MedFlow - Medical Appointment Booking System

A full-stack web application that streamlines the appointment booking process between patients and doctors. Built with Node.js, Express, MySQL, and EJS templates.

🚀 Features

For Patients

- ✅ User registration and secure authentication
- 🔍 Browse doctors by specialty
- 📅 Book appointments with preferred doctors
- 📊 View appointment history and status
- 📧 Receive email notifications for updates

For Doctors

- 🔐 Secure login system
- 📋 View all pending appointment requests
- ✅ Approve or decline appointments
- 📆 Manage appointment schedule
- 🔔 Automatic notifications for new bookings

System Features

- 🔒 JWT-based authentication
- 📫 Automated email notifications
- 💾 MySQL relational database
- 🔐 Password encryption with bcrypt
- 🎨 Responsive UI with Bootstrap 5
- 🚀 Server-side rendering with EJS

📋 Prerequisites

Before you begin, ensure you have installed:

- **Node.js** (v14 or higher)

- **MySQL** (v5.7 or higher)
- **npm** or **yarn**

🛠️ Installation

1. Clone the repository

```bash
git clone https://github.com/yourusername/medflow.git
cd medflow
```

2. Install dependencies

```bash
npm install
```

3. Setup Database

```bash
# Login to MySQL
mysql -u root -p

# Create database and tables
source database/schema.sql

# Or manually create:
mysql -u root -p < database/schema.sql
```

4. Configure Environment Variables

```bash
# Copy the example env file
cp .env.example .env

# Edit .env with your configurations
nano .env
```

Required environment variables:

```env
NODE_ENV=development
PORT=3000

# Database
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=your_mysql_password
DB_NAME=medflow

# JWT
JWT_SECRET=your_super_secret_key_min_32_chars
JWT_EXPIRES_IN=24h

# Email (Gmail example)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your_email@gmail.com
EMAIL_PASSWORD=your_app_specific_password

# Frontend URL
FRONTEND_URL=http://localhost:3000
```

5. Setup Gmail for Email Notifications (Optional but Recommended)

1. Go to your Google Account settings
2. Enable 2-Factor Authentication
3. Generate an App Password:
   - Go to Security → 2-Step Verification → App passwords
   - Select "Mail" and "Other (Custom name)"
   - Copy the 16-character password
   - Use this as your `EMAIL_PASSWORD` in .env

6. Generate Sample Data (Optional)

```bash
```

```
# The schema.sql already includes sample doctors
# To create a test patient account, register through the app
# Or use this bcrypt hashed password: password123
```

7. Start the Server

```bash
# Development mode with auto-restart
npm run dev

# Production mode
npm start
```

The application will be available at `http://localhost:3000`

📁 Project Structure

```
medflow/
├── backend/
│   ├── config/
│   │   └── db.js              # MySQL configuration
│   ├── src/
│   │   ├── controllers/
│   │   │   ├── authController.js
│   │   │   ├── appointmentController.js
│   │   │   ├── doctorController.js
│   │   │   └── viewController.js
│   │   ├── models/
│   │   │   ├── User.js
│   │   │   └── Appointment.js
│   │   ├── routes/
│   │   │   ├── authRoutes.js
│   │   │   ├── appointmentRoutes.js
│   │   │   ├── doctorRoutes.js
│   │   │   └── viewRoutes.js
│   │   ├── middleware/
│   │   │   ├── authMiddleware.js
│   │   │   └── roleMiddleware.js
│   │   └── utils/
│   │       └── emailService.js
│   ├── views/                 # EJS templates (to be created)
│   ├── public/              # Static assets (CSS, JS, images)
│   ├── database/
│   │   └── schema.sql
│   ├── server.js
│   ├── .env
│   ├── .env.example
│   ├── .gitignore
│   ├── package.json
│   └── README.md
```

🔌 API Endpoints

Authentication

```
POST   /api/auth/register    - Register new user
POST   /api/auth/login       - Login user
POST   /api/auth/logout      - Logout user
GET    /api/auth/me          - Get current user
GET    /api/auth/verify      - Verify JWT token
```

Doctors

```
GET    /api/doctors          - Get all doctors
GET    /api/doctors/:id      - Get doctor by ID
GET    /api/doctors/specialty/:specialty - Get doctors by specialty
GET    /api/doctors/specialties - Get all specialties
```

Appointments (Patient)

```
POST   /api/appointments     - Create appointment
GET    /api/appointments/my-appointments - Get patient appointments
GET    /api/appointments/:id - Get appointment details
DELETE /api/appointments/:id - Cancel appointment
```

Appointments (Doctor)

```
GET    /api/appointments/doctor/appointments - Get all appointments
GET    /api/appointments/doctor/pending - Get pending appointments
GET    /api/appointments/doctor/stats - Get statistics
```

```
PATCH  /api/appointments/:id/approve - Approve appointment
PATCH  /api/appointments/:id/decline - Decline appointment
```

Views (Server-side rendered)

```
GET    /                    - Home page
GET    /login               - Login page
GET    /register            - Register page
GET    /patient/dashboard   - Patient dashboard
GET    /doctor/dashboard    - Doctor dashboard
GET    /doctors             - Doctors listing
GET    /book-appointment/:doctorId - Book appointment form
GET    /appointment/:id     - Appointment details
```

🧪 Testing

```bash
# Run tests (when implemented)
npm test

# Run with coverage
npm run test:coverage
```

📧 Email Templates

The system sends automated emails for:

1. **Welcome Email** - New user registration
2. **Appointment Confirmation** - Patient books appointment
3. **Doctor Notification** - New appointment request
4. **Approval Email** - Doctor approves appointment
5. **Decline Email** - Doctor declines appointment
6. **Reminder Email** - 24 hours before appointment

🔒 Security Features

- **Password Hashing**: bcrypt with 10 salt rounds
- **JWT Authentication**: Secure token-based auth

- **HTTP-only Cookies**: XSS protection
- **Role-based Access Control**: Patient/Doctor permissions
- **SQL Injection Prevention**: Parameterized queries
- **Input Validation**: Server-side validation

🎨 Creating Views (Next Steps)

You'll need to create EJS templates in the `views/` folder:

```
views/
├── partials/
│   ├── header.ejs
│   ├── navbar.ejs
│   └── footer.ejs
├── index.ejs
├── login.ejs
├── register.ejs
├── error.ejs
├── patient/
│   ├── dashboard.ejs
│   ├── doctors.ejs
│   ├── book-appointment.ejs
│   └── appointment-details.ejs
└── doctor/
    ├── dashboard.ejs
    └── appointment-details.ejs
```

🚀 Deployment

Environment Setup

1. Set `NODE_ENV=production` in .env
2. Use strong JWT_SECRET (minimum 32 characters)
3. Configure production database
4. Setup SSL certificates
5. Use process manager (PM2)

PM2 Deployment

```bash
npm install -g pm2
pm2 start server.js --name medflow
pm2 save
pm2 startup
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch: `git checkout -b feature-name`
3. Commit changes: `git commit -m 'Add feature'`
4. Push to branch: `git push origin feature-name`
5. Submit a pull request

📝 License

This project is licensed under the MIT License.

👥 Authors

Your Name - @yourhandle

🙏 Acknowledgments

- Express.js team
- MySQL community
- Nodemailer contributors
- Bootstrap team

📞 Support

For support, email support@medflow.com or create an issue in the repository.

Made with ❤️ by MedFlow Team