**🏟️ Book a Court - Backend API**

This is the backend server for the **Book a Court** application. It allows users to register, verify their email, login, manage their profile, and book courts securely.

**🚀 Features**

* User registration with email verification.
* Secure user login and logout.
* Profile view, update, and profile picture upload.
* Book court for a specific date and time.
* View, edit, cancel, and delete court bookings.
* Pagination and filtering for booking history.
* Secure authentication using sessions.
* File uploads (images only) handled safely.
* Environment variables for sensitive configurations.

**🛠️ Technologies Used**

* Node.js
* Express.js
* MySQL 8.0.41
* Multer (file uploads)
* bcrypt (password hashing)
* nodemailer (email sending)
* dotenv (environment management)

**📂 Project Structure**

backend/

|

├── config/

│ ├── db.js # Database connection

│ ├── multerConfig.js # File upload configuration

│

├── middleware/

│ └── authMiddleware.js # Authentication middleware

│

├── routes/

│ ├── authRoutes.js # User auth and profile routes

│ └── bookingRoutes.js # Court booking routes

│

├── uploads/ # Uploaded profile pictures

│

├── server.js # Main backend server entry point

|

└── .env # Environment variables (Not uploaded publicly)

**⚙️ Setup Instructions**

1. **Clone the project**

git clone https://your-repo-url.git

cd backend

1. **Install dependencies**

npm install

1. **Configure environment**

Create a .env file inside the backend/ folder. Example:

PORT=5000

SESSION\_SECRET=your\_secret\_session\_key

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=159635741?Ny

DB\_NAME=book\_a\_court\_db

EMAIL\_HOST=sandbox.smtp.mailtrap.io

EMAIL\_PORT=587

EMAIL\_USER=68d463128ceb3f

EMAIL\_PASS=37888de1cc25d4

1. **Import the Database**

* Open your MySQL Command Line Client or phpMyAdmin.
* Create a database named book\_a\_court\_db.
* Import the provided book\_a\_court\_db.sql file to setup tables and initial data.

CREATE DATABASE book\_a\_court\_db;

USE book\_a\_court\_db;

-- Then import the SQL script content.

1. **Start the server**

npm start

Server will run at:  
http://localhost:5000

**📩 API Endpoints Overview**

| **Route** | **Method** | **Protected** | **Description** |
| --- | --- | --- | --- |
| /api/auth/register | POST | ❌ | Register new user |
| /api/auth/login | POST | ❌ | User login |
| /api/auth/logout | POST | ✅ | User logout |
| /api/auth/profile | GET | ✅ | Get user profile |
| /api/auth/profile | PUT | ✅ | Update user profile |
| /api/auth/profile/upload | POST | ✅ | Upload profile picture |
| /api/auth/verify?token=xxx | GET | ❌ | Email verification |
| /api/book | POST | ✅ | Create a booking |
| /api/book | GET | ✅ | View bookings (filter/pagination supported) |
| /api/book/:id | GET | ✅ | Get booking by ID |
| /api/book/:id | PUT | ✅ | Edit booking |
| /api/book/:id/cancel | PUT | ✅ | Cancel booking |
| /api/book/:id | DELETE | ✅ | Delete booking |

**🔐 Security Notes**

* Sessions are protected via SESSION\_SECRET.
* Uploaded files are restricted to images only (.jpg, .jpeg, .png).
* Booking times in the past are rejected.
* Sensitive data is loaded securely via .env file.

**🧹 Things to Improve Later**

* Add rate limiting (express-rate-limit) to prevent spamming.
* Move to connection pooling for better database performance.
* Setup HTTPS when moving to production.

**🏆 That's it!**

Backend is ready, secure, and production-quality.

**📦 Additional Setup: Database Tables (MySQL)**

Example SQL to create tables:

-- Users Table

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

birthdate DATE DEFAULT NULL,

gender ENUM('Male','Female','Other') DEFAULT NULL,

phone\_number VARCHAR(20) DEFAULT NULL,

country\_code VARCHAR(5) DEFAULT NULL,

location VARCHAR(100) DEFAULT NULL,

profile\_picture VARCHAR(255),

strength TINYINT DEFAULT NULL,

wins INT DEFAULT 0,

losses INT DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

is\_verified TINYINT(1) DEFAULT 0,

verification\_token VARCHAR(255)

);

-- Courts Table

CREATE TABLE courts (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) DEFAULT NULL,

location VARCHAR(100) DEFAULT NULL,

type ENUM('indoor','outdoor') DEFAULT NULL,

is\_available TINYINT(1) DEFAULT 1

);

-- Bookings Table

CREATE TABLE bookings (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

court\_id INT NOT NULL,

booking\_date DATE NOT NULL,

booking\_time TIME NOT NULL,

status ENUM('booked','canceled') DEFAULT 'booked',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

court\_name VARCHAR(100),

location VARCHAR(100),

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (court\_id) REFERENCES courts(id) ON DELETE CASCADE

);

-- Friends Table

CREATE TABLE friends (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

friend\_id INT NOT NULL,

status ENUM('pending','accepted','rejected') DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (friend\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- Match Requests Table

CREATE TABLE match\_requests (

id INT AUTO\_INCREMENT PRIMARY KEY,

requester\_id INT NOT NULL,

opponent\_id INT NOT NULL,

proposed\_date DATE DEFAULT NULL,

proposed\_time TIME DEFAULT NULL,

status ENUM('pending','confirmed','rejected') DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (requester\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (opponent\_id) REFERENCES users(id) ON DELETE CASCADE

);

-- Matches Table

CREATE TABLE matches (

id INT AUTO\_INCREMENT PRIMARY KEY,

player1\_id INT NOT NULL,

player2\_id INT NOT NULL,

court\_id INT DEFAULT NULL,

match\_date DATE DEFAULT NULL,

match\_time TIME DEFAULT NULL,

score VARCHAR(50) DEFAULT NULL,

winner\_id INT DEFAULT NULL,

location VARCHAR(100) DEFAULT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (player1\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (player2\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (court\_id) REFERENCES courts(id) ON DELETE CASCADE,

FOREIGN KEY (winner\_id) REFERENCES users(id) ON DELETE CASCADE

);

🔗 **Make sure to create the uploads/ folder manually inside backend/!**

**🚀 Ready to Deploy!**