Develop a RESTful API for a Personalized Workout Plan system that allows users to create and manage customized workout plans and track their fitness goals. The system should provide a structured way to plan workouts, incorporating predefined exercises with detailed descriptions, and enable progress tracking.
Core Features:

1. User Authentication:
    - Secure user registration, login, and logout functionality.
    - Implement JWT for session management and secure API access.
2. Predefined Exercises Database:
    - A database of exercises, each with descriptions, instructions for execution, target muscles, etc.
    - Populate with at least 20 diverse predefined exercises initially.
3. Personalized Workout Plans:
    - Functionality for users to create tailored workout plans, specifying workout frequency, goals, exercise types, and daily session duration.
    - Ability for users to select exercises from the predefined list and customize their workout by setting repetitions, sets, duration, or distance.
4. Tracking and Goals:
    - Features for users to track their weight over time and set personal fitness goals, including weight objectives and exercise-specific achievements.
5. API documentation using tools like Swagger for easy endpoint testing and interaction.

Bonus Features:

6. Workout Mode:
    - A guided, real-time workout feature showing next exercises, sets, repetitions, and rest periods during a workout session.
    - Options for users to mark exercises as complete and note any adjustments to the planned workout.
7. Docker & Docker Compose files for easy setup and deployment.

Tech Stack:

- Backend Framework: Python with Flask/Django/FastAPI.
- Database: PostgreSQL or SQLite.
- Authentication: JWT for API security.
- Version Control: Git, focusing on a clear and meaningful commit history.

Specific Deliverables:

1. Source Code: Hosted on GitHub, including all core and bonus features as developed.
2. Database Seed Script: To initially populate the database with predefined exercises.
3. API Documentation: Detailed documentation for all endpoints, using Swagger or a similar tool, including authentication details.
4. README: Comprehensive setup and usage instructions, including environment setup and how to access API documentation.

Evaluation Criteria:

- Feature Implementation: Fulfillment of core features as specified, with bonus features adding extra merit.
- Code Quality: Cleanliness, organization, and adherence to Python best practices.
- Documentation Quality: Clarity and completeness of API documentation and project setup instructions.
- Security Practices: Robust implementation of JWT and adherence to security best practices.
- Commit History: Logical, well-documented commit history reflecting the development process.