**GymSync — Unified Fitness Lifestyle Dashboard**

**1. Executive Summary**

**GymSync** is a web-based platform designed to integrate gym workouts, meal planning, and time-blocked scheduling into a single productivity-focused dashboard. By consolidating these three core lifestyle components, GymSync empowers users to streamline their fitness routines with seamless tracking, intelligent suggestions, and visual insights.

**2. Problem Statement**

Current fitness tools typically address workouts, nutrition, and scheduling as separate apps. This approach leads to fragmented tracking, duplicative data entry, and inefficient context switching. GymSync aims to unify these domains and support a holistic fitness journey.

**3. Proposed Solution**

* **Workout Planner & Logger**: Create and log workouts using templates (e.g., push/pull/legs), track sets, reps, duration, and calories burned.
* **Meal Planner & Nutrition Tracker**: Log daily meals, track calories and macros, and use nutrition templates (e.g., keto, bulking, vegetarian).
* **Time-Block Scheduler**: Calendar view for scheduling workout and meal prep sessions with drag-and-drop functionality and notification reminders.
* **Dashboard Analytics**: Visual display of calories in vs. out, workout streaks, meal patterns, and time allocation.
* **Stretch Features**:
  + AI-powered recommendations for workouts and meals.
  + Calendar sync with Google/Outlook.
  + PDF report export.
  + Integration with fitness platforms (Fitbit, Apple Health).
  + User role customization (beginner, intermediate, pro).

**4. Technical Architecture**

**4.1 Frontend**

* React.js with Redux or Context API
* UI styling via Material UI or Tailwind CSS

**4.2 Backend**

* Django REST Framework
* PostgreSQL (SQLite for development)
* Celery + Redis (optional) for scheduling and notifications
* JWT Authentication (via djangorestframework-simplejwt)

**4.3 Storage & Sync**

* PostgreSQL for user data
* AWS S3 for media (optional; local during development)
* Import/export using JSON/CSV
* Optional Google Calendar API and Firebase for real-time sync

**4.4 Charts & Visuals**

* Chart.js or Recharts for dashboards (meal overviews, workout trends, time allocation)

**4.5 AI/ML (Stretch Goals)**

* Leverage OpenAI API or ML models in Django to generate personalized workout/meal suggestions and optimal scheduling.

**5. Weekly Development Plan**

| **Week** | **Goals & Deliverables** |
| --- | --- |
| **1** | Finalize feature scope, create UI mockups, define backend models and APIs |
| **2** | Implement meal planner UI; backend endpoints for meals; front-back integration |
| **3** | Develop workout planner and logging features; connect to API |
| **4** | Build calendar/time-blocking component with drag-and-drop scheduling and reminders |
| **5** | Implement dashboard with charts and data visualizations |
| **6** | Integrate optional AI recommendation engine; begin testing |
| **7** | Merge modules, polish UI/UX, integrate CI/CD pipelines |
| **8** | Conduct full testing, bug-fixing, prepare demo, and deploy to Vercel + Render/Railway |

**6. Figma Wireframe Overview**

Outlined below is a simplified structure for designing wireframes in Figma (detailed layout to follow in Figma file):

* **Login / Onboarding Screen**
  + Simple login form; onboarding questionnaire (goals, dietary preferences, fitness level)
* **Main Dashboard**
  + Top navigation (Workouts / Meals / Calendar / Dashboard)
  + Summary cards (today’s calories, upcoming workout/meal blocks, streak)
* **Workout Planner**
  + Left panel: weekly view with days
  + Center: add/edit workout modal (name, sets/reps, duration)
  + Right panel: template library
* **Meal Planner**
  + Similar structure to workout planner, with nutritional details in modal
* **Calendar Scheduler**
  + Full-week calendar grid
  + Draggable blocks for workouts and meals
  + Reminder icon and settings per block
* **Dashboard**
  + Multi-chart view: calories in/out, workout frequency, time allocation, streaks

**7. System Design Diagram**

**[Central Backend: Django REST API]**

* **Clients**
  + React Web / Optional Electron Desktop App
* **API Layer**
  + JWT-authenticated routes for Users, Workouts, Meals, Schedule blocks
* **Database**
  + PostgreSQL with tables: Users, Workouts, WorkoutEntries, MealEntries, ScheduleBlocks
* **Background Services (Optional)**
  + Celery workers + Redis for reminders, scheduled notifications, sync jobs
* **External APIs**
  + OpenAI API / ML service for recommendations
  + Google/Outlook Calendar sync
  + Fitness platform integrations (Fitbit, Apple Health)
* **Storage**
  + Media on AWS S3; data export to JSON/CSV
* **Frontend Data Flow**
  + Redux store managing authentication, data caches, UI state

**8. Why GymSync?**

* Consolidates fitness, nutrition, and scheduling in one dashboard
* Minimizes app-switching and redundant data entry
* Leverages structured visualization for tracking progress
* Provides extensibility via AI, calendar sync, PDF exports

**9. Next Steps**

* Full Figma wireframes + interactive prototype
* Detailed Django REST API route documentation
* Task breakdown with team assignments
* MongoDB schema + Django model sync plan
* Starter GitHub repository scaffolding with frontend and backend templates