**School Vaccination Portal Documentation**

**Submitted by**

**Arpit Saxena**

**BITS ID** [**2024TM93225@wilp.bits-pilani.ac.in**](mailto:2024TM93225@wilp.bits-pilani.ac.in)

**GitHub Link:** [**https://github.com/arpitsaxen/FSD\_Assignment.git**](https://github.com/arpitsaxen/FSD_Assignment.git)

**1. System Overview**

The School Vaccination Portal is a comprehensive web application designed to manage and track vaccination drives in educational institutions. The system enables school coordinators to:

* Manage student records
* Schedule and track vaccination drives
* Update vaccination statuses
* Generate comprehensive reports

The application follows a full-stack architecture with a Django REST API backend and a React-based frontend, providing a seamless user experience for coordinating school vaccination efforts.

**Key Features**

* **User Authentication**: Secure JWT-based authentication for school coordinators
* **Dashboard**: Real-time metrics and insights about vaccination status
* **Student Management**: Individual and bulk student record management
* **Vaccination Drive Management**: Planning and scheduling vaccination drives
* **Report Generation**: Customizable reports with filtering options and export capabilities

**2. Application Architecture**

The application follows a client-server architecture with a clear separation between the frontend and backend:

**A diagram of a software developer

AI-generated content may be incorrect.**

**Backend Architecture**

The Django backend follows a modular structure with separate apps for different aspects of the system:

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

**Frontend Architecture**

The React frontend follows a component-based architecture with context for state management:

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Frontend-Backend Interaction**

The frontend communicates with the backend using RESTful API calls. All data interchange happens through JSON, and authentication is handled via JWT tokens:

1. **Authentication Flow**:
   * User submits login credentials
   * Backend validates and returns JWT access and refresh tokens
   * Frontend stores tokens in localStorage
   * Tokens are included in Authorization header for subsequent requests
   * Refresh tokens are used to obtain new access tokens when they expire
2. **Data Fetching**:
   * React components request data via Axios API utilities
   * API endpoints return JSON data
   * React components update state and re-render with new data
3. **Error Handling**:
   * Backend validation errors are sent to frontend
   * Frontend displays appropriate error messages
   * Network errors are handled gracefully

**3. Database Schema**

The application uses a relational database with the following schema:

**A diagram of a student vaccination

AI-generated content may be incorrect.**

**Entity Relationships**

1. **User** (Custom user model extending Django's AbstractUser)
   * Primary user entity representing school coordinators
   * Has is\_coordinator boolean field
2. **Student**
   * Represents a student with personal and class information
   * No direct relationship to User model
3. **Vaccine**
   * Represents types of vaccines available
   * Simple model with name and description
4. **VaccinationDrive**
   * Represents a scheduled vaccination event
   * Associated with a specific vaccine
   * Contains date, doses available, applicable grade ranges
5. **StudentVaccination**
   * Junction table representing a student receiving a vaccine
   * Links Student and VaccinationDrive models
   * Records the date of vaccination and optional notes
   * Has unique\_together constraint to prevent duplicate vaccinations

**4. API Documentation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/auth/login/ | POST | User login | { "username": "string", "password": "string" } | { "access": "string", "refresh": "string", "user": {...} } |
| /api/auth/refresh/ | POST | Refresh access token | { "refresh": "string" } | { "access": "string" } |

**Authentication Endpoints**

**Student Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/students/ | GET | List all students |  | Array of student objects |
| /api/students/ | POST | Create a new student | Student data | Created student object |
| /api/students/{id}/ | GET | Get a student by ID |  | Student object |
| /api/students/{id}/ | PUT | Update a student | Student data | Updated student object |
| /api/students/{id}/ | DELETE | Delete a student |  |  |
| /api/students/bulk\_import/ | POST | Import students from CSV | Form data with CSV file | { "message": "string", "errors": [] } |
| /api/students/export/ | GET | Export students as CSV |  | CSV file |
| /api/students/template/ | GET | Get CSV template |  | CSV template file |

**Vaccination Drive Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/drives/ | GET | List all vaccination drives |  | Array of drive objects |
| /api/drives/ | POST | Create a new drive | Drive data | Created drive object |
| /api/drives/{id}/ | GET | Get a drive by ID |  | Drive object |
| /api/drives/{id}/ | PUT | Update a drive | Drive data | Updated drive object |
| /api/drives/{id}/ | DELETE | Delete a drive |  |  |
| /api/drives/{id}/mark\_students/ | POST | Mark students as vaccinated | { "student\_ids": [] } | { "message": "string", "errors": [] } |

**Vaccine Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/vaccines/ | GET | List all vaccines |  | Array of vaccine objects |
| /api/vaccines/ | POST | Create a new vaccine | Vaccine data | Created vaccine object |
| /api/vaccines/{id}/ | GET | Get a vaccine by ID |  | Vaccine object |
| /api/vaccines/{id}/ | PUT | Update a vaccine | Vaccine data | Updated vaccine object |
| /api/vaccines/{id}/ | DELETE | Delete a vaccine |  |  |

**Vaccination Records Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/vaccinations/ | GET | List all vaccination records |  | Array of vaccination objects |
| /api/vaccinations/ | POST | Create a vaccination record | Vaccination data | Created vaccination object |
| /api/vaccinations/{id}/ | GET | Get a vaccination by ID |  | Vaccination object |
| /api/vaccinations/check\_eligibility/ | POST | Check student eligibility | { "student\_ids": [], "drive\_id": int } | Eligibility results |

**Report Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Query Parameters** | **Response** |
| /api/reports/dashboard\_stats/ | GET | Get dashboard statistics |  | Stats object |
| /api/reports/vaccination\_report/ | GET | Get vaccination report | vaccine\_id, grade, start\_date, end\_date, format | Report data or CSV file |

**5. Frontend Components**

**Authentication Components**

1. **Login.js** - Login form for school coordinators
2. **PrivateRoute.js** - Route protection for authenticated users

**Layout Components**

1. **Layout.js** - Main layout with sidebar navigation and header

**Dashboard**

1. **Dashboard.js** - Main dashboard with key metrics and quick links

**Student Management**

1. **Students.js** - Student listing with search, filtering, and bulk import
2. **StudentForm.js** - Form for adding/editing individual students

**Vaccination Drive Management**

1. **VaccinationDrives.js** - Listing of vaccination drives with tabs for upcoming/past
2. **DriveForm.js** - Form for adding/editing drives and managing vaccinations

**Vaccine Management**

1. **Vaccines.js** - Interface for managing vaccine types

**Reporting**

1. **Reports.js** - Vaccination report generation with filters and export

**6. Setup Instructions**

**Backend Setup**

1. **Clone the Repository**

git clone https://github.com/yourusername/school-vaccination-portal.git

cd school-vaccination-portal

1. **Set Up Virtual Environment**

python -m venv venv

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

1. **Install Dependencies**

pip install -r requirements.txt

1. **Configure Environment Variables (Optional)**

Create a .env file in the project root:

SECRET\_KEY=your\_secret\_key

DEBUG=True

ALLOWED\_HOSTS=localhost,127.0.0.1

1. **Run Migrations**

python manage.py migrate

1. **Create Admin User**

python manage.py setup\_admin

# This creates a default admin user (username: admin, password: admin)

1. **Generate Sample Data (Optional)**

python manage.py generate\_sample\_students --per\_grade 100

1. **Start the Django Server**

python manage.py runserver

The backend will be available at <http://localhost:8000/>

**Frontend Setup**

1. **Navigate to Frontend Directory**

cd frontend

1. **Install Dependencies**

npm install

1. **Configure Environment Variables**

Create a .env file in the frontend directory:

REACT\_APP\_API\_URL=http://localhost:8000

1. **Start the Development Server**

npm start

The frontend will be available at <http://localhost:3000/>

1. **Build for Production (Optional)**

npm run build

**7. Assumptions and Limitations**

1. **Authentication**
   * The system assumes a single user role (school coordinator)
   * Authentication is simulated with JWT tokens and can be extended for production
2. **Student Management**
   * Students are uniquely identified by student\_id
   * CSV import assumes specific column formats
   * Grade values are stored as strings to support different formats
3. **Vaccination Drive Management**
   * Drives must be scheduled at least 15 days in advance
   * Past drives cannot be edited
   * Assumes no conflicting drives on the same date for the same vaccine
4. **Reporting**
   * Reports assume straightforward vaccination statuses
   * Only CSV export is currently supported
5. **General Limitations**
   * SQLite database for development (should be replaced with PostgreSQL for production)
   * Limited security features for a demonstration system
   * No email notifications or integrations with external systems

**8. Future Enhancements**

1. **Authentication and User Management**
   * Multiple user roles (administrators, nurses, parents)
   * Enhanced security features (2FA, password reset)
2. **Student Management**
   * Parent/guardian information and contact details
   * Medical history and contraindications
   * Photo upload capabilities
3. **Vaccination Drive Management**
   * Automated notifications for upcoming drives
   * Resource allocation (nurses, equipment)
   * QR code-based check-in system
4. **Reporting and Analytics**
   * Advanced analytics dashboard with visualizations
   * Predictive modeling for vaccination coverage
   * Export in multiple formats (PDF, Excel)
5. **Integration**
   * Integration with school management systems
   * Mobile app for on-the-go management
   * Public portal for parents to view vaccination status

UI/UX Screenshots.

Login Page:

<http://127.0.0.1:3000>

A screenshot of a computer

AI-generated content may be incorrect.

Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Student List

A screenshot of a computer

AI-generated content may be incorrect.

Student Form

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Vaccination Drives

A screenshot of a computer

AI-generated content may be incorrect.

Vaccines

A screenshot of a computer

AI-generated content may be incorrect.

Add Student to Drive

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Report

A screenshot of a computer

AI-generated content may be incorrect.