SmartPrep AI - Complete Documentation

Version: 1.0.0

Last Updated: December 2024

Author: SmartPrep AI Development Team

Table of Contents

---

1. Project Overview

SmartPrep AI is a full-stack web and mobile application designed to help students prepare for competitive exams including:

• GATE CS (Computer Science)

• GATE DA (Data Science & AI)

• JEE Mains

• JEE Advanced

• EAMCET

• PlacementPrep (IT)

Key Features

• **Personalized Learning Plans**: AI-generated day-by-day study schedules

• **Multi-Course Support**: 6 different exam preparation tracks

• **Progress Tracking**: Streak tracking, task completion, and analytics

• **Responsive Design**: Web and native mobile applications

• **Secure Authentication**: JWT-based authentication system

• **MongoDB Integration**: Flexible NoSQL database for user management

• **OpenAI Integration**: ChatGPT-powered schedule generation

---

2. Technology Stack

Backend

• **Framework**: Django 4.2.7

• **API**: Django REST Framework 3.14.0

• **Database**: SQLite (dev) / PostgreSQL (prod)

• **NoSQL**: MongoDB with PyMongo 4.6.0

• **Authentication**: JWT with PyJWT 2.8.0

• **AI**: OpenAI API 1.3.0

• **Environment**: django-environ 0.11.2

• **CORS**: django-cors-headers 4.3.1

Frontend (Web)

• **Framework**: React.js 18.2

• **Routing**: React Router DOM 6.8

• **Styling**: Tailwind CSS 3.2

• **HTTP Client**: Axios 1.3.4

• **Build Tool**: react-scripts 5.0.1

Mobile (React Native)

• **Framework**: React Native 0.72

• **Navigation**: React Navigation 6.1

• **Storage**: AsyncStorage 1.19

• **Vector Icons**: 10.0.0

DevOps

• **Containerization**: Docker & Docker Compose

• **Python**: Python 3.11

• **Node**: Node.js 18+

---

3. Project Structure and File Paths

```
d:\NoviraLearn\                          # Project root
■
■■■ backend\                              # Django Backend
■   ■■■ authentication\                   # Authentication app
■   ■   ■■■ __init__.py
■   ■   ■■■ apps.py                       # App configuration
■   ■   ■■■ jwt_auth.py                   # JWT implementation
■   ■   ■■■ models.py                     # User model (lines 1-18)
■   ■   ■■■ mongodb_client.py             # MongoDB connection
■   ■   ■■■ serializers.py                # DRF serializers
■   ■   ■■■ urls.py                       # URL patterns
■   ■   ■■■ views.py                      # View functions (lines 1-60)
■   ■   ■■■ migrations\                   # Database migrations
```

```
■   ■   ■■■ __pycache__\
■   ■
■   ■■■ courses\                          # Courses app
■   ■   ■■■ __init__.py
■   ■   ■■■ admin.py                      # Admin interface
■   ■   ■■■ apps.py
■   ■   ■■■ models.py                     # Course models (lines 1-41)
■   ■   ■■■ serializers.py                # Serializers
■   ■   ■■■ urls.py                       # URL patterns
■   ■   ■■■ views.py                      # View functions (lines 1-41)
■   ■   ■■■ management\
■   ■   ■   ■■■ commands\
■   ■   ■       ■■■ populate_data.py      # Data population
■   ■   ■■■ migrations\
■   ■
■   ■■■ study_schedule\                   # Study schedule app
■   ■   ■■■ __init__.py
■   ■   ■■■ admin.py
■   ■   ■■■ apps.py
■   ■   ■■■ chatgpt_integration.py        # OpenAI integration (lines 1-212)
■   ■   ■■■ models.py                     # Schedule models (lines 1-93)
■   ■   ■■■ serializers.py
■   ■   ■■■ urls.py                       # Schedule URLs
■   ■   ■■■ views.py                      # View functions (lines 1-203)
■   ■   ■■■ migrations\
■   ■
■   ■■■ visitor_tracking\                 # Analytics app
■   ■   ■■■ __init__.py
■   ■   ■■■ admin.py
■   ■   ■■■ apps.py
■   ■   ■■■ models.py
■   ■   ■■■ urls.py
■   ■   ■■■ views.py
■   ■
■   ■■■ smartprep_backend\                # Main Django project
■   ■   ■■■ __init__.py
■   ■   ■■■ asgi.py                       # ASGI config
■   ■   ■■■ settings.py                   # Main settings (lines 1-150)
■   ■   ■■■ settings_mongodb.py           # MongoDB settings
■   ■   ■■■ settings.prod.py              # Production settings
■   ■   ■■■ urls.py                       # Root URL config (lines 1-13)
■   ■   ■■■ wsgi.py                       # WSGI config
■   ■
■   ■■■ .env                              # Environment variables (CRITICAL)
■   ■■■ manage.py                         # Django management
■   ■■■ requirements.txt                  # Python dependencies
■   ■■■ Dockerfile                        # Docker config
■   ■■■ db.sqlite3                        # SQLite database
■   ■■■ README.md                         # Backend README
■   ■■■ OPENAI_SETUP.md                   # OpenAI guide
■   ■■■ MONGODB_GUIDE.md                  # MongoDB guide
■
■■■ frontend\                             # React Web App
■   ■■■ public\
■   ■   ■■■ index.html                    # HTML template
■   ■
■   ■■■ src\
■   ■   ■■■ App.js                        # Main app (lines 1-68)
■   ■   ■■■ index.js                      # Entry point
■   ■   ■■■ index.css                     # Global styles
■   ■   ■
■   ■   ■■■ contexts\
■   ■   ■   ■■■ AuthContext.js            # Auth context
■   ■   ■
■   ■   ■■■ pages\
■   ■   ■   ■■■ LandingPage.js            # Landing page
■   ■   ■   ■■■ LoginPage.js              # Login
■   ■   ■   ■■■ SignupPage.js             # Signup
```

```
    ■   ■   ■   ■■■ Dashboard.js             # Dashboard
    ■   ■   ■   ■■■ CoursePlan.js            # Course details
    ■   ■   ■   ■■■ Profile.js               # User profile
    ■   ■   ■   ■■■ ScheduleCreation.js       # Schedule creation
    ■   ■   ■
    ■   ■   ■■■ services\
    ■   ■       ■■■ api.js                   # API client
    ■   ■
    ■   ■■■ package.json                     # NPM config
    ■   ■■■ package-lock.json
    ■   ■■■ tailwind.config.js               # Tailwind config
    ■   ■■■ postcss.config.js
    ■   ■■■ Dockerfile
    ■   ■■■ node_modules\
    ■
    ■■■ mobile\                              # React Native App
    ■   ■■■ src\
    ■   ■   ■■■ App.js                       # Main app (lines 1-52)
    ■   ■   ■
    ■   ■   ■■■ contexts\
    ■   ■   ■   ■■■ AuthContext.js           # Auth context
    ■   ■   ■
    ■   ■   ■■■ screens\
    ■   ■   ■   ■■■ LandingScreen.js
    ■   ■   ■   ■■■ LoginScreen.js
    ■   ■   ■   ■■■ SignupScreen.js
    ■   ■   ■   ■■■ DashboardScreen.js
    ■   ■   ■   ■■■ CoursePlanScreen.js
    ■   ■   ■   ■■■ ProfileScreen.js
    ■   ■   ■
    ■   ■   ■■■ services\
    ■   ■       ■■■ api.js                   # API client
    ■   ■
    ■   ■■■ index.js                         # Entry point
    ■   ■■■ app.json                         # App manifest
    ■   ■■■ package.json                     # NPM config
    ■   ■■■ node_modules\
    ■
    ■■■ docker-compose.yml                    # Docker Compose dev
    ■■■ docker-compose.prod.yml               # Docker Compose prod
    ■■■ mongodb_sample_data.js                # Sample data
    ■■■ mongodb_setup.md                      # MongoDB guide
    ■■■ README.md                             # Main README
    ■■■ env.example                           # Environment template
```

---

4. Environment Setup

Prerequisites

Install the following software:

Python 3.8+: [Download](https://www.python.org/downloads/)

• Verify: `python --version`

Node.js 16+: [Download](https://nodejs.org/)

• Verify: `node --version`

MongoDB: [Download](https://www.mongodb.com/try/download/community)

• Verify: `mongosh --version`

Git: [Download](https://git-scm.com/downloads)

Docker (Optional): [Download](https://www.docker.com/products/docker-desktop)

Backend Setup

Step 1: Navigate to Backend Directory

```
cd backend
```

Path: `d:\NoviraLearn\backend\`

Step 2: Create Virtual Environment

Windows PowerShell:

```
python -m venv venv
.\venv\Scripts\activate
```

Windows CMD:

```
python -m venv venv
venv\Scripts\activate
```

Linux/Mac:

```
python -m venv venv
source venv/bin/activate
```

Step 3: Install Dependencies

```
pip install -r requirements.txt
```

File: `backend\requirements.txt`

This installs:

• Django==4.2.7

• djangorestframework==3.14.0

• django-cors-headers==4.3.1

• PyJWT==2.8.0

• python-decouple==3.8

• django-environ==0.11.2

• pymongo==4.6.0

• dnspython==2.4.2

• openai==1.3.0

Step 4: Create Environment File

Create `.env` file in `backend\` directory:

File Path: `backend\.env`

```
# Django Configuration
SECRET_KEY=django-insecure-smarterp-ai-key-change-this-in-production-12345
DEBUG=True
DATABASE_URL=sqlite:///db.sqlite3
JWT_SECRET_KEY=jwt-secret-key-for-smartprep-ai-change-in-production-12345
ALLOWED_HOSTS=localhost,127.0.0.1,0.0.0.0
# MongoDB Configuration
MONGODB_HOST=localhost
MONGODB_PORT=27017
MONGODB_NAME=Noviradb
# OpenAI Configuration
OPENAI_API_KEY=sk-proj-sm_HSrkund40VY-hiMID-OGPHNxZi1Ca7oyspc7EWj_Yp9kvN96NUJjlDItcGnbc9YgDjQhKo8T3B
OPENAI_MODEL=gpt-4
```

Step 5: Run Migrations

```
python manage.py migrate
```

File: `backend\manage.py`

Step 6: Populate Initial Data

```
python manage.py populate_data
```

File: `backend\courses\management\commands\populate_data.py`

Step 7: Create Superuser (Optional)

```
python manage.py createsuperuser
```

Step 8: Start Development Server

```
python manage.py runserver
```

Access: http://localhost:8000

Admin Panel: http://localhost:8000/admin

---

5. Database Configuration

SQLite Database (Default)

File: `backend\db.sqlite3`

• Automatically created on first migration

• No additional configuration needed

• Used for Course, Schedule, and Task data

MongoDB Setup

Database Name: `Noviradb`

Collection: `authentication-signup`

Step 1: Install MongoDB

Download from: https://www.mongodb.com/try/download/community

Step 2: Start MongoDB Service

Windows:

```
mongod --dbpath "C:\data\db"
```

Linux/Mac:

```
sudo systemctl start mongod
# or
brew services start mongodb-community
```

Step 3: Create Database and Collection

Open MongoDB shell:

```
mongosh
```

Run these commands:

```
// Switch to database
use Noviradb
// Create collection
db.createCollection("authentication-signup")
// Create indexes
db["authentication-signup"].createIndex({ email: 1 }, { unique: true })
db["authentication-signup"].createIndex({ username: 1 }, { unique: true })
db["authentication-signup"].createIndex({ is_active: 1 })
// Verify
db["authentication-signup"].getIndexes()
```

Step 4: Configure in .env

Already configured in `backend\.env`:

```
MONGODB_HOST=localhost
MONGODB_PORT=27017
```

```
MONGODB_NAME=Noviradb
```

Step 5: Test Connection

```
# In Django shell
python manage.py shell
from authentication.mongodb_client import get_mongodb_connection
db = get_mongodb_connection()
print("Connected to:", db.name)
```

PostgreSQL Setup (Production)

For production deployment, update `.env`:

```
DATABASE_URL=postgresql://user:password@localhost:5432/smartprep_db
```

---

6. API Documentation

Base URL

Development: http://localhost:8000

Production: https://yourdomain.com

Authentication Endpoints

6.1 Register User

Endpoint: `POST /api/register/`

File: `backend\authentication\views.py` (lines 17-28)

Request:

```
{
  "first_name": "John",
  "last_name": "Doe",
  "username": "johndoe",
  "email": "john@example.com",
  "password": "password123",
  "confirm_password": "password123"
}
```

Response (201 Created):

```
{
  "message": "User created successfully",
```

```
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "user": {
    "id": 1,
    "email": "john@example.com",
    "username": "johndoe",
    "first_name": "John",
    "last_name": "Doe"
  }
}
```

6.2 Login

Endpoint: `POST /api/login/`

File: `backend\authentication\views.py` (lines 31-44)

Request:

```
{
  "email": "john@example.com",
  "password": "password123"
}
```

Response (200 OK):

```
{
  "message": "Login successful",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "user": {
    "id": 1,
    "email": "john@example.com",
    "username": "johndoe"
  }
}
```

6.3 Get Profile

Endpoint: `GET /api/profile/`

File: `backend\authentication\views.py` (lines 47-60)

Authentication: Required (Bearer token)

Headers:

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
```

Response (200 OK):

```
{
  "id": 1,
  "email": "john@example.com",
  "username": "johndoe",
  "first_name": "John",
  "last_name": "Doe"
}
```

6.4 Update Profile

Endpoint: `PUT /api/profile/`

Authentication: Required

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "username": "johnsmith"
}
```

Response (200 OK):

```
{
  "id": 1,
  "email": "john@example.com",
  "username": "johnsmith",
  "first_name": "John",
  "last_name": "Smith"
}
```

Course Endpoints

6.5 Get All Courses

Endpoint: `GET /api/courses/`

File: `backend\courses\views.py` (lines 9-15)

Authentication: Required

Response (200 OK):

```
[
  {
    "id": 1,
    "name": "GATE CS",
    "description": "Computer Science and Information Technology",
    "status": "active"
  },
  {
    "id": 2,
    "name": "GATE DA",
    "description": "Data Science and Artificial Intelligence",
    "status": "active"
  },
  {
    "id": 6,
    "name": "PlacementPrep (IT)",
    "description": "IT Placement Preparation",
    "status": "maintenance"
  }
]
```

6.6 Get Course Details

Endpoint: `GET /api/courses/{course_id}/`

File: `backend\courses\views.py` (lines 18-40)

Authentication: Required

Response (200 OK):

```
{
  "id": 1,
  "name": "GATE CS",
  "description": "Computer Science and Information Technology",
  "status": "active",
  "plans": [
    {
      "id": 1,
      "topic_name": "Data Structures and Algorithms",
      "description": "Learn fundamental data structures",
      "schedule_type": "daily",
      "order": 1,
      "is_completed": false
    }
  ]
}
```

Schedule Endpoints

6.7 Create Schedule

Endpoint: `POST /api/schedules/create/`

File: `backend\study_schedule\views.py` (lines 13-79)

Authentication: Required

Request:

```
{
  "course_id": 1,
  "target_year": 2025,
  "days_available": 120,
  "weekday_hours": 4,
  "weekend_hours": 8,
  "overall_proficiency": "Intermediate",
  "learning_pace": "Moderate",
  "current_stage": "Midway",
  "mock_test_frequency": "weekly",
  "daily_practice_problems": 20,
  "revision_rounds": 1,
  "syllabus_status": {
    "covered": 50,
    "remaining": 50
  }
}
```

Response (201 Created):

```
{
  "schedule_id": 1,
  "message": "Schedule generated successfully",
  "status": "completed",
  "total_tasks": 120
}
```

6.8 Get User Schedules

Endpoint: `GET /api/schedules/`

File: `backend\study_schedule\views.py` (lines 82-94)

Authentication: Required

Response (200 OK):

```
{
  "schedules": [
    {
      "id": 1,
      "course_name": "GATE CS",
      "start_date": "2024-01-01",
      "end_date": "2024-05-01",
      "total_days": 120,
      "completed_tasks": 30,
      "current_streak": 15
    }
  ],
  "count": 1
}
```

6.9 Get Schedule Details

Endpoint: `GET /api/schedules/{schedule_id}/`

File: `backend\study_schedule\views.py` (lines 97-108)

Authentication: Required

Response (200 OK):

```
{
  "id": 1,
  "course_name": "GATE CS",
  "start_date": "2024-01-01",
  "end_date": "2024-05-01",
  "tasks": [
    {
      "id": 1,
      "date": "2024-01-01",
      "task_type": "study",
      "subject": "Data Structures",
      "topic": "Arrays and Linked Lists",
      "estimated_duration": 120,
      "problem_count": 0,
      "status": "pending"
    }
  ]
}
```

6.10 Update Task Status

Endpoint: `PATCH /api/schedules/tasks/{task_id}/`

File: `backend\study_schedule\views.py` (lines 111-151)

Authentication: Required

Request:

```
{
  "status": "completed"
}
```

Response (200 OK):

```
        {
          "success": true,
          "task": {
            "id": 1,
            "status": "completed",
            "is_completed": true
          }
        }
```

---

7. Authentication Flow

JWT Implementation

File: `backend\authentication\jwt_auth.py`

Token Expiration: 24 hours

Algorithm: HS256

Login Flow

User submits credentials

Backend validates user

JWT token generated

Token returned to client

Client stores token in localStorage (web) or AsyncStorage (mobile)

Client includes token in subsequent requests via Authorization header

Protected Routes

File: `backend\smartprep_backend\settings.py` (lines 122-132)

```
        REST_FRAMEWORK = {
            'DEFAULT_AUTHENTICATION_CLASSES': [
                'authentication.jwt_auth.JWTAuthentication',
            ],
            'DEFAULT_PERMISSION_CLASSES': [
                'rest_framework.permissions.IsAuthenticated',
            ],
        }
```

Frontend Auth Context

File: `frontend\src\contexts\AuthContext.js`

File: `mobile\src\contexts\AuthContext.js`

---

8. OpenAI Integration

Configuration

API Key Location: `backend\.env`

```
OPENAI_API_KEY=sk-proj-sm_HSrkund40VY-hiMID-OGPHNxZi1Ca7oyspc7EWj_Yp9kvN96NUJjlDItcGnbc9YgDjQhKo8T3B
OPENAI_MODEL=gpt-4
```

Implementation

File: `backend\study_schedule\chatgpt_integration.py` (lines 1-212)

Main Function: `generate_study_schedule()` (lines 10-59)

How It Works

User completes schedule creation form

System calls `generate_study_schedule()` function

ChatGPT API receives comprehensive prompt (lines 62-114)

API generates personalized schedule

Response parsed to JSON format (lines 117-125)

Schedule stored in database

Tasks created for each day

Cost Estimates

• **GPT-4**: ~$0.30 per schedule

• **GPT-3.5-turbo**: ~$0.05 per schedule

File: `backend\OPENAI_SETUP.md`

---

9. Frontend Setup

Step 1: Navigate to Frontend Directory

```
cd frontend
```

Path: `d:\NoviraLearn\frontend\`

Step 2: Install Dependencies

```
npm install
```

File: `frontend\package.json`

Installs:

• react@18.2.0

• react-dom@18.2.0

• react-router-dom@6.8

• axios@1.3.4

• tailwindcss@3.2

Step 3: Configure Environment

Create `frontend\.env` (if needed):

```
REACT_APP_API_URL=http://localhost:8000
```

Step 4: Start Development Server

```
npm start
```

Access: http://localhost:3000

Step 5: Build for Production

```
npm run build
```

Output: `frontend\build\`

File Structure

Main App File: `frontend\src\App.js` (lines 1-68)

Pages:

• `frontend\src\pages\LandingPage.js`

• `frontend\src\pages\LoginPage.js`

• `frontend\src\src\pages\SignupPage.js`

• `frontend\src\pages\Dashboard.js`

• `frontend\src\pages\CoursePlan.js`

• `frontend\src\pages\Profile.js`

• `frontend\src\pages\ScheduleCreation.js`

API Service: `frontend\src\services\api.js`

---

10. Mobile App Setup

Prerequisites

• React Native CLI

• Android Studio (for Android)

• Xcode (for iOS, macOS only)

Step 1: Navigate to Mobile Directory

```
cd mobile
```

Path: `d:\NoviraLearn\mobile\`

Step 2: Install Dependencies

```
npm install
```

File: `mobile\package.json`

Step 3: Install iOS Dependencies (macOS only)

```
cd ios
pod install
cd ..
```

Step 4: Run on Android

```
npm run android
# or
npx react-native run-android
```

Step 5: Run on iOS (macOS only)

```
npm run ios
# or
npx react-native run-ios
```

File Structure

Main App File: `mobile\src\App.js` (lines 1-52)

Screens:

• `mobile\src\screens\LandingScreen.js`

• `mobile\src\screens\LoginScreen.js`

• `mobile\src\screens\SignupScreen.js`

• `mobile\src\screens\DashboardScreen.js`

• `mobile\src\screens\CoursePlanScreen.js`

• `mobile\src\screens\ProfileScreen.js`

API Service: `mobile\src\services\api.js`

---

11. Docker Deployment

Development Docker Setup

File: `docker-compose.yml`

Start all services:

```
docker-compose up --build
```

Services:

• PostgreSQL: localhost:5432

• Backend: localhost:8000

• Frontend: localhost:3000

Production Docker Setup

File: `docker-compose.prod.yml`

```
docker-compose -f docker-compose.prod.yml up --build -d
```

Backend Dockerfile

File: `backend\Dockerfile` (lines 1-37)

• Base: Python 3.11-slim

• Installs PostgreSQL client

• Copies requirements and installs dependencies

• Exposes port 8000

Frontend Dockerfile

File: `frontend\Dockerfile` (lines 1-24)

• Base: Node.js 18-alpine

• Installs dependencies

• Builds React app

• Exposes port 3000

---

12. Troubleshooting

Backend Issues

Database Connection Error

Problem: Cannot connect to SQLite

Solution:

```
cd backend
python manage.py migrate
```

MongoDB Connection Error

Problem: Cannot connect to MongoDB

Solution:

Verify MongoDB is running:

```
mongosh
```

Check .env configuration

Restart Django server

OpenAI API Error

Problem: "OpenAI API key not configured"

Solution:

Check `backend\.env` file exists

Verify `OPENAI_API_KEY` is set correctly

Restart Django server after adding key

File: `backend\OPENAI_SETUP.md`

CORS Error

Problem: CORS policy blocks requests

Solution: Check CORS settings in `backend\smartprep_backend\settings.py` (lines 135-140)

Frontend Issues

npm install fails

Problem: Node modules installation errors

Solution:

```
cd frontend
rm -rf node_modules package-lock.json
npm install
```

Port Already in Use

Problem: Port 3000 already in use

Solution:

```
# Change port
set PORT=3001 &amp;&amp; npm start
```

Mobile Issues

Android Build Fails

Problem: Gradle build errors

Solution:

```
cd android
./gradlew clean
cd ..
npm run android
```

iOS Build Fails (macOS)

Problem: CocoaPods errors

Solution:

```
cd ios
pod deintegrate
pod install
cd ..
```

---

13. Contributing

Development Workflow

Create feature branch

Make changes

Test thoroughly

Submit pull request

Code Style

• Python: Follow PEP 8

• JavaScript: Use ESLint config

• React: Use functional components

File Paths Reference

Important Files:

• Backend settings: `backend\smartprep_backend\settings.py`

• Backend URLs: `backend\smartprep_backend\urls.py`

• Environment: `backend\.env`

• Frontend app: `frontend\src\App.js`

• Mobile app: `mobile\src\App.js`

Quick Reference

Common Commands

```
# Backend
cd backend
python manage.py runserver
python manage.py migrate
python manage.py createsuperuser
python manage.py populate_data

# Frontend
cd frontend
npm start
npm run build

# Mobile
cd mobile
npm run android
npm run ios

# Docker
docker-compose up --build
docker-compose -f docker-compose.prod.yml up --build -d
```

Key File Paths

• **Django Settings**: `d:\NoviraLearn\backend\smartprep_backend\settings.py`

• **Environment Variables**: `d:\NoviraLearn\backend\.env`

• **Main URL Config**: `d:\NoviraLearn\backend\smartprep_backend\urls.py`

• **Authentication Views**: `d:\NoviraLearn\backend\authentication\views.py`

• **Course Views**: `d:\NoviraLearn\backend\courses\views.py`

• **Schedule Views**: `d:\NoviraLearn\backend\study_schedule\views.py`

• **OpenAI Integration**: `d:\NoviraLearn\backend\study_schedule\chatgpt_integration.py`

• **Frontend App**: `d:\NoviraLearn\frontend\src\App.js`

• **Mobile App**: `d:\NoviraLearn\mobile\src\App.js`

Environment Setup Checklist

• [ ] Python 3.8+ installed

• [ ] Node.js 16+ installed

• [ ] MongoDB installed and running

• [ ] Virtual environment created and activated

• [ ] Backend dependencies installed

• [ ] Frontend dependencies installed

• [ ] Mobile dependencies installed

• [ ] Environment variables configured

• [ ] Database migrated

• [ ] OpenAI API key configured

• [ ] Superuser created

Support and Contact

For issues and questions:

• Check existing documentation in docs/

• Open GitHub issue

• Contact: support@smartprep.ai

Made with ❤■ for students preparing for competitive exams

Version: 1.0.0

Last Updated: December 2024