EDGE PROJECT: Department of ICT, MBSTU

Deliverable

Project title: Personalized Fitness Plan Generator

Submitted by:Nusrat Jahan Bindu

Supervisor Name: Dr.Ziaur Rohman

November 7, 2024

Date	Revision	Release Notes
07-11-2024	Rev 01	Initial Release

Contents

1 Abstract

This project aims to create a web-based application that generates personalized fitness plans tailored to each user's fitness level, goals, and preferences. By using the Django framework, the project provides a secure and maintainable backend that manages user data and dynamically generates personalized workout and nutrition recommendations. This application aims to improve users' fitness journeys by offering customized plans and progress tracking, enabling them to meet their health goals more effectively.

2 Introduction

In today's world, there is a growing demand for personalized fitness solutions as users seek workout and dietary plans that cater specifically to their needs. Generic fitness plans often fail to accommodate individual preferences and abilities, leading to a lack of motivation and limited results. This project, "Personalized Fitness Plan Generator," addresses this gap by creating an adaptable platform that offers personalized plans, monitors progress, and suggests adjustments to keep users on track.

Django is chosen for this project due to its robust capabilities for backend development, including user authentication, database management, and template rendering, making it a great fit for creating a scalable web application.

3 Completion Plan

3.1 Phase 1: Planning and Requirements Gathering

Objective: Define project requirements and gather the necessary resources for implementation.

- User Research: Identify user expectations, such as workout types (e.g., strength, cardio), dietary requirements, and progress tracking.
- Feature Definition: List features required for the application, such as user authentication, plan customization, progress tracking, and a workout library.
- Data Sources: Research and source fitness-related data, like exercises categorized by muscle groups and nutritional guidelines.

- Requirement Analysis: Determine the technical requirements (e.g., backend, frontend, database) and non-functional requirements (e.g., performance, security).
- Project Timeline: Define the project timeline with milestones for each phase to ensure timely completion.

3.2 Phase 2: Design

Objective: Design the application architecture, database schema, and user interface.

3.2.1 System Architecture:

- Use the Django Model-View-Template (MVT) pattern to separate the application's logic, user interface, and data.
- Identify Django models for representing users, fitness plans, exercises, dietary preferences, and user progress.
- Plan for RESTful API endpoints if the application will support mobile or external clients in the future.

3.2.2 Database Design:

- User Model: Includes personal data like age, gender, fitness level, and goals.
- Exercise Model: Stores details about different exercises, such as name, type, difficulty, and muscle groups targeted.
- Plan Model: Connects to the User Model and holds information about each user's personalized fitness plan.
- Progress Model: Tracks user progress, including completed workouts, metrics like weight, and performance improvements.

3.2.3 User Interface:

Design wireframes for key screens, including:

- Signup/Login: For user authentication. Dashboard: Displays an overview of the user's current fitness plan and progress. Plan Customization Form: Allows users to select their goals and preferences. Exercise Library: Showcases exercises available in the plan.

Progress Tracking Page: Visualizes metrics like weight and performance improvements over time.

3.3 Phase 3: Development

Objective: Develop the backend, frontend, and integrate necessary features.

- Set up the Django project with the required models, views, and forms.
- Implement authentication using Django's built-in user management (or Django AllAuth for social login).
- Create views to handle user actions, such as:
- Registering and logging in/out.
- Updating profiles with fitness goals and preferences.
- Generating fitness plans based on user input.
- Recording completed workouts and other progress metrics.

3.3.1 Frontend Development:

- Use Django templates to render dynamic HTML content.
- Use CSS and JavaScript to style the interface and add interactivity (such as form validation and data visualizations).
- Integrate charting libraries (e.g., Chart.js or Plotly) to display progress metrics on the dashboard.

3.3.2 Personalization Logic:

- Write algorithms to generate workout and dietary plans based on user inputs (fitness level, goals, preferences).
- Example logic: If a user wants to build muscle, the algorithm should prioritize strength exercises with higher intensity and progressive overload.

3.4 Phase 4: Testing

Objective: Test the application to ensure it meets functionality and performance standards

- Unit Testing: Write tests for individual components, such as the user model, exercise model, and fitness plan generation logic.
- Integration Testing: Test interactions between different modules, such as user authentication combined with fitness plan creation.
- User Testing: Gather feedback from a sample group to evaluate user experience and make improvements.
- Performance Testing: Ensure the application can handle expected loads, especially during peak hours or when handling large datasets.

3.5 Phase 5: Deployment

Objective: Deploy the application to a production environment.

- Platform Selection: Deploy on a cloud platform like Heroku or DigitalOcean. Choose a platform based on budget, scalability, and ease of integration with Django.
- Database Configuration: Set up a managed PostgreSQL database for production.
- Environment Configuration: Configure environment variables for secret keys, database connections, and other sensitive information.
- Continuous Integration/Deployment (CI/CD): Use a CI/CD pipeline (e.g., GitHub Actions) to automate testing and deployment.

3.6 Phase 6: Maintenance

Objective: Monitor and update the application as needed.

- Bug Fixes: Identify and resolve bugs reported by users or detected in logs.
- Feature Updates: Add new features over time, such as more exercises or nutrition plans.

- Data Backup and Security: Regularly back up the database and review security configurations.
- User Feedback: Gather user feedback to improve functionality and address usability issues.

4 Proposed Application Details

4.1 Features:

- User Authentication: Secure user login and registration, possibly with social login options.
- Plan Customization: Users can specify their fitness level, goals (e.g., weight loss, muscle gain), and exercise preferences.
- Progress Tracking: Track metrics like weight, workout completion, and strength improvements over time.
- Exercise Library: A curated list of exercises with filters for difficulty and muscle groups.
- Recommendations: Generate personalized workout plans and dietary advice based on user profiles.

4.2 Technical Stack:

- Backend: Django (Python) for handling data management and business logic.
- Frontend: HTML, CSS, and JavaScript (with potential for React or Vue.js in the future).
- Database: PostgreSQL or SQLite during development.
- Deployment Platform: Heroku, DigitalOcean, or AWS.

5 Discussion

5.1 Challenges:

- Data Accuracy: Ensuring that workout recommendations are appropriate for each user's fitness level to prevent injury.

- User Engagement: Keeping users motivated through notifications, reminders, or gamified elements like rewards for consistency.

5.2 Benefits:

- Customization: Personalized plans make fitness more accessible and effective.
- Convenience: Users can access their plans anytime, on any device, as it's a web-based platform.

5.3 Future Enhancements:

- Mobile App: Develop Android and iOS versions for increased accessibility.
- AI-Powered Recommendations: Use machine learning to make smarter recommendations based on user feedback and progress.
- Social Features: Implement social features, like sharing workouts or joining challenges.

6 Conclusion

This project offers a structured, goal-oriented solution for users aiming to improve their fitness. By leveraging Django's capabilities, it creates a personalized, scalable, and user-friendly platform, encouraging users to achieve their fitness goals through customized plans and consistent tracking.