Pain Points: Wants advanced plans tailored to her needs.  
- Needs: Plans that focus on specific muscle groups and advanced workout suggestions.

3. Omar – The Busy Parent

- Age: 40  
- Occupation: Accountant  
- Goals: Stay healthy with limited time.  
- Pain Points: Doesn’t have time for complex workout routines.   
( Will be pinpointed by days/week available )  
- Needs: Short and effective workout plans and simple meal recommendations.

**System Specifications**

**1. Introduction**

The Workout Planner App is a web-based application designed to generate personalized workout and diet plans for users based on their fitness goals. The system provides a user-friendly platform for beginners, fitness enthusiasts, and busy professionals to simplify fitness planning.

**2. System Objectives**

- Provide tailored plans based on user inputs.  
- Maintain a simple and intuitive interface.  
- Ensure the app is mobile-friendly and works on all modern browsers.  
- Deliver results quickly, even with a high number of users.  
- Protect sensitive user data with robust security measures.

**3. Functional Specifications**

**3.1. User Management**

- Users can register, log in, log out, and reset passwords.  
- Secure authentication for all user accounts.

**3.2. Profile Management**

- Users can input and update personal details: age, gender, weight, height, activity level, and fitness goals.

**3.3. Workout Plan Generation**

- The app generates workout plans tailored to fitness level, user goals, and available days per week.

**3.4. Diet Recommendations**

- Calorie calculation based on BMR and user goals (maintenance, surplus, deficit).  
- Macronutrient breakdown (protein, fats, carbohydrates).

**3.5. Calorie Calculator Using BMR For Unlogged In Users**

- Pure BMR Calculator for people who’s only interested in calculating their needed calories

**3.6. Calorie Calculator Using BMR For Unlogged In Users**

- FFMI Calculator ( Fat-Free Mass Index ) Calculator

**4. Non-Functional Specifications**

**4.1. Performance**

- Generate workout and diet plans within 10 seconds or less.

**4.2. Scalability**

- Handle up to 1000 concurrent users effectively.

**4.3. Security**

- Encrypt user password.  
- Prevent unauthorized data access with secure authentication mechanisms.

**4.4. Compatibility**  
- Support major modern browsers: Chrome, Firefox, Safari, Edge.

**4.5. Usability**

- Maintain an intuitive design with clear instructions.

**4.6. Maintainability**

- Use modular and well-documented code.

**5. System Architecture**

The system consists of client-side components (frontend), server-side components (backend), and a database for storing user data and plans.

**5.1. Client-Side**

- Frontend Framework: HTML, CSS, JavaScript.  
- Responsive Design: Adapt layout for mobile and desktop views.

**5.2. Server-Side**

- Backend Framework: Django (Python).  
- Authentication: Django's built-in authentication system.  
- APIs: RESTful endpoints for managing user data, workout plans, and diet recommendations.

**5.3. Database**

- Database Type: SQLite

- Tables:  
 - Users: Stores user information (e.g., email, hashed password).  
 - Profiles: Stores user inputs (e.g., height, weight, goals).

- Meals: Parts of diet plans.  
 - Exercises: Parts of workouts.

- Workouts: Complete Workout Plan.  
 - Diet Plans: Complete Diet Plan.

**6. Use Case Examples ( More Comprehensive in the Diagrams Section For Documentation )**

**6.1. User Registration**

Actors: User, System  
Steps:  
1. User enters email and password.  
2. System validates input and creates a new account.  
3. User receives a confirmation email.

**6.2. Generate Workout Plan**

Actors: User, System  
Steps:  
1. User selects a goal (e.g., weight loss).  
2. User inputs details (e.g., weight, height, activity level).  
3. System generates a workout plan tailored to the user.

**6.3. Generate Diet Recommendation**

Actors: User, System  
Steps:  
1. User selects a goal (e.g., calorie deficit).  
2. System calculates calorie needs and macro distribution.  
3. User receives a detailed diet recommendation

**Deployment Plan**

**1. Objective**

The purpose of this deployment plan is to outline the steps required to deploy the Workout Planner application to the production environment. This deployment includes the implementation of new features, bug fixes, and performance enhancements, ensuring minimal downtime and disruption to users.

**2. Pre-Deployment Checklist**

Ensure the following steps are completed before deployment:

Verify that the production environment is ready and configured.

Ensure that all dependencies and packages are installed.

Take a complete backup of the existing database and application.

Verify the integrity of the new release package.

Inform the stakeholders and set up a communication channel for updates.

**3. Environment Details**

The deployment will take place in the following environments:

1. \*\*Development Environment\*\*: For testing and bug fixing.

2. \*\*Staging Environment\*\*: For final verification before production.

3. \*\*Production Environment\*\*: For end-user access.

All environments use the following technology stack: Django, PostgreSQL, and a Linux-based deployment server.

**4. Deployment Process**

The deployment process involves the following steps:

Pull the latest code from the repository.

Verify the code and run unit tests.

Package the application using Docker (if applicable).

Deploy the code to the production server.

Run database migrations and restart the application services.

**5. Rollback Plan**

If deployment fails or issues arise, the following steps will be taken:

1. Stop the deployed application.

2. Restore the backup of the previous application version.

3. Notify stakeholders of the rollback and the expected resolution timeline.

**6. Testing and Validation**

After deployment, perform the following tests to validate the application:

Smoke tests to ensure basic functionality is working.

Integration tests for critical workflows.

End-user tests to confirm UI and API functionality.

**7. Communication Plan**

Stakeholders will be notified at each stage of the deployment process. Key updates will be provided through a shared communication channel (e.g., Slack or email).

**8. Timeline**

The deployment will be scheduled as follows:

1. \*\*Preparation\*\*: [Date/Time]

2. \*\*Deployment\*\*: [Date/Time]

3. \*\*Validation and Monitoring\*\*: [Date/Time]

**9. Resources**

The following resources will be required for deployment:

Deployment Team: Responsible for executing the deployment process.

Backup Tools: For database and application backups.

Monitoring Tools: To track application performance post-deployment.

**10. Post-Deployment Activities**

After deployment, the following steps will be performed:

Monitor system performance and address any issues.

Validate all key functionalities with the stakeholders.

Archive deployment logs and update the documentation.

**Automated Testing**

**1. Acceptance Criteria**

The acceptance criteria for this project define the expected functionality and requirements that must be fulfilled for the application to be considered complete. These include:

- The user should be able to sign up with a username, email, and password, and the system should validate all inputs.

- Users should be able to log in and log out of their accounts.

- Users should be redirected to their profile page after logging in, where their BMI & Data is displayed.

- Users should be able to download workout plans based on their BMI/Goals.

- Users should be able to update their profile information such as age, weight, and height.

- The system should render static pages like Home, About, and Sign-Up without errors.

**2. Automated Tests**

The following automated tests were implemented to verify the functionality of the application:

1. `test\_render\_signup\_view\_get`: Tests that the Sign-Up page renders correctly.

2. `test\_render\_signup\_view\_post`: Tests that users can successfully sign up with valid data.

3. `test\_render\_login\_view\_get`: Tests that the Log-In page renders correctly.

4. `test\_render\_login\_view\_post`: Tests that users can log in with correct credentials.

5. `test\_render\_home\_page`: Tests that the Home page renders correctly.

6. `test\_render\_about\_page`: Tests that the About page renders correctly.  
  
More is inside the “ Diagrams “ Section For Documentation

API

LiftRight API Documentation provides details on the available endpoints, their request and response formats, error handling, and functionality. The API supports user management, profile updates, and the generation of personalized fitness and nutrition plans.

**1. Overview**

The API serves as the backbone for the LiftRight application, allowing users to interact seamlessly with features such as creating accounts, generating plans, and managing profiles. It is designed to be scalable, secure, and user-friendly.

**2. Authentication**

The API uses session-based authentication for most endpoints. Users must log in before accessing protected resources. Authentication ensures secure communication between the client and server.

Security features include:

- Encrypted passwords stored using industry standards.

- CSRF protection on all forms.

- HTTPS enforced for all API requests.

**3. Endpoints**

**3.1 User Management**

\*\*User Registration\*\*

\*\*URL:\*\* `/signup/`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Creates a new user account.

\*\*Request Body:\*\*

- `username`: string

- `email`: string

- `password1`: string

- `password2`: string

- `age`: integer (optional)

- `height`: integer (optional)

- `weight`: integer (optional)

\*\*Response:\*\*

- `message`: `User registered successfully.`

- `redirect\_url`: `/home/`

\*\*User Login\*\*

\*\*URL:\*\* `/login/`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Authenticates the user.

\*\*Request Body:\*\*

- `username`: string

- `password`: string

\*\*Response:\*\*

- `message`: `Login successful.`

- `redirect\_url`: `/profile/`

**3.2 Profile Management**

\*\*Get Profile Details\*\*

\*\*URL:\*\* `/profile/`

\*\*Method:\*\* `GET`

\*\*Description:\*\* Fetches user profile data.

\*\*Response:\*\*

- `username`: string

- `email`: string

- `age`: integer

- `height`: integer

- `weight`: integer

- `bmi`: float

\*\*Update Profile\*\*

\*\*URL:\*\* `/update-profile/`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Updates user profile information.

\*\*Request Body:\*\*

- `age`: integer

- `height`: integer

- `weight`: integer

\*\*Response:\*\*

- `message`: `Profile updated successfully.`

**3.3 Workout and Diet Plans**

\*\*Generate and Download Workout Plan\*\*

\*\*URL:\*\* `/download/`

\*\*Method:\*\* `GET`

\*\*Description:\*\* Generates and downloads a PDF of a customized workout plan.

\*\*Response:\*\*

- `Content-Type`: `application/pdf`

- `Content-Disposition`: `attachment; filename="{plan\_type}\_workout\_plan.pdf"`

\*\*Calorie Calculator\*\*

\*\*URL:\*\* `/calorie-calculator/`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Calculates daily calorie requirements.

\*\*Request Body:\*\*

- `weight`: float

- `height`: float

- `age`: integer

- `activity\_level`: choice (`sedentary`, `moderate`, `active`)

\*\*Response:\*\*

- `calories`: integer

**3.4 Utility Endpoints**

\*\*Fetch Exercises\*\*

\*\*URL:\*\* `/fetch-exercises/`

\*\*Method:\*\* `GET`

\*\*Description:\*\* Fetches exercises grouped by muscle target.

\*\*Response:\*\*

- `exercises`: List of exercises.

\*\*Fetch Meals\*\*

\*\*URL:\*\* `/fetch-meals/`

\*\*Method:\*\* `GET`

\*\*Description:\*\* Fetches recommended meals with macronutrient details.

\*\*Response:\*\*

- `meals`: List of meals with calories, protein, fats, and carbs.

**4. Error Handling**

The API uses standard HTTP status codes for errors.

- `400`: Bad Request – Invalid input provided.

- `401`: Unauthorized – Authentication required.

- `403`: Forbidden – Access denied.

- `404`: Not Found – The resource is unavailable.

- `500`: Internal Server Error – An unexpected error occurred.

**5. Example Request and Response Formats**

\*\*Request Example for `/calorie-calculator/`:\*\*

{  
 "weight": 70,  
 "height": 175,  
 "age": 25,  
 "activity\_level": "moderate"  
}

\*\*Response Example for `/calorie-calculator/`:\*\*

{  
 "calories": 2500  
}

DIAGRAMS: INCLUDED IN PROJECT DOCUMENTATION FOLDER SEPERATELY.

1. TEST CASES DOCUMENTATION
2. TEST CASES VISUALIZATION
3. USE CASE DIAGRAMS
4. SEQUENCE DIAGRAM
5. ACTIVITY DIAGRAM
6. CLASS DIAGRAM