CSCI 5709 – Advanced Topics in Web Development

Assignment – 2

Repo link: https://github.com/shraymoza/webAssignment2/tree/main/eventsparkapi

Render URI: https://webassignment2-jo5k.onrender.com

Under the guidance of

Dr. Oladapo Oyebode

Faculty of Computer Science

Submitted By: Shray Moza (B00987463)

1. Core-Feature Endpoint Specifications

For every endpoint the following are provided: **Method**, **Route**, **Purpose**, **Request** *(path / query / body)*, **Success Response** *(status & JSON)*, and **Error Conditions**.

1.1 Feature A – Seat-Map Selection & Booking

#	Method	Route	Purpose	Request	Success Status & JSON	Error cases
1	GET	/events/ **eventId**/ seatmap	Return real-time seat layout with pricing & availability.	**Headers,** **eventId** query parameter.	**200** : { eventId: " ", venueId: " ", seats: [{ id: " ", row: " ", col,price: " ", status: " ", }] }	**404**: Event not found. **423:** Venue locked
2	POST	/events/ **eventId**/ bookings	Create a booking for the authenticated user.	**Headers,** **Authorization**: Bearer <JWT>. **Body**: { seatIds: [seats added], paymentIntentId: " " }	**201** : { bookingId: " ", status:" ", expiresAt: " ", }	**400**: invalid seats. **409**: seat already taken. **402**: payment intent invalid.
3	GET	/bookings/ **bookingId**	Fetch booking details.	**bookingId** query parameter.	**200**: { bookingId: " ", seats: " ", total: " ", status: " ", }	**404**: not found.
4	DELETE	/bookings/ **bookingId**	Cancel a booking & release seat.	**bookingId** query parameter.	**204**: { status: " ", }	**404**: not found.
5	PUT	/bookings/ **bookingId**	Amend seats prior to payment captured.	**Headers,** **Body**: { add: [seats added], remove: [seats removed] }	**200:** updated booking JSON	**409**: seat clashes. **422**: past amendment window.

1.2 Feature B – E-Ticket Generation & Validation

#	Method	Route	Purpose	Request	Success Status & JSON	Error cases
1	**GET**	/tickets/**ticketId**	Download ticket PDF.	Query parameter **ticketId.**	**200:** binary PDF stream; Content-Disposition: attachment; filename="ticket_<id>.pdf"	**404**: ticket not found. **403**: not owner.
2	**GET**	/tickets/**ticketId/qrcode**	Retrieve QR code image.	Query parameter **ticketId.**	PNG returned.	**404:** QR not found.

3	**POST**	/tickets/**ticketId**/validate	Venue scanner verifies entry.	**Headers,**
				
				Authorization: Bearer <ScannerJWT>,
				
				Body: {
				scannerId: " ",
				scannedAt: " ",
				}
					200: {
					valid: "BOOLEAN ",
					attendee: {
					name:" ",
					bookingId:" ",
					}
					}
						404: ticket not found.
						409: already scanned.
						401: invalid scanner token.

4	**GET**	/users/**useId**/tickets	List user's tickets.	**Headers,**
				
				Authorization: Bearer <JWT>
					200: {
					[
					{
					ticketId:" ",
					eventId:" ",
					seat:" ",
					status:" ",
					}
					]
					}
						401: unauthenticated.

1.3 Feature C – Organizer Dashboard (Management & Analytics)

#	Method	Route	Purpose	Request	Success Status & JSON	Error cases
1	**GET**	/organizer/dashboard	Retrieve organiser-level KPIs (totals, revenue).	**Header, Authorization**: Bearer <JWT>	**200:** { eventsTotal:" ", ticketsSold:" ", grossRevenue:" ", upcomingCount:" ", pastCount:" ", }	**401**: unauthenticated. **403:** role not organizer.
2	**GET**	/organizer/events	Paginated list of organiser's events.	**Header** auth; Query ?page, ?limit, optional `?status=" "	**200:** { page:" ", totalPages:" ", data: [{ id:" ", title:" ", status:" ", ticketsSold:" ", revenue:" ", }] }	**401:** unauthenticated. **403:** role not organizer. **400**: invalid query.
3	**GET**	/organizer/events/**eventId**/overview	Detailed metrics for one event.	**Header** auth; Path **eventId**	**200:** { eventId:" ", ticketsSoldByDay: [...], seatAvailability: {...}, revenue:" ", }	**404:** event not found.
4	**PUT**	/organizer/events/**eventId**/publish	Publish/unpublish event.	**Header** auth; Body: { status:" ", }	**200:** { eventId:" ", status:" ", }	**404**: not found. **422**: cannot unpublish after ticket sales started.
5	**GET**	/organizer/events/**eventId**/bookings	List bookings for event with filters.	**Header** auth, Path **eventId**; Query?status=" "	**200:** [{ bookingId:" ", user: {name}, seats:" ", amount:" ", status:" ", createdAt:" ", }]	**404**: not found. **400**: invalid date filter.

2. Security & Protection Endpoints

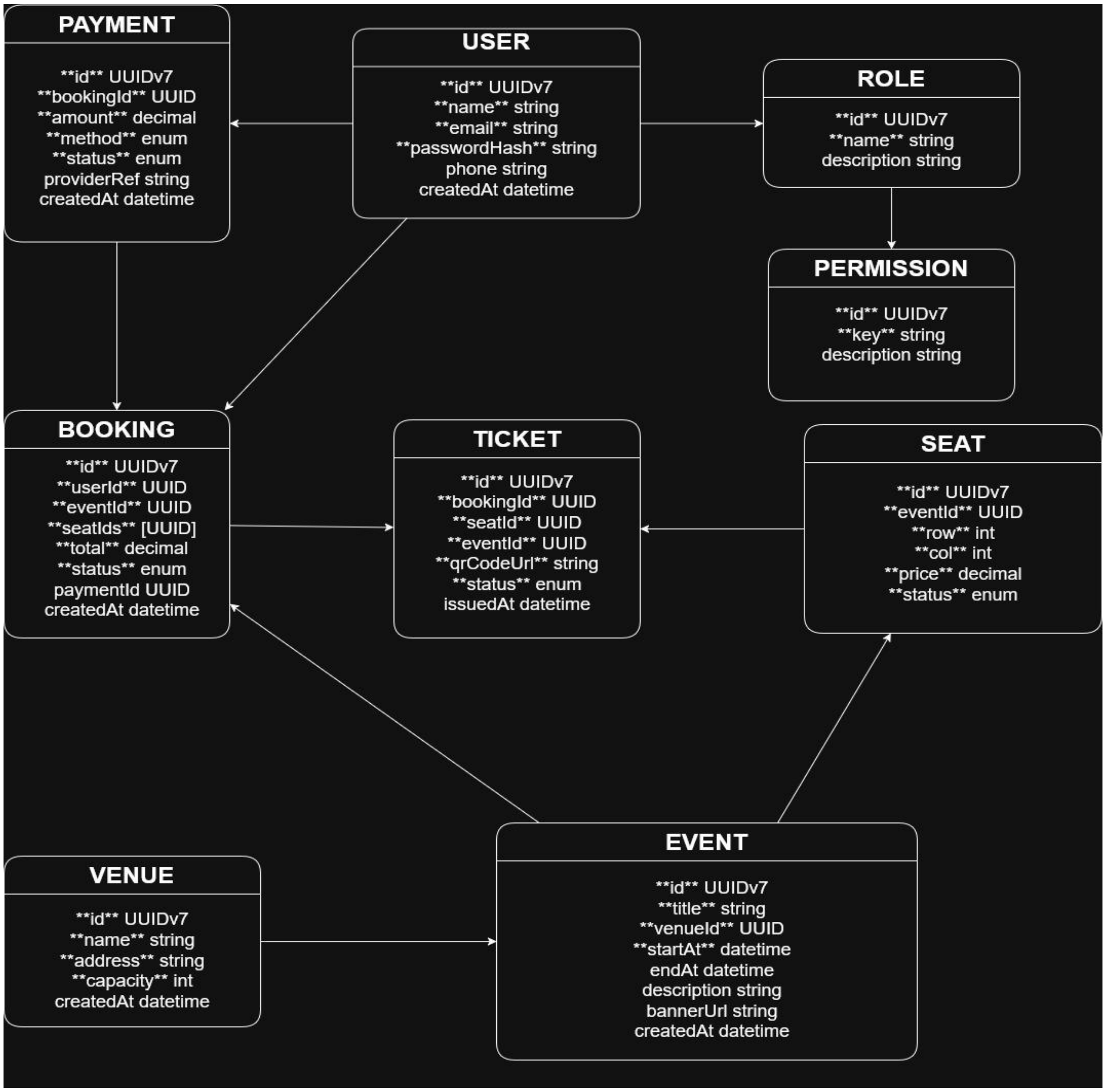
#	Method	Route	Purpose	Request (condensed)	Success Status & JSON	Error cases
1	**POST**	/auth/register	Register a new user account.	**Body**: { name:" ", email:" ", password:" ", }	**201**: { userId:" ", email:" ", }	**400:** validation error. **409:** email already exists.
2	**POST**	/auth/login	Obtain access + refresh tokens.	**Body**: { email:" ", password:" ", }	**200**: { accessToken:" ", refreshToken:" ", expiresIn:" ", }	**401:** invalid credentials. **429:** rate-limit exceeded.
3	**POST**	/auth/refresh	Rotate tokens (refresh token flow).	**Body**: { refreshToken:" ", }	**200**: { accessToken:" ", refreshToken:" ", }	**40: 1** token expired/blacklisted.
4	**POST**	/auth/logout	Invalidate refresh token server-side.	**Body**: { refreshToken:" ", }	**204**: no-content	**400:** missing/invalid token.
5	**POST**	/auth/password-reset/request	Send password-reset e-mail.	**Body**: { Email:" ", }	**200**: { message:" " }	**404:** email not found.
6	**POST**	/auth/password-reset/confirm	Reset password using token.	**Body**: { token:" ", newPassword:" ", }	**200**: { message:" " }	**400:** token invalid or expired.

2.1 Mitigation summary

- **Broken Access Control** – Role-based middleware (RBAC) checks req.user.role against route ACL, plus ownership checks for personal resources.
- **Injection** – MongoDB queries built with Mongoose parameters; inputs sanitised via express-mongo-sanitize.

- **Brute-Force** – /auth/login is rate-limited to 5 attempts per minute per IP; IP and user ID are tracked in Redis.
- **CSRF** – Stateless JWT transported exclusively in the **Authorization** header.
- **Transport Security** – Enforce HTTPS via HSTS; cookies (when used) flagged Secure, HttpOnly, SameSite=Strict.
- **Token Replay** – Refresh-token identifiers are stored with rotation detection, and blacklist is cached in Redis.

3. Data-Integration Design Data-Integration Design



3.1 CRUD Matrix

Entity	Create Endpoints	Read Endpoints	Update Endpoints	Delete Endpoints
Event	**POST** /events	**GET** /events/:id, **GET** /events	**PUT** /events/:id, **PUT** /organizer/events/:eventId/publish	**DELETE** /events/:id
Seat	—	**GET** /events/:eventId/seatmap	—	—
Booking	**POST** /events/eventId/bookings	**GET** /bookings/:bookingId, **GET** /organizer/events/:eventId/bookings	**PUT** /bookings/:bookingId	**DELETE** /bookings/:bookingId
Ticket	—	**GET** /tickets/:ticketId, **GET** /tickets/:ticketId/qrcode, **GET** /users/me/tickets	**POST** /tickets/:ticketId/validate	—
Payment	**POST** /payments/intent	**GET** /payments/:paymentId	**POST** /payments/webhook	—

4. Deployment Feature A — Seat-Map & Booking

4.1 Prerequisites

Item	**Version / Link**
Node ≥ 18 LTS	https://nodejs.org
Git clone	git clone https://github.com/shraymoza/webAssignment2/tree/main/eventsparkapi
Atlas cluster credentials	already embedded in .env
Postman or curl	any version
Render URI	https://webassignment2-jo5k.onrender.com

4.2 Run this for token generation before running any API (temporary login)

Run this command in the project directory to generate a token that will be used in auth headers of the APIs. This will generate a demo.token.txt in the project directory that holds the token—paste it into Postman's "Authorization → Bearer Token" field for all protected calls.

```
node -r dotenv/config generateToken.js
```

4.3 API Reference (live deployment)

For testing you may use eventId as **evt_demo123** in the eventId parameter of API for testing purpose as it's the only event currently in mongo dB for testing purpose.

#	**Method**	**Route**	**Auth?**	**Body (JSON)**	**Success / Sample**	**Errors**
A-1	GET	/events/:eventId/seatmap	None	–	**200**{ eventId, seats:[{id,row,col,price,status}] }	404
A-2	POST	/events/:eventId/holds	**Bearer**	{ "seatIds": ["sea_…"] }	**200** { held:[…] }	400, 409
A-3	POST	/events/**:eventId**/bookings	**Bearer**	{ "seatIds": ["sea_…"] }	**201** { bookingId, status, expiresAt }	400, 409

A-4	GET	/bookings/**:bookingId**	**Bearer**	–	**200** booking doc	403, 404
A-5	DELETE	/bookings/**:bookingId**	**Bearer**	–	**204**	403, 404

4.4 Reset Script (if you need a clean slate)

resetSeats.js sets every seat in evt_demo123 back to AVAILABLE
so you can re-demo the flow endlessly.

```
node -r dotenv/config scripts/resetSeats.js
```