EventBookingSystem Frontend Documentation

Project Structure

```
frontend/
│
├── index.html                # Main landing page
├── js/                       # JavaScript files
│   ├── auth.js               # Handles authentication (login, register,
token, UI updates)
│   ├── events.js             # Handles event listing, booking, and event
actions
│   ├── mybookings.js         # Handles the 'My Bookings' page logic
│   ├── main.js               # General UI logic, notifications, helpers
│   └── config.js             # API base URL and config
│
├── css/
│   └── styles.css            # Main stylesheet
│
├── pages/                    # Sub-pages
│   ├── login.html            # Login page
│   ├── signup.html           # Registration page
│   └── mybookings.html       # User's bookings page
│
├── .vscode/
│   └── launch.json           # VS Code launch config (for Live Server or
Chrome)
│
└── launch.json               # (duplicate, can be ignored if using
.vscode/launch.json)
```

How to Run the Frontend

1. Open a terminal and navigate to the 'frontend' directory:
   ```
   cd path/to/EventBookingSystem/frontend
   ```

2. Start a static server:
   - With Node.js: `npx live-server`
   - With Python: `python -m http.server 8080`
   - Or use the VS Code Live Server extension (right-click index.html → "Open with Live Server").

3. Open your browser and go to:
   http://localhost:8080/
   - Main page: /index.html
   - Login: /pages/login.html
   - Signup: /pages/signup.html
   - My Bookings: /pages/mybookings.html

How the Frontend Works

Authentication
- Login and registration are handled in auth.js.
- On login, the JWT token and user info are saved in localStorage.
- The UI updates to show the user's name and a logout button when logged in.

- Token expiry is checked on every page load; expired tokens log the user out automatically.

API Integration
- The frontend communicates with the backend API using fetch (or authFetch for authenticated requests).
- The API base URL is set in js/config.js.
- Protected endpoints (like booking creation) include the JWT token in the Authorization header.

Pages
- index.html: Shows the main event listing and navigation.
- login.html: Login form, calls /api/Accounts/login.
- signup.html: Registration form, calls /api/Accounts/Register.
- mybookings.html: Placeholder for user bookings (requires login).

JavaScript Files
- auth.js: Handles login, registration, logout, token management, and UI updates.
- events.js: Fetches and displays events, handles booking, and event management actions.
- mybookings.js: Handles logic for displaying user bookings (currently a placeholder).
- main.js: General UI helpers, notifications, and enhancements.
- config.js: Stores the API base URL.

Styling
- All pages use css/styles.css for consistent, modern styling.
- Google Fonts and Font Awesome are used for typography and icons.

VS Code Launch Configuration
- .vscode/launch.json can be used to launch Chrome or Live Server for development.

How to Connect with the Backend

- The frontend expects the backend API to be running and accessible at the URL specified in js/config.js.
- All API requests are made to endpoints like /api/Accounts/login, /api/Events, /api/Bookings/CreateBooking, etc.
- JWT tokens are required for protected endpoints and are automatically attached by the frontend.

Troubleshooting

- 404 errors for JS/CSS: Make sure you are running the server from the frontend directory and all files exist in their correct folders.
- CORS issues: The backend must allow requests from the frontend's origin (handled by CORS policy in the backend).
- Token/auth issues: Ensure the backend is running and the API base URL is correct in config.js.

Summary

- The frontend is a static site (HTML, CSS, JS) that interacts with the backend API for authentication, event management, and bookings.
- It is easy to run locally with any static server.
- All code is modular and organized for maintainability.