PremedHQ Ontario - Setup Guide

A comprehensive premed tracking application designed specifically for Ontario medical school applicants.

Features

- **User Authentication** - Secure registration and login system
- **GPA Tracker** - Monitor cumulative and 2-year GPA with Ontario credit system
- **Test Tracking** - Track MCAT, CASPer, and other test scores
- **Activity Logger** - Record extracurricular activities with journal entries
- **Application Manager** - Monitor OMSAS application status
- **Timeline Planner** - Plan important premed milestones
- **Dashboard** - Overview of your premed journey

Quick Start

Prerequisites

- Node.js (v16 or higher)
- MongoDB (local installation or MongoDB Atlas account)
- Git

Backend Setup

1. **Clone and setup backend:**

```bash
mkdir premed-hq-backend
cd premed-hq-backend

# Copy the server.js file into this directory
# Copy the package.json file into this directory
```

2. **Install dependencies:**

```bash
npm install
```

3. **Environment Setup:** Create a `.env` file in the backend directory:

```bash
# Required environment variables
MONGODB_URI=mongodb://localhost:27017/premed-hq
JWT_SECRET=your-super-secret-jwt-key-here
PORT=5000
NODE_ENV=development
```

For MongoDB Atlas (recommended for production):

```bash
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/premed-hq
```

4. **Start MongoDB:**

Local MongoDB:

```bash
# On macOS with Homebrew:
brew services start mongodb/brew/mongodb-community

# On Ubuntu:
sudo systemctl start mongod

# On Windows:
# Start MongoDB service from Services panel
```

Or use MongoDB Atlas:

- Create account at MongoDB Atlas
- Create a new cluster
- Get connection string and update `MONGODB_URI`

5. **Start the backend server:**

```bash
```

```
# Development mode (with auto-restart)
npm run dev

# Production mode
npm start
```

The backend will run on `http://localhost:5000`

Frontend Setup

1. **Setup React frontend:**

```
bash

# Create React app
npx create-react-app premed-hq-frontend
cd premed-hq-frontend

# Install required dependencies
npm install lucide-react

# Copy the frontend React component code into src/App.js
# Replace the default App.js content with the provided React component
```

2. **Update API URL:** In your React component, make sure the API_BASE_URL points to your backend:

```
javascript

const API_BASE_URL = 'http://localhost:5000/api';
```

3. **Start the frontend:**

```
bash

npm start
```

The frontend will run on `http://localhost:3000`

Database Setup

The application will automatically create the necessary collections when you first use it. No manual database setup is required.

Collections Created:

- `users` - User accounts
- `courses` - Academic courses and grades
- `activities` - Extracurricular activities
- `tests` - Test scores (MCAT, CASPer, etc.)
- `applications` - Medical school applications
- `journals` - Journal entries
- `timelines` - Timeline events

Testing the Application

1. **Access the application:**
   - Navigate to `http://localhost:3000`
   - Create a new account or use the demo login
2. **Demo Account:**
   - The application includes a demo login feature
   - Check the login component for demo credentials
3. **Test Features:**
   - Add courses and see GPA calculations
   - Track test scores and dates
   - Log extracurricular activities
   - Plan your timeline

API Endpoints

Authentication

- `POST /api/register` - Create new account
- `POST /api/login` - Login user

Courses

- `GET /api/courses` - Get user's courses
- `POST /api/courses` - Add new course
- `PUT /api/courses/:id` - Update course
- `DELETE /api/courses/:id` - Delete course

Tests

- `GET /api/tests` - Get user's tests
- `POST /api/tests` - Add new test
- `PUT /api/tests/:id` - Update test
- `DELETE /api/tests/:id` - Delete test

Activities

- `GET /api/activities` - Get user's activities
- `POST /api/activities` - Add new activity
- `PUT /api/activities/:id` - Update activity
- `DELETE /api/activities/:id` - Delete activity

Applications

- `GET /api/applications` - Get user's applications
- `POST /api/applications` - Add new application
- `PUT /api/applications/:id` - Update application
- `DELETE /api/applications/:id` - Delete application

Journal

- `GET /api/journal` - Get user's journal entries
- `POST /api/journal` - Add new journal entry
- `PUT /api/journal/:id` - Update journal entry
- `DELETE /api/journal/:id` - Delete journal entry

Timeline

- `GET /api/timeline` - Get user's timeline events
- `POST /api/timeline` - Add new timeline event
- `PUT /api/timeline/:id` - Update timeline event
- `DELETE /api/timeline/:id` - Delete timeline event

Dashboard

- `GET /api/dashboard` - Get all user data for dashboard

Security Features

- JWT-based authentication
- Password hashing with bcrypt
- User data isolation (users can only access their own data)
- Input validation
- CORS protection

Deployment

Backend Deployment (Railway/Heroku)

1. **Environment Variables:**

```bash
MONGODB_URI=your-production-mongodb-uri
JWT_SECRET=your-production-jwt-secret
NODE_ENV=production
```

2. **Deploy to Railway:**

```bash
# Install Railway CLI
npm install -g @railway/cli

# Login and deploy
railway login
railway link
railway up
```

Frontend Deployment (Netlify/Vercel)

1. **Build for production:**

```bash
npm run build
```

2. **Update API URL:** Update `API_BASE_URL` to point to your deployed backend.

3. **Deploy to Netlify:**

- Connect your GitHub repository
- Set build command: `npm run build`
- Set publish directory: `build`

Troubleshooting

Common Issues

1. **MongoDB Connection Error:**
   - Ensure MongoDB is running
   - Check your `MONGODB_URI` in `.env`
   - For Atlas, whitelist your IP address
2. **CORS Errors:**
   - Ensure backend is running on port 5000
   - Check that API_BASE_URL matches your backend URL
3. **Authentication Issues:**
   - Check JWT_SECRET is set
   - Clear localStorage and try again
   - Check browser network tab for API errors
4. **Form Submission Not Working:**
   - Check browser console for errors
   - Ensure all required fields are filled
   - Verify API endpoints are accessible

Logs and Debugging

- Backend logs: Check your terminal where you ran `npm run dev`
- Frontend logs: Check browser console (F12)
- Network issues: Check browser Network tab

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly

5. Submit a pull request

License

MIT License - see LICENSE file for details

Support

For questions and support:

- Check the troubleshooting section
- Review the API documentation
- Check browser console for errors
- Verify backend server is running

Happy pre-med tracking! 🩺📚