MERN Stack Final Project Proposal

1. Project Title and Description (10 pts)

Project Title

Event Planner: The Social Party Planner

Description

Event Planner is our app designed to make planning parties and social gatherings with friends super easy. Users can sign up and create a profile, build a friends list, and then create, invite, and manage all their events from one place.

2. Planned Features (10 pts)

Here are the four main things users will be able to do:

1. Sign Up and Your Profile: Users can register and log in securely. Once inside, they can manage their profile.
2. Friend Connect: Users can search for and find other people using the app. They can send and accept friend requests. This friends list is the only pool they can pull from to send invitations.
3. Event Maker and RSVP: Anyone can create a new event with a title, date, and description, and then invite friends from their list. Invited friends see the details and can easily RSVP.
4. Home Dashboard (The Feed): This is the main screen when you log in. It will show a personalized list of all your upcoming events—the ones you're hosting and the ones you've been invited to—so you can quickly check the status.

3. Technical Implementation Plan (15 pts)

Technical Stack

We're using the standard MERN stack: React for the user interface, Express.js and Node.js for the server, and MongoDB for the database. Axios is the library we'll use to make sure the frontend and backend can talk to each other.

React Pages and Components

We will have the following main pages and components:

- The entry gates for the app, landing page, logging in, and signing up, is at the root `/` and uses the Home, Login, and Register components.
- The user's home screen showing all their events is at `/dashboard` and uses the Dashboard component.
- Viewing a user's details, searching for new friends, and managing friend requests is at `/profile/:id` and uses the Profile, FriendList, and FriendSearch components.
- Where the user fills out the details to start a new party is at `/event/create` and uses the EventForm component.
- The individual event page, showing who is coming and where to click to RSVP is at `/event/:id` and uses the EventDetails, RSVPButton, and AttendeeList components.

How the Data Moves (Axios to Express)

Axios is essential for moving data around. For example:

- Signing Up: The Register component sends user data using a POST request to the server at `/api/users/register`.
- Making an Event: When a user hits "Create," the EventForm sends the event details and the list of invited friends via POST to `/api/events`.
- Loading the Dashboard: The Dashboard component grabs all the events relevant to the current user via a GET request to `/api/users/:id/events`.

MongoDB Data Structures (Models)

We're going to use two main types of data records:

1. User Data (Stored in the 'users' collection)
    - `username`, `email`, and `password` (all required)
    - `friends`: An array of IDs for people they've successfully befriended.
    - `pendingRequests`: An array of IDs for people who have sent them a friend request.
    - `events`: An array of IDs for all the events they are part of (hosting or invited).
2. Event Data (Stored in the 'events' collection)
    - `title`, `host` ID, `date`, and `location` (all required)
    - `description`
    - `attendees`: An array that holds objects for every invited user, showing their ID and current RSVP status (e.g., Attending, Not Attending, Pending).

The CRUD Operations

Here is how the main Create, Read, Update, and Delete actions are handled:

- Create: Making a new account or a new party uses a POST request to routes like `/api/users/register` or `/api/events`.
- Read: Getting information (like a profile or event details) uses a GET request to routes like `/api/users/:id` or `/api/events/:id`.
- Update: Editing profile details or changing an RSVP status uses a PUT/PATCH request to routes like `/api/users/:id/update` or `/api/events/:id/rsvp`.
- Delete: The host cancelling/deleting an event uses a DELETE request to a route like `/api/events/:id`.

4. Team Member Roles (10 pts)

We have four people, and we've divided the workload to make sure we cover everything:

- Kyler and Paddy are the Lead Frontend Developer, focusing on React Component Development (Authentication & Profile UI) and State Management.
- Christian is the Full-Stack Developer, focusing on Backend Development (Express API Routes), Axios integration, and Database Querying.
- Sydney is the Database & Event Specialist, focusing on MongoDB Schema Design (User/Event Models), CRUD implementation, and Event Feature logic (RSVP system).
- Dami is the Frontend/UX Designer & Documentation Lead, focusing on React Component Development (Forms & Dashboard UI), Styling (Tailwind/CSS), and Final Documentation & Presentation prep.

5. Timeline & Milestones (5 pts)

We'll tackle the project in three stages to make sure the core stuff works before we make it pretty.

- Phase 1: Foundation (Due: [Suggested Date - e.g., Next Friday])
    - Deliverables: MongoDB/Express connection established. Complete User Model and Authentication (register/login) implemented and fully functional. Initial React components drafted.
    - Responsible: Christian Sydney
- Phase 2: Core Features (Due: [Suggested Date - e.g., Two Weeks from Now])
    - Deliverables: Complete Event Model and Event CRUD operations. Event Creation/Viewing fully operational on the frontend. Basic Friend Management (viewing friends list) implemented.
    - Responsible: Kyler, Paddy, Christian, Sydney
- Phase 3: Polish and Presentation (Due: [Suggested Date - e.g., One Week before Final Due Date])

- Deliverables: Final RSVP logic and Friend Request acceptance implemented. Full application styling and responsive design finalized. Comprehensive project documentation and presentation prepared.
- Responsible: Kyler, Paddy, Dami