Event Planning App – Codebase Explained

1. Project Overview

This is a full-stack event planning application with user/admin authentication, event management, and a modern dashboard UI. Built with Next.js (React), Node.js API routes, and MongoDB.

---

2. Directory Structure

```
/ (root)
├── app/              # Next.js app directory (frontend + API routes)
│   ├── api/          # API endpoints (auth, events, users)
│   ├── dashboard/    # Dashboard page and components
│   ├── events/       # Event-related pages (create, edit, details)
│   ├── contexts/     # React context (e.g., AuthContext)
│   └── ...
├── models/           # Mongoose models (Event, User)
├── lib/              # Utility functions (e.g., db connection, JWT)
├── package.json      # Project dependencies
└── ...
```

---

3. Backend/API

Authentication

- **JWT-based:** Login returns a token, which is used for protected API calls.
- **API routes:** /api/auth/login, /api/auth/register, /api/auth/me
- **Role-based:** User and admin roles checked in API middleware.

Event Management

- **CRUD endpoints:** /api/events (GET, POST), /api/events/[id] (GET, PUT, DELETE)
- **Validation:** Uses Zod to validate event data.
- **Overlap prevention:** Backend checks for overlapping events for the same organizer.

Example: Event API Route (POST)

```
// app/api/events/route.ts
import { createEvent } from '../../../services/eventService';
export async function POST(req) {
  // Parse and validate request body
  // Check for overlaps
  // Save event to DB
  // Return created event or error
}
```

---

4. Frontend

Main Pages/Components

- **Dashboard:** app/dashboard/page.tsx – Calendar, event cards, stats widgets
- **Event Card:** Renders event image, title, details, and action buttons
- **Event Details:** /events/[id]/page.tsx – Shows full event info
- **Admin Pages:** /admin/events, /admin/users (if implemented)

Data Fetching

- Uses fetch to call API endpoints (with JWT in headers)
- Data is grouped and displayed by date in the dashboard

Role-based UI

- Admins see extra widgets and controls (edit/delete, stats)
- Regular users see only their events

---

5. Key Code Snippets

Event Model (Mongoose)

```
// models/Event.js
const EventSchema = new mongoose.Schema({
  title: String,
  description: String,
  date: String,
  time: String,
  duration: Number,
  location: String,
  category: String,
  status: String,
  organizer: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  image: String,
  ...
});
```

Dashboard Event Card (React/JSX)

```
<div className="bg-white border rounded-2xl p-8 shadow-md">
  <img src={event.image} alt="Event image" />
  <h4>{event.title}</h4>
  <p>{event.description}</p>
  <Link href={`/events/${event._id}`}>View Details</Link>
</div>
```

---

6. Data Flow

1. User logs in → receives JWT
2. Frontend stores JWT and uses it for API requests
3. API validates JWT, fetches data from MongoDB
4. Frontend displays data (events, user info, etc.)

---

7. How to Extend or Modify

- **Add new event fields:** Update Event model, validation, and forms
- **Add new API endpoints:** Create new files in app/api/
- **Add new pages/components:** Add to app/ directory
- **Change UI:** Edit React components and Tailwind classes

For more details, see the code comments in each file. This summary is suitable for PDF export and onboarding new developers or demoing the project.