# Timetable Management System API Documentation

This API documentation provides details on how to interact with the Timetable Management System API.

## Base URL

The base URL for all API endpoints is `localhost:5000`.

## Authentication

The API requires authentication using an API key. Include the API key in the headers of your requests as follows:(JWT Toekn)

**Authorization: Bearer <api_key>**

# Endpoints

## User Roles and Authentication

### 1. Register User

- **URL:** `localhost:5000/api/user/register`
- **Method:** `POST`
- **Description:** Registers a new user.
- **Request Body:**
  - `firstName` (string, required): The first name of the user.
  - `lastName` (string, required): The last name of the user.
  - `email` (string, required): The email address of the user.
  - `mobile` (string, required): The mobile number of the user.
  - `password` (string, required): The password for the user account.
  - `role` (string, required): The role of the user ( `Admin` , `Faculty` , `Student` ).
- **Response:**
  - `200 OK` - Returns the newly created user object.
  - `400 Bad Request` - If the request body is invalid.

    - `500 Internal Server Error` - If an unexpected error occurs.

Example Request:

```
POST localhost:5000/api/user/register
Content-Type: application/json

{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@example.com",
  "mobile": "1234567890",
  "password": "password123",
  "role": "Admin"
}
```

## 2. Login User

- **URL:** `/api/user/login`
- **Method:** `POST`
- **Description:** Logs in a user.
- **Request Body:**
    - `email` (string, required): The email address of the user.
    - `password` (string, required): The password for the user account.
- **Response:**
    - `200 OK` - Returns user details along with an authentication token.
    - `400 Bad Request` - If the email or password is invalid.
    - `500 Internal Server Error` - If an unexpected error occurs.

Example Request:

```
POST localhost:5000/api/user/login
Content-Type: application/json
{

  "email": "john@example.com",
  "password": "password123"

}
```

## 3. Get All Users

- **URL:** `localhost:5000/api/user/allusers`
- **Method:** `GET`
- **Description:** Retrieves a list of all users.
- **Authentication:** Requires authentication.
- **Authorization:** Requires admin privileges.
- **Response:**
  - `200 OK` - Returns a JSON array of user objects.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `403 Forbidden` - If the user does not have admin privileges.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 4. Get a User by ID

- **URL:** `localhost:5000/api/user/user/:id`
- **Method:** `GET`
- **Description:** Retrieves a user by their ID.
- **Authentication:** Requires authentication.
- **Response:**
  - `200 OK` - Returns the user object.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `404 Not Found` - If no user is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 5. Delete a User by ID

- **URL:** `localhost:5000/api/user/user/:id`
- **Method:** `DELETE`
- **Description:** Deletes a user by their ID.
- **Authentication:** Requires authentication.
- **Response:**
  - `200 OK` - Returns the deleted user object.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `404 Not Found` - If no user is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 6. Update a User by ID

- **URL:** `localhost:5000/api/user/user/:id`
- **Method:** `PUT`
- **Description:** Updates a user by their ID.
- **Authentication:** Requires authentication.
- **Request Body:** The updated user data.
- **Response:**
  - `200 OK` - Returns the updated user object.
  - `400 Bad Request` - If the request body is invalid.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `404 Not Found` - If no user is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 7. Refresh Token

- **URL:** `localhost:5000/api/user/refreshtoken`
- **Method:** `GET`
- **Description:** Refreshes the authentication token.
- **Authentication:** Requires authentication.
- **Response:**
  - `200 OK` - Returns a new access token.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 8. Logout User

- **URL:** `localhost:5000/api/user/logout`
- **Method:** `GET`
- **Description:** Logs out the user and clears the refresh token.
- **Authentication:** Requires authentication.
- **Response:**
  - `200 OK` - Returns a message indicating successful logout.
  - `401 Unauthorized` - If the request lacks proper authentication.
  - `500 Internal Server Error` - If an unexpected error occurs.

# Course Management

## 1. Create Course

- **URL:** `localhost:5000/api/course/create`
- **Method:** `POST`
- **Description:** Creates a new course.
- **Request Body:**
  - `name` (string, required): The name of the course.
  - `code` (string, required): The code of the course.
  - `description` (string, required): Description of the course.
  - `credits` (number, required): Number of credits for the course.
- **Response:**
  - `200 OK` - Returns the newly created course object.
  - `400 Bad Request` - If the request body is invalid.
  - `500 Internal Server Error` - If an unexpected error occurs.

Example Request:

```
POST localhost:5000/api/course/create
Content-Type: application/json
Authorization: Bearer <api_key>

{
  "name": "Introduction to Computer Science",
  "code": "CS101",
  "description": "An introductory course to computer science concepts.",
  "credits": 3
}
```

## 2. Get Course by ID

- **URL:** `localhost:5000/api/course/:id`
- **Method:** `GET`
- **Description:** Retrieves a course by its ID.
- **Response:**
  - `200 OK` - Returns the course object.
  - `404 Not Found` - If no course is found with the specified ID.

    - `500 Internal Server Error` - If an unexpected error occurs.

## 3. Get All Courses

- **URL:** `localhost:5000/api/course`
- **Method:** `GET`
- **Description:** Retrieves a list of all courses.
- **Response:**
    - `200 OK` - Returns a JSON array of course objects.
    - `500 Internal Server Error` - If an unexpected error occurs.

## 4. Update Course by ID

- **URL:** `localhost:5000/api/course/:id`
- **Method:** `PUT`
- **Description:** Updates a course by its ID.
- **Request Body:** The updated course data.
- **Response:**
    - `200 OK` - Returns the updated course object.
    - `400 Bad Request` - If the request body is invalid.
    - `404 Not Found` - If no course is found with the specified ID.
    - `500 Internal Server Error` - If an unexpected error occurs.

## 5. Delete Course by ID

- **URL:** `localhost:5000/api/course/:id`
- **Method:** `DELETE`
- **Description:** Deletes a course by its ID.
- **Response:**
    - `200 OK` - Returns the deleted course object.
    - `404 Not Found` - If no course is found with the specified ID.
    - `500 Internal Server Error` - If an unexpected error occurs.

## 6. Assign Faculty to Course

- **URL:** `localhost:5000/api/course/course/:courseId`
- **Method:** `PUT`

- **Description:** Assigns a faculty to a course.
- **Request Body:**
  - `facultyId` (string, required): The ID of the faculty to be assigned.
- **Response:**
  - `200 OK` - Returns the updated course object with assigned faculty.
  - `400 Bad Request` - If the request body is invalid.
  - `404 Not Found` - If no course is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

Example Request:

```
PUT localhost:5000/api/course/course/:courseId
Content-Type: application/json
Authorization: Bearer <api_key>

{
  "Faculty": facultyid
}
```

# Room and Resource Booking

## 1. Create Booking

- **URL:** `localhost:5000/api/booking/create`
- **Method:** `POST`
- **Description:** Creates a new booking.
- **Request Body:**
  - `roomId` (string, required): The ID of the room.
  - `resourceId` (string, required): The ID of the resource.
  - `dayOfWeek` (string, required): The day of the week (Monday, Tuesday, Wednesday, Thursday, Friday).
  - `startTime` (string, required): The start time of the booking.
  - `endTime` (string, required): The end time of the booking.
- **Response:**
  - `200 OK` - Returns the newly created booking object.
  - `400 Bad Request` - If the request body is invalid.
  - `500 Internal Server Error` - If an unexpected error occurs. Example Request:

```
POST localhost:5000/api/booking/create HTTP/1.1
Content-Type: application/json
Authorization: Bearer <api_key>

{
  "roomId": "607a22cf1369200049e785e5",
  "resourceId": "607a22cf1369200049e785e6",
  "dayOfWeek": "Monday",
  "startTime": "09:00",
  "endTime": "11:00"
}
```

## 2. Get Booking by ID

- **URL:** `localhost:5000/api/booking/:id`
- **Method:** `GET`
- **Description:** Retrieves a booking by its ID.
- **Response:**
  - `200 OK` - Returns the booking object.
  - `404 Not Found` - If no booking is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 3. Get All Bookings

- **URL:** `localhost:5000/api/booking`
- **Method:** `GET`
- **Description:** Retrieves a list of all bookings.
- **Response:**
  - `200 OK` - Returns a JSON array of booking objects.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 4. Update Booking by ID

- **URL:** `localhost:5000/api/booking/:id`
- **Method:** `PUT`
- **Description:** Updates a booking by its ID.
- **Request Body:** The updated booking data.
- **Response:**

- `200 OK` - Returns the updated booking object.
- `400 Bad Request` - If the request body is invalid.
- `404 Not Found` - If no booking is found with the specified ID.
- `500 Internal Server Error` - If an unexpected error occurs.

### 5. Delete Booking by ID

- **URL:** `localhost:5000/api/booking/:id`
- **Method:** `DELETE`
- **Description:** Deletes a booking by its ID.
- **Response:**
  - `200 OK` - Returns the deleted booking object.
  - `404 Not Found` - If no booking is found with the specified ID.
  - `500 Internal Server Error` - If an unexpected error occurs.

## Timetable Management

### 1.Create Session in Course

- **URL:** `localhost:5000/api/timetablesession/create/:courseId`
- **Method:** `POST`
- **Description:** Adds a new session to the course timetable.
- **Request Body:**
  - `location` (string, required): The location ID of the session.
  - `sessionStatus` (string, required): The status of the session (Scheduled, Cancelled, Postponed).
  - `description` (string, required): Description of the session.
  - `dayOfWeek` (string, required): The day of the week for the session (Monday, Tuesday, Wednesday, Thursday, Friday).
  - `startTime` (string, required): The start time of the session.
  - `endTime` (string, required): The end time of the session.
  - `faculty` (string, required): The ID of the faculty member.
- **Response:**
  - `200 OK` - Returns the updated timetable for the course.
  - `400 Bad Request` - If the request body is invalid.

- `404 Not Found` - If the course with the specified ID is not found.
- `500 Internal Server Error` - If an unexpected error occurs.

Example Request:

```
POST localhost:5000/api/timetablesession/create/:courseId HTTP/1.1
Host: your-api-host.com
Content-Type: application/json
Authorization: Bearer your_access_token

{
  "location": "room_id",
  "sessionStatus": "Scheduled",
  "description": "Lecture on Introduction to Computer Science",
  "dayOfWeek": "Monday",
  "startTime": "09:00 AM",
  "endTime": "10:30 AM",
  "faculty": "faculty_id"
}
```

## 2.Update Session in Course

- **URL:** `localhost:5000/api/timetablesession/update/:courseId/:sessionId`
- **Method:** `PUT`
- **Description:** Updates an existing session in the course timetable.
- **Request Body:**
  - Same as the request body for creating a session.
- **Response:**
  - `200 OK` - Returns the updated timetable for the course.
  - `400 Bad Request` - If the request body is invalid.
  - `404 Not Found` - If the course or session with the specified ID is not found.
  - `500 Internal Server Error` - If an unexpected error occurs.

## 3.Delete Session in Course

- **URL:** `localhost:5000/api/timetablesession/delete/:courseId/:sessionId`
- **Method:** `DELETE`
- **Description:** Deletes a session from the course timetable.
- **Response:**
  - `200 OK` - Returns the updated timetable for the course.

- `404 Not Found` - If the course or session with the specified ID is not found.
- `500 Internal Server Error` - If an unexpected error occurs.

### 4.Get All Sessions in Course

- **URL:** `localhost:5000/api/timetablesession/all/:courseId`
- **Method:** `GET`
- **Description:** Retrieves all sessions in the course timetable.
- **Response:**
  - `200 OK` - Returns a JSON array of all sessions in the course timetable.
  - `404 Not Found` - If the course with the specified ID is not found.
  - `500 Internal Server Error` - If an unexpected error occurs.

## Student Enrollment

### 1.Create Course Enrollment

Enroll a student in a course.

- **URL:** `localhost:5000/api/courseenrollment/:courseId/:studentId`
- **Method:** `POST`
- **Auth Required:** Yes
- **Permissions Required:** Student
- **Request Body:** None
- **Parameters:**
  - `courseId` (string): ID of the course to enroll in.
  - `studentId` (string): ID of the student to enroll.
- **Success Response:**
  - **Code:** `200 OK`
  - **Content:** JSON object representing the newly created course enrollment.
- **Error Responses:**
  - **Code:** `404 Not Found`
    - **Content:** `{ "error": "Course not found" }`
  - **Code:** `400 Bad Request`
    - **Content:** `{ "error": "Student is already enrolled in the course" }`

## 2.Delete Course Enrollment

Unenroll a student from a course.

- **URL:** `localhost:5000/api/courseenrollment/:courseId/:studentId`
- **Method:** `DELETE`
- **Auth Required:** Yes
- **Permissions Required:** Student
- **Request Body:** None
- **Parameters:**
  - `courseId` (string): ID of the course to unenroll from.
  - `studentId` (string): ID of the student to unenroll.
- **Success Response:**
  - **Code:** `200 OK`
  - **Content:** `{ "message": "Student unenrolled successfully" }`
- **Error Responses:**
  - **Code:** `404 Not Found`
    - **Content:** `{ "error": "Course not found" }`
  - **Code:** `400 Bad Request`
    - **Content:** `{ "error": "Student is not enrolled in the course" }`

## 3.Get Enrolled Students

Get all students enrolled in a course.

- **URL:** `localhost:5000/api/courseenrollment/:courseId`
- **Method:** `GET`
- **Auth Required:** Yes
- **Permissions Required:** Admin or Faculty
- **Request Body:** None
- **Parameters:**
  - `courseId` (string): ID of the course to retrieve enrolled students for.
- **Success Response:**
  - **Code:** `200 OK`
  - **Content:** JSON array of objects representing enrolled students.
- **Error Response:**
  - **Code:** `404 Not Found`

- **Content:** `{ "error": "Course not found" }`