# One-Month Development Plan for ATMA

## Week 1: Setup, Learning, & Planning

### Day 1-2: Kick-off & Environment Setup

- Review the ATMA requirements document.

- Confirm the tech stack:  
 - Backend: Django & Django REST Framework  
 - Frontend: React (using Create React App)

- Set up your development environment:  
 - Install Python, Node.js, Git, and IDEs for Django and React.  
 - Create a Git repository for version control.

### Day 3-4: Learn Django Basics

- Follow an introductory Django tutorial to understand:  
 - Django project structure, URL routing, views, and templates.  
 - How Django handles models and migrations.

- Create a basic “Hello World” Django app to get comfortable.

### Day 5: Introduction to Django REST Framework (DRF)

- Learn the fundamentals of DRF:  
 - Creating API endpoints, serializers, and viewsets.

- Build a simple API endpoint to return static JSON data.

### Day 6: Database & ORM with Django

- Explore Django’s ORM:  
 - Set up a local SQLite database.  
 - Create sample models and practice migrations.

- Familiarize yourself with Django’s admin interface.

### Day 7: Finalize System Architecture & Design Document

- Sketch the overall architecture:  
 - Backend: Django models for teachers, classrooms, courses, and timetable entries; API endpoints for scheduling.  
 - Frontend: React component hierarchy and data flow for dashboards and forms.

- Create/refine your design document with key models and API endpoints.

## Week 2: Backend Development & Scheduling Module

### Day 8-9: Develop Data Models in Django

- Define models for Teachers, Classrooms, Courses, and Timetable entries.

- Set up relationships and constraints in your models.

- Test basic CRUD operations via the Django admin and shell.

### Day 10-11: Implement the Scheduling Algorithm

- Develop the core scheduling logic in Python:  
 - Ensure no conflicts (e.g., overlapping classes, teacher availability).  
 - Start with a basic version using dummy data.

- Validate your logic with simple test cases.

### Day 12: Create REST API Endpoints with DRF

- Build API endpoints to:  
 - Trigger timetable generation.  
 - Fetch schedules for different user roles.  
 - Handle manual adjustments.

- Ensure responses are in JSON format for React to consume.

### Day 13: Write Unit Tests for Backend Functionality

- Create tests for the scheduling algorithm and API endpoints.

- Use these tests to refine your logic and ensure reliability.

### Day 14: Code Review & Documentation Update

- Review and refactor your Django code.

- Update your design document with any changes or improvements.

## Week 3: Frontend Development with React & Integration

### Day 15: Set Up React Environment & Learn Basics

- Create a new React app using Create React App.

- Familiarize yourself with React fundamentals:  
 - Components, state, props, hooks.

- Learn how to fetch data from your Django API (using Axios or Fetch API).

### Day 16-17: Design UI Wireframes & Component Architecture

- Sketch wireframes for key pages:  
 - Admin dashboard, timetable display, and manual adjustment forms.

- Plan your React component hierarchy and state management strategy.

### Day 18-19: Develop Core React Components

- Build key components (e.g., timetable display, forms for manual changes).

- Set up routing with React Router to handle different views (admin, faculty, student).

### Day 20: Integrate React with Django API

- Connect your React components to the Django REST API endpoints.

- Ensure data flows correctly:  
 - Fetch timetable data and display it.  
 - Submit form data to update schedules.

### Day 21: UI Polishing & State Management Refinement

- Enhance the UI using styling libraries (Bootstrap or Material-UI).

- Refine state management:  
 - Consider Context API or Redux if your state grows complex.

- Test user interactions and refine as needed.

## Week 4: Testing, Debugging, Documentation & Final Touches

### Day 22-23: Comprehensive Integration Testing

- Perform end-to-end tests to ensure the frontend and backend work together.

- Test key user flows for all roles (admin, faculty, student).

### Day 24: Debugging & Optimization

- Debug issues found during testing.

- Optimize the scheduling algorithm and address any performance bottlenecks.

- Polish both backend and frontend code.

### Day 25: Documentation

- Write user guides for the system (how to use ATMA).

- Document developer notes:  
 - Setup instructions, API endpoints, and architecture overview.

- Ensure your code is well-commented.

### Day 26: Deployment Preparation

- Outline your deployment strategy:  
 - Use a platform like Heroku, AWS, or DigitalOcean for Django.  
 - Build your React app (create a production build) and configure Django to serve static files.

- Test deployment in a staging environment.

### Day 27-28: Buffer Time & Mentor Feedback

- Reserve these days for unexpected issues or to incorporate mentor feedback.

- Iterate on features and fix any last-minute bugs.

### Day 29: Final Testing & Polish

- Conduct a final round of testing:  
 - Check UI responsiveness, API integration, and overall performance.

- Finalize any documentation and ensure a smooth user experience.

### Day 30: Wrap-Up & Submission Preparation

- Commit all final changes and tag the release version.

- Prepare any presentation/demo materials.

- Reflect on the process and note improvements for future projects.