### DMS RoadMap

### Phase 1: Mastering Node.js and Express Basics

**1.1 Set Up Your Development Environment**

* **Task**: Install Node.js and npm (Node Package Manager) if not already installed.
* **Action**: Create a new directory for your project and initialize it with npm init to generate the package.json file.

**1.2 Learn Node.js Core Concepts**

* **Task**: Dive deeper into Node.js to understand how it works outside the browser.
* **Topics to Cover**:
  + **File system module (fs)**: Learn how to interact with files (e.g., read/write data).
  + **HTTP module**: Understand how Node.js handles HTTP requests and responses directly.
  + **Event loop**: Learn how Node.js is asynchronous and handles concurrency.

**1.3 Learn Express Basics**

* **Task**: Set up a basic **Express** server.
* **Topics to Cover**:
  + **Routing**: Create basic routes (GET, POST, PUT, DELETE).
  + **Middleware**: Learn how to use middleware for tasks like logging, authentication, etc.
  + **Request/Response cycle**: Understand how to handle incoming requests and send responses.

**Phase 2: Build Basic CRUD Operations for Core Features**

With Express set up, let’s start by building some of the core functionalities of the **Discipline Management System**:

**2.1 Set Up Basic Routes and Controllers**

* **Task**: Create routes to manage **users** and **infractions**.
* **Action**:
  + **Users**: Create a route to add and retrieve users (students, teachers, and admins).
  + **Infractions**: Create a route to log infractions and fetch a list of infractions for each user.

**2.2 Use a Database for Persistence**

* **Task**: Learn about connecting a database to your app.
  + **Option 1: MongoDB (NoSQL)**: Use MongoDB to store data. Install **Mongoose** for handling data models and interactions.
  + **Option 2: SQL (Relational)**: If you prefer relational databases, you can use **MySQL** or **PostgreSQL**.
* **Action**: Create models for **users** and **infractions** in the database:
  + For **users**: Include fields like name, role (admin, teacher, student), email, and password.
  + For **infractions**: Include fields like student ID, infraction type, date, and severity.

**Phase 3: Implement User Authentication and Authorization**

Security is important, especially when dealing with user data and roles (admin, teacher, student). Let’s implement **JWT (JSON Web Tokens)** for authentication:

**3.1 Implement JWT Authentication**

* **Task**: Learn how to implement **JWT-based authentication** to allow users to log in securely.
* **Action**:
  + Create a **signup** route where users can register.
  + Create a **login** route where users can authenticate and get a JWT token.
  + Implement **middleware** to protect certain routes (e.g., admin routes) and ensure only authenticated users can access them.

**3.2 Role-based Authorization**

* **Task**: Implement role-based access control (RBAC) to distinguish between **admin**, **teacher**, and **student**.
* **Action**: Create middleware to restrict access based on roles. For example, only **admin** can view and manage all infractions.

**Phase 4: Implement Features for Discipline Management**

With the authentication in place, let’s focus on the key features of the Discipline Management System:

**4.1 Infractions Logging and Penalties**

* **Task**: Create a route to log infractions and assign severity levels (e.g., minor, moderate, major).
* **Action**: Set up the ability to **add**, **update**, and **delete** infractions.
  + Each infraction should include details such as **student ID**, **infraction type**, **severity**, **date**.
  + Implement logic for **assigning points** (positive or negative) based on the infraction.

**4.2 Points/Rewards System**

* **Task**: Develop a **points/rewards system** where students earn or lose points based on their behavior.
* **Action**:
  + Deduct points for negative infractions and award points for good behavior (optional).
  + Display the points on the user’s dashboard (Admin, Teacher, or Student).

**Phase 5: Create Admin and Teacher Dashboards**

The **Admin** and **Teacher** dashboards will be the main user interfaces where users interact with the system.

**5.1 Admin Dashboard**

* **Task**: Create routes and functionality for **admins** to:
  + View and manage all users.
  + View detailed reports of infractions and behavior trends.
  + Apply penalties and rewards.

**5.2 Teacher Dashboard**

* **Task**: Create routes for **teachers** to:
  + Log new infractions.
  + View behavior data for their assigned students.
  + Send notifications or alerts to students or parents if needed.

**Phase 6: Real-Time Notifications and Reports**

Adding **real-time notifications** and generating **reports** for admins and teachers can enhance the system.

**6.1 Implement Real-Time Notifications (Optional)**

* **Task**: Use **Socket.io** or **WebSockets** to implement real-time notifications for important updates (e.g., infractions or rewards).
* **Action**: Send immediate updates to users (admins/teachers) when actions such as logging an infraction or awarding points are performed.

**6.2 Generate Reports for Admins**

* **Task**: Create a report generation system that allows **admins** to generate and view detailed reports of student behavior over time.
* **Action**: Allow admins to filter reports based on severity, student, and date range.

**Phase 7: Testing and Deployment**

**7.1 Testing**

* **Task**: Write tests for your application to ensure it works as expected.
  + **Backend Testing**: Write tests for your routes (e.g., testing the user authentication, infraction logging).
  + Use testing frameworks like **Jest** or **Mocha** to automate your tests.

**7.2 Deployment**

* **Task**: Deploy your application to a cloud platform like **Heroku** or **Vercel**.
* **Action**:
  + Set up your production environment and database (MongoDB Atlas or a cloud SQL solution).
  + Make sure your application works online and is accessible by users.

**Phase 8: Refinement and Adding Advanced Features**

Once your core system is working, you can refine it and add extra features:

**8.1 Improve UI/UX**

* **Task**: Refine the user interface to make it more user-friendly.
* **Action**: Use frontend libraries like **React** (or pure HTML/CSS) to create more polished dashboards and forms for logging infractions.

**8.2 Implement Advanced Features**

* **Task**: Add additional features as you learn more.
  + **Calendar Integration**: Schedule events like detentions or meetings.
  + **Bulk Import**: Allow admins to bulk upload student data or infractions.
  + **Advanced Reports**: Implement more complex reporting features (e.g., trends analysis).