

ForAthlete MVP - Development Setup Guide

Project Overview

ForAthlete is a cross-platform training app that combines running and badminton training into one adaptive system. It acts as a personal coach powered by AI.

Target: Summer 2026 Internship Portfolio Project

Companies: Microsoft, Wise, Twilio, Ericsson

Tech Stack

Backend

- **Framework:** FastAPI (Python)
- **Database:** PostgreSQL (hosted on Supabase)
- **Authentication:** JWT tokens with passlib/bcrypt
- **Migrations:** Alembic
- **Deployment Target:** Railway, Render, or Fly.io

Frontend

- **Framework:** Next.js 15 with App Router
- **Styling:** Tailwind CSS v4
- **Language:** TypeScript
- **HTTP Client:** Axios
- **Deployment Target:** Vercel

Project Structure



```
forAthlete/
├── backend/
│   ├── app/
│   │   ├── core/
│   │   │   ├── __init__.py
│   │   │   ├── config.py      # Settings and environment config
│   │   │   ├── database.py   # SQLAlchemy setup
│   │   │   └── security.py   # Password hashing & JWT
│   │   ├── models/
│   │   │   ├── __init__.py
│   │   │   └── user.py       # User database model
│   │   ├── routes/
│   │   │   ├── __init__.py
│   │   │   └── auth.py       # Authentication endpoints
│   │   ├── schemas/
│   │   │   ├── __init__.py
│   │   │   └── user.py       # Pydantic models
│   │   ├── services/
│   │   │   ├── __init__.py
│   │   │   └── main.py       # FastAPI app entry point
│   │   └── alembic/          # Database migrations
│   ├── venv/                 # Python virtual environment
│   ├── .env                  # Environment variables
│   ├── alembic.ini
│   └── requirements.txt
└── frontend/
    ├── app/
    │   ├── dashboard/
    │   │   └── page.tsx      # Protected dashboard
    │   ├── login/
    │   │   └── page.tsx      # Login/register page
    │   ├── globals.css
    │   ├── layout.tsx        # Root layout
    │   └── page.tsx          # Home page
    └── lib/
        ├── api.ts            # API client and endpoints
        └── auth.ts           # Auth utilities
    ├── node_modules/
    ├── package.json
    └── tsconfig.json
    .gitignore
```

Setup Instructions

Prerequisites

- Python 3.9+
- Node.js 18+
- Git
- Supabase account (free)

Backend Setup

1. Create Project Structure



bash

```
mkdir forAthlete
```

```
cd forAthlete
```

```
git init
```

```
mkdir backend frontend
```

2. Backend Environment



bash

```
cd backend
```

```
python3 -m venv venv
```

```
source venv/bin/activate # Windows: venv\Scripts\activate
```

```
# Install dependencies
```

```
pip install fastapi uvicorn python-dotenv sqlalchemy psycopg2-binary \
alembic pydantic-settings python-jose passlib python-multipart bcrypt
```

```
pip freeze > requirements.txt
```

3. Create Environment Variables

Create backend/.env:



env

```
# Database
DATABASE_URL=postgresql://postgres.[project-ref]:[password]@aws-0-eu-central-1.pooler.supabase.com:6543/postgres

# Security (generate with: python3 -c "import secrets; print(secrets.token_urlsafe(32))")
SECRET_KEY=your-generated-secret-key-here
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=30

# API Keys (for future AI integration)
OPENAI_API_KEY=
```

4. Database Setup (Supabase)

1. Go to <https://supabase.com>
2. Create new project in Europe (Frankfurt) region
3. Copy the connection pooling URI from Settings → Database
4. Update DATABASE_URL in .env

5. Initialize Database



bash

```
# Initialize Alembic
alembic init alembic

# Create migration
alembic revision --autogenerate -m "Create users table"

# Apply migration
alembic upgrade head
```

6. Run Backend Server



```
uvicorn app.main:app --reload
```

Visit <http://localhost:8000/docs> to see API documentation.

Frontend Setup

1. Create Next.js App



bash

```
cd ./frontend
```

```
npx create-next-app@latest . --typescript --tailwind --app --no-src-dir --import-alias "@/*"
```

```
# Install dependencies
```

```
npm install axios
```

2. Create Folder Structure



bash

```
mkdir -p app/login app/dashboard lib
```

```
touch lib/api.ts lib/auth.ts
```

3. Run Frontend Server



bash

```
npm run dev
```

Visit <http://localhost:3000>

Database Schema

Users Table



sql

```
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    hashed_password VARCHAR NOT NULL,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
```

```
CREATE INDEX ix_users_email ON users(email);
```

Future Tables (to be implemented):

- user_profiles - Training preferences and schedule
- training_plans - Weekly training plans
- planned_workouts - Individual workout sessions
- workout_logs - Completed workouts with metrics
- daily_checkins - HRV, sleep, soreness data
- ai_conversations - AI coach chat history
- weekly_metrics - Aggregated performance data

API Endpoints

Authentication

POST /api/auth/register

Register a new user.

Request:



json

```
{
    "email": "user@example.com",
    "name": "John Doe",
    "password": "securepassword123"
}
```

Response (201):



json

```
{  
  "id": "uuid",  
  "email": "user@example.com",  
  "name": "John Doe",  
  "created_at": "2025-10-01T17:45:08.884292"  
}
```

POST /api/auth/login

Login and receive JWT token.

Request (form-urlencoded):



```
username=user@example.com  
password=securepassword123
```

Response (200):



json

```
{  
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
  "token_type": "bearer"  
}
```

GET /api/auth/me

Get current user info (requires authentication).

Headers:



Authorization: Bearer {token}

Response (200):



json

```
{  
    "id": "uuid",  
    "email": "user@example.com",  
    "name": "John Doe",  
    "created_at": "2025-10-01T17:45:08.884292"  
}
```

Key Code Files

Backend: app/main.py



```
from fastapi import FastAPI  
from fastapi.middleware.cors import CORSMiddleware  
from app.routes import auth  
  
app = FastAPI(title="ForAthlete API", version="1.0.0")  
  
app.add_middleware(  
    CORSMiddleware,  
    allow_origins=["http://localhost:3000", "http://localhost:3001"],  
    allow_credentials=True,  
    allow_methods=["*"],  
    allow_headers=["*"],  
)  
  
app.include_router(auth.router)  
  
@app.get("/")  
def read_root():  
    return {"message": "ForAthlete API is running", "version": "1.0.0"}  
  
@app.get("/health")  
def health_check():  
    return {"status": "healthy"}
```

Frontend: lib/api.ts



typescript

```
import axios from 'axios';

const API_BASE_URL = 'http://localhost:8000';

export const api = axios.create({
  baseURL: API_BASE_URL,
  headers: { 'Content-Type': 'application/json' },
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const authAPI = {
  register: (email: string, name: string, password: string) =>
    api.post('/api/auth/register', { email, name, password }),

  login: (email: string, password: string) =>
    api.post('/api/auth/login', new URLSearchParams({
      username: email,
      password: password,
    }), {
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    }),

  me: () => api.get('/api/auth/me'),
};
```

MVP Features Completed

Phase 1: Foundation (Complete)

- User registration and authentication
- JWT token-based security
- Database setup with migrations
- CORS configuration
- API documentation with Swagger UI

Phase 2: Frontend Auth (Complete)

- Login/register page with form validation
 - Protected dashboard route
 - Token storage and management
 - User session handling
 - Logout functionality
-

Next Development Phases

Phase 3: User Profile & Onboarding

- User profile model (badminton days, running goals)
- Onboarding flow to collect training schedule
- Profile settings page

Phase 4: Training Plan Generation

- Rule-based training plan generator
- Weekly plan creation algorithm
- Load management (15% increase cap)
- Fixed training days (Wed = badminton, etc.)
- REST API endpoints for plans

Phase 5: Daily Check-ins

- Morning check-in form (HRV, sleep, soreness)
- Apple Health integration planning
- Recovery metrics dashboard
- Check-in history view

Phase 6: AI Coach Integration

- OpenAI/Claude API integration
- Conversational workout adjustments
- Training recommendations based on recovery
- Chat interface in dashboard

Phase 7: Workout Logging

- Manual workout entry
- Strava API integration planning
- RPE (Rate of Perceived Exertion) tracking
- Workout history and trends

Phase 8: Analytics & Metrics

- Weekly volume tracking
 - Load calculation and visualization
 - Compliance percentage
 - Progress trends charts
-

Common Issues & Solutions

Issue: Port 3000 already in use

Solution: Update CORS to allow port 3001 or kill the process:



```
kill -9 $(lsof -ti:3000)
```

Issue: bcrypt version error

Solution: Reinstall bcrypt:



```
pip install --upgrade bcrypt
```

Issue: Tailwind CSS errors

Solution: Use Tailwind v4 syntax:



```
@import "tailwindcss";  
@import "tailwindcss/theme" layer(theme);  
@import "tailwindcss/preflight" layer(base);  
@import "tailwindcss/utilities" layer(utilities);
```

Issue: CORS blocked requests

Solution: Verify backend CORS middleware includes frontend URL

Git & Deployment

Git Setup



```
# Initialize and commit
git add .
git commit -m "Initial MVP setup: Backend auth + Frontend login/dashboard"

# Push to GitHub
git remote add origin git@github.com:USERNAME/forAthlete.git
git push -u origin main
```

.gitignore



```
# Environment
```

```
.env
.env.local
```

```
# Dependencies
```

```
node_modules/
venv/
```

```
# Build
```

```
.next/
__pycache__/
*.pyc
```

```
# OS
```

```
.DS_Store
```

Future Deployment

Backend:

- Railway, Render, or Fly.io
- Docker containerization
- Environment variables in platform

Frontend:

- Vercel (one-click deployment)
- Automatic preview deployments
- Custom domain

Database:

- Already on Supabase (managed)
- Automatic backups included

Interview Talking Points

Why This Stack:

- FastAPI: Modern async Python, auto-generated docs, production-grade
- PostgreSQL: Relational data fits training plans well, ACID compliance
- Next.js: Industry standard, excellent DX, built-in routing
- Tailwind: Rapid prototyping, mobile-first, no CSS conflicts

Architecture Decisions:

- JWT tokens: Stateless auth, scalable, works across devices
- Supabase: Managed database reduces ops overhead, free tier generous
- Monorepo: Easier development, atomic commits across stack
- Rule-based + AI: Safety guarantees with flexibility

Scaling Considerations:

- Redis caching for training plans
- Celery for background jobs (weekly plan generation)
- Database indexing on frequently queried fields
- CDN for frontend assets

Future Enhancements:

- React Native mobile app conversion
- Offline mode with service workers
- Real-time coach notifications
- Social features (training partners)
- Integration with wearables (Garmin, Apple Watch)

Resources

Documentation:

- FastAPI: <https://fastapi.tiangolo.com>
- Next.js: <https://nextjs.org/docs>
- Supabase: <https://supabase.com/docs>
- Tailwind CSS: <https://tailwindcss.com>

Repository:

- GitHub: <https://github.com/ChrisRobinT/forAthlete>

Summary

You have successfully built a production-ready MVP foundation with:

- Complete authentication system
- Database persistence
- Clean API architecture
- Responsive web interface
- Git version control

The project is well-structured for incremental feature development and demonstrates full-stack capabilities suitable for technical internship applications.

Total Development Time: ~4 hours (October 1, 2025)

Lines of Code: ~1,500 (backend + frontend)

Next Session Goals: User profiles, training plan generation logic, or daily check-in form.