Product Requirements & Specification Document

Project Name

Eventify - Event Management Backend

Description

Eventify is a backend API for creating and managing events, featuring user authentication and CRUD operations for events. Built with Node.js, Express, and PostgreSQL, it targets startups needing a scalable event management solution.

1. Goals & Objectives

Goal	Description
User Authentication	Secure registration and login for users
Event CRUD	Create, read, update, and delete events
Data Persistence	Store users and events in PostgreSQL
RESTful API	Expose endpoints for all core operations
Scalability & Maintainability	Clean, modular codebase for future enhancements

2. Functional Requirements

2.1 User Management

Feature	Description
Register	Users can register with email & password
Login	Users can log in and receive JWT token
Auth Middleware	Protect event endpoints with JWT auth

2.2 Event Management

Feature	Description
Create Event	Authenticated users can create events
Read Events	List all events or a single event
Update Event	Users can update their own events
Delete Event	Users can delete their own events

3. Non-Functional Requirements

Requirement	Description
Security	Password hashing, JWT authentication
Performance	API responds within 500ms for standard ops
Documentation	API documented via OpenAPI/Swagger
Error Handling	Consistent error responses (JSON)
Code Quality	Follows standard Node.js/Express best practices

---

4. API Endpoints

4.1 Authentication

Method	Endpoint	Description	Auth Required
POST	/api/register	Register user	No
POST	/api/login	Login user	No

4.2 Events

Method	Endpoint	Description	Auth Required
POST	/api/events	Create event	Yes
GET	/api/events	List all events	No
GET	/api/events/:id	Get event by ID	No
PUT	/api/events/:id	Update event (owner only)	Yes
DELETE	/api/events/:id	Delete event (owner only)	Yes

---

5. Data Model

5.1 User

Field	Type	Constraints
id	UUID	PK, auto-generated
email	String	Unique, required
password	String	Hashed, required
created_at	Timestamp	Auto-generated

5.2 Event

Field	Type	Constraints
id	UUID	PK, auto-generated

title	String	Required
description	String	Optional
date	Date	Required
location	String	Optional
owner_id	UUID	FK -> User(id)
created_at	Timestamp	Auto-generated

6. Security

- Passwords hashed with bcrypt
- JWT for authentication (HTTP-only cookies or Authorization header)
- Input validation and sanitization
- Users can only modify their own events

7. Technology Stack

Layer	**Technology**
Language	Node.js (ES6+)
Framework	Express.js
Database	PostgreSQL
Auth	JWT, bcrypt
ORM/Query	Knex.js or Sequelize (optional)
Testing	Jest or Mocha
Documentation	Swagger/OpenAPI

8. Implementation Notes

- Use environment variables for config (e.g., DB, JWT secret)
- Modularize routes, controllers, and models
- Seed initial data for testing
- Provide sample `.env.example` file

9. Out of Scope

- Frontend/UI
- Event invitations, RSVPs, or notifications
- Payment processing

10. Sample API Request

```
POST /api/events
Authorization: Bearer <token>
Content-Type: application/json

{
  "title": "Startup Launch",
  "description": "Product launch event",
  "date": "2024-07-01",
  "location": "San Francisco"
}
```

11. Milestones

Milestone	Description
1. Project Setup	Repo, dependencies, base structure
2. User Auth	Register, login, JWT middleware
3. Event CRUD	All event endpoints
4. Testing & Docs	Unit tests, API docs
5. Deployment	Dockerfile, deployment scripts

12. Acceptance Criteria

- All endpoints function as specified
- Only authenticated users can create/update/delete their events
- API returns appropriate status codes and error messages
- Database persists users and events
- API documentation is complete

End of Document