User stories

User Login

As a user, I want to authenticate myself by providing a username and password, so that I can receive an authentication token and access the application.

API Specification:

Endpoint: users/login
Method: POST
Request Body:
```
{
   "username": "admin",
   "password": "admin123"
}
```

Success Response:
Status Code: 200 OK
Content-Type: application/json
Response Body:
```
{
  "message": "Authentication successful.",
  "token":
"eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJjb3Vyc2VzX2FwcCIsInN1YiI6ImFkbWluIiwiZXhwIjoxNzQzOTk5Njk2LCJpYXQiOjE3NDM5NzA4OTYsInNjb3BlIjoiUk9MRV9BRE1JTiJ9.uPZRhbbuhjOUf5njgwLy2OKMn1pdqRbICF-fTact1GI",
  "username": "admin",
  "fullname": "admin admin",
  "role": "admin"
}
```

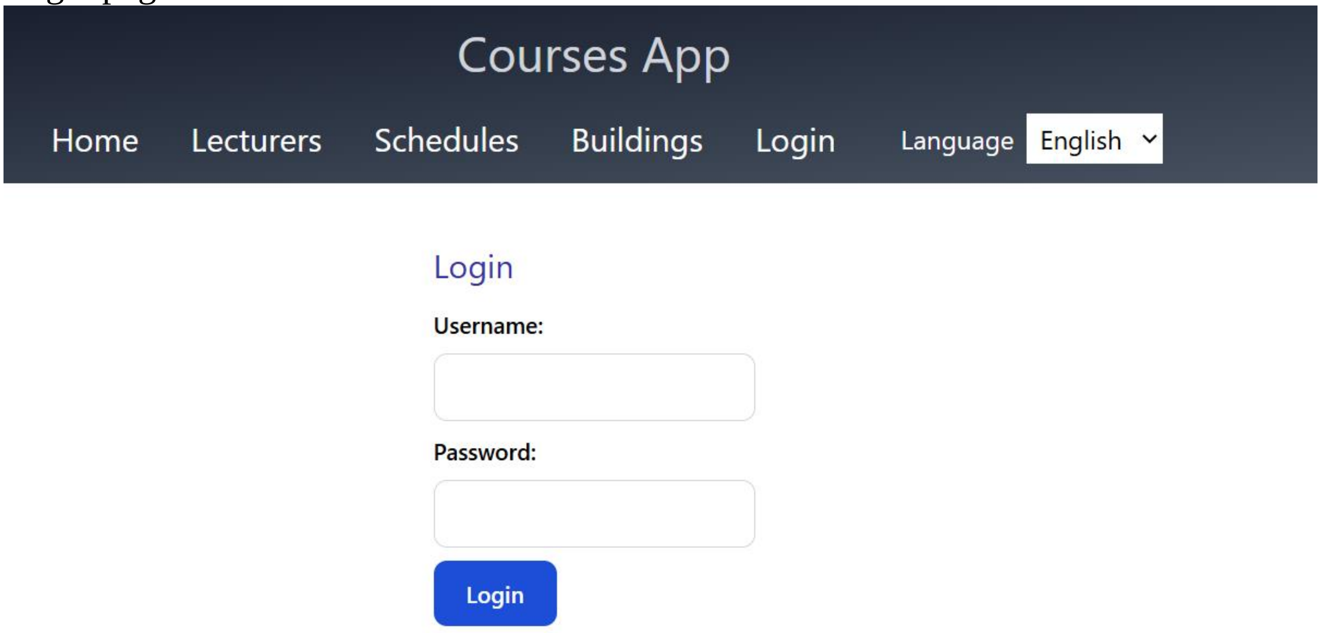
Error Response:
Status Code: 400 Unathorized
Content-Type: application/json
Response Body:
```
{
   "message": "Incorrect login or password"
```

}

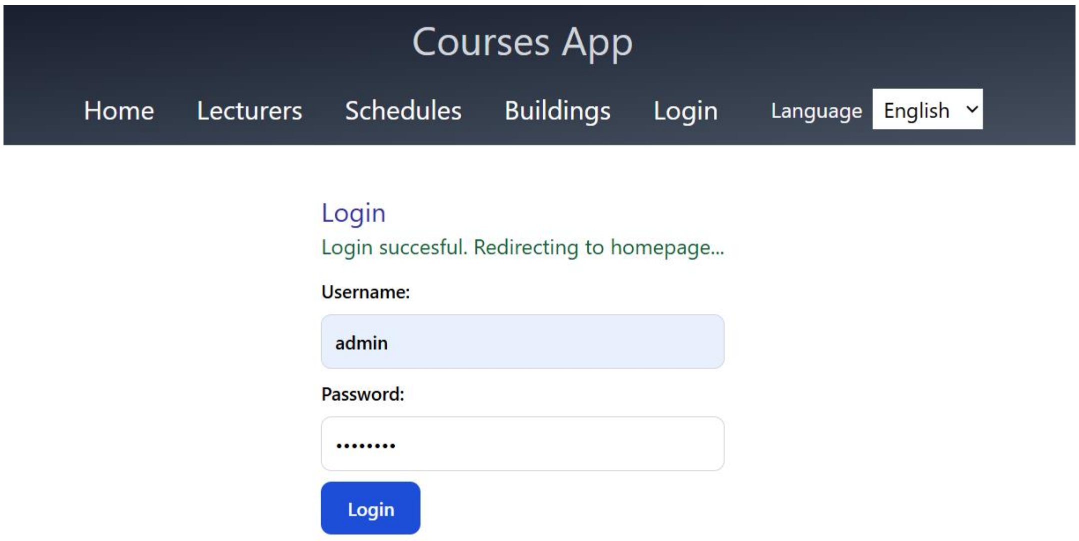
Acceptance Criteria:

1. When the user provides a valid username and password, the system should authenticate the user and return a `200 OK` status along with the authentication message, JWT token, username, full name, and role.
2. The response body must contain the fields `message`, `token`, `username`, `fullName`, and `role`.
3. If authentication is successful, the user receives an authentication token in the response.
4. If the username provided does not exist in the system, the system should return a `400 Unathorized` status with the message: "Incorrect login or password"

Mocks:
Login page



Right username and password:



Login

Login succesful. Redirecting to homepage...

Username: admin

Password: ••••••••

User Signup

User story:

As a new user, I want to register by providing a username, first name, last name, email, password, and role, so that I can create an account and access the application.

API Specification:

Endpoint: users/signup
Method: POST
Request Body:

```
{
   "username": "newUser",
   "firstName": "John",
   "lastName": "Doe",
   "email": "johndoe@example.com",
   "password": "password123",
   "role": "STUDENT"
}
```

Success Response:
Status Code: 201 Created
Content-Type: application/json
Response Body:

```
{
   "id": 1,
   "username": "newUser",
   "firstName": "John",
   "lastName": "Doe",
   "email": "johndoe@example.com",
   "role": "STUDENT"

}
```

Error Response (Username Already Exists):
Status Code: 400 Bad Request
Content-Type: application/json
Response Body:

```
{
    "message": "Username is already in use."
}
```

Acceptance Criteria:

1. When the user provides a unique username, first name, last name, email, password, and role, the system should create a new user account and return a `201 Created` status along with the user's information.
2. The response body must contain the fields `id`, `username`, `firstName`, `lastName`, `email`, `role`, `createdAt`, and `updatedAt`.
3. If the username already exists, the system should return a `400 Bad Request` status with the message: "Username is already in use."
4. The password should be hashed before being stored, and the email should be validated.

Get All Users

User Story:

As an admin, I want to retrieve a list of all users, so that I can manage the users in the application.

API Specification:

Endpoint: /users
Method: GET
Request Body: None

Success Response:
Status Code: 200 OK
Content-Type: application/json
Response Body:

```
[
   {
      "id": 1,
      "username": "admin",
      "firstName": "Admin",
      "lastName": "User",
      "email": "admin@example.com",
      "role": "ADMIN"
   },
   {
      "id": 2,
      "username": "user1",
      "firstName": "John",
      "lastName": "Doe",
      "email": "johndoe@example.com",
      "role": "STUDENT"

   }
]
```

Acceptance Criteria:

1. When the admin sends a GET request to `/users`, the system should return a list of all users.
2. The response body should contain an array of user objects with the fields `id`, `username`, `firstName`, `lastName`, `email`, `role`, `createdAt`, and `updatedAt`.
3. The system should return a `200 OK` status along with the list of users in the response body.
5. The response should not include sensitive information such as the password field.

Get All Students

User Story:

As an admin, I want to retrieve a list of all students, so that I can manage student data in the application.

API Specification:

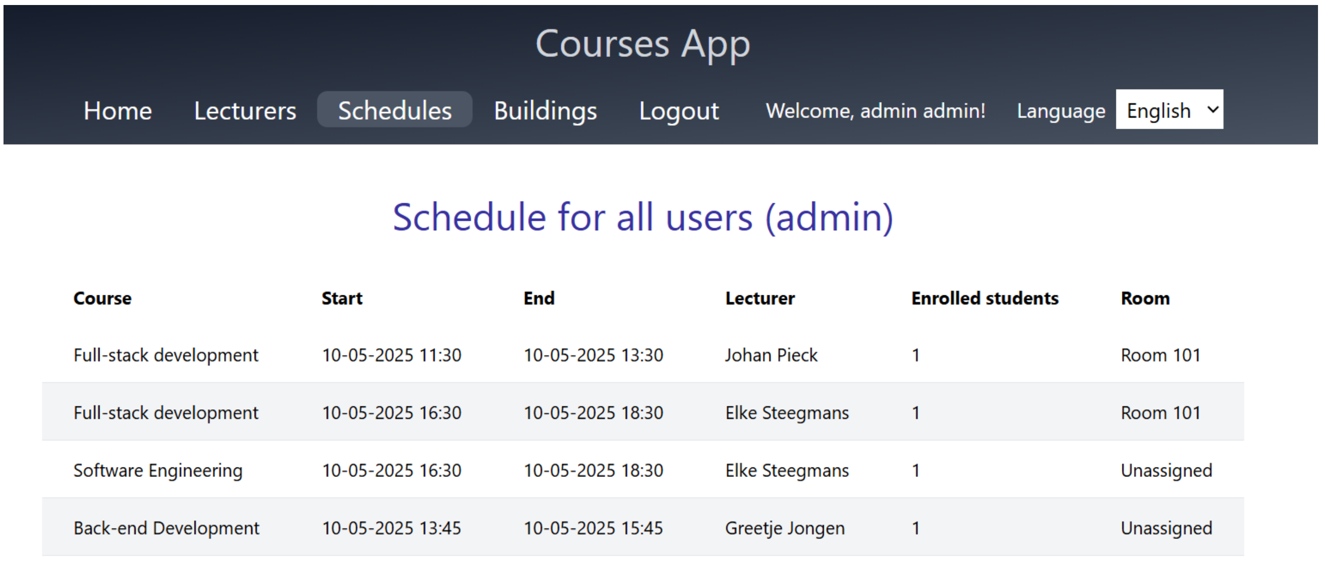
Endpoint: /students
Method: GET
Request Body: None

Success Response:
Status Code: 200 OK
Content-Type: application/json
Response Body:

```
[
  {
    "id": 1,
    "username": "admin",
    "firstName": "admin",
    "lastName": "admin",
    "email": "administration@ucll.be",
    "role": "admin"
  },
  {
    "id": 2,
    "username": "johanp",
    "firstName": "Johan",
    "lastName": "Pieck",
    "email": "johan.pieck@ucll.be",
    "role": "lecturer"
  }
]
```

Error:
400 Unathorized

Mocks:
Go to schedules page:

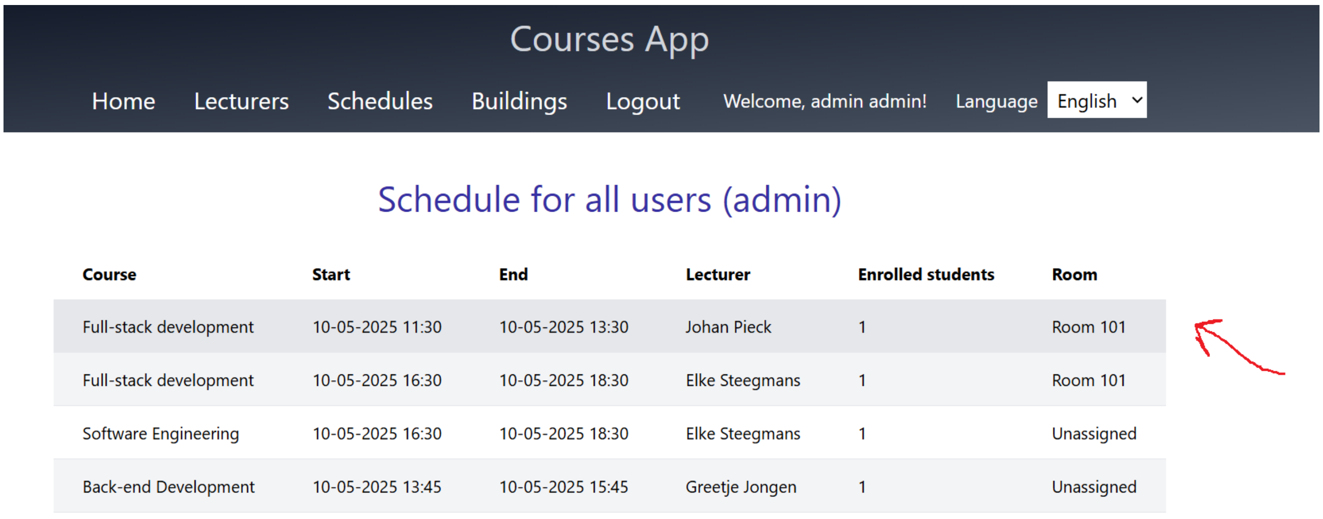

Courses App

Home | Lecturers | Schedules | Buildings | Logout | Welcome, admin admin! | Language | English

Schedule for all users (admin)

Course	Start	End	Lecturer	Enrolled students	Room
Full-stack development	10-05-2025 11:30	10-05-2025 13:30	Johan Pieck	1	Room 101
Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned

Click on schedule:

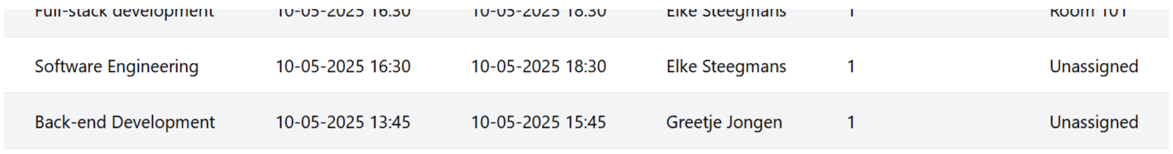

Courses App

Home | Lecturers | Schedules | Buildings | Logout | Welcome, admin admin! | Language | English

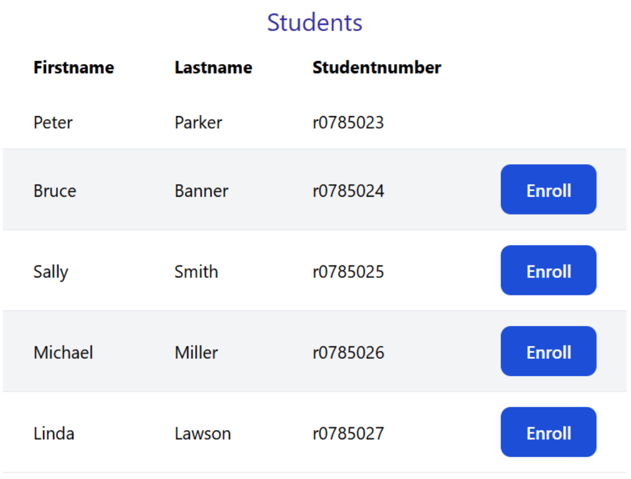
Schedule for all users (admin)

Course	Start	End	Lecturer	Enrolled students	Room
Full-stack development	10-05-2025 11:30	10-05-2025 13:30	Johan Pieck	1	Room 101
Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned

Then you can see students (and enroll them):



Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned



Students

Firstname	Lastname	Studentnumber	
Peter	Parker	r0785023	
Bruce	Banner	r0785024	Enroll
Sally	Smith	r0785025	Enroll
Michael	Miller	r0785026	Enroll
Linda	Lawson	r0785027	Enroll

Acceptance Criteria:

1. When the admin sends a GET request to `/students`, the system should return a list of all students.
2. The response body should contain an array of student objects with the fields `id`, `studentNumber`, `user`, and `schedules`.
3. The `user` field must contain the user details related to the student, including `id`, `username`, `firstName`, `lastName`, `email`, and `role`.
4. The `schedules` field must be a list of courses assigned to the student, with course details like `id`, `course`, and `semester`.
5. The system should return a `200 OK` status along with the list of students in the response body.
6. The response should not include sensitive information, such as the student's password or other private data.

Get Course Information by ID

User story:

As a student, I want to be able to get information about a course by its ID, so that I can view the details of the course such as name, description, phase, credits, and associated lecturers.

API Specification:

Endpoint: /courses/{id}
Method: GET
URL Path Parameter: id (required): The unique ID of the course to retrieve.
Request Body: None.
Response Body:

```
{
  "id": 1,
  "name": "Full-stack development",
  "description": "Learn how to build a full stack web application.",
  "phase": 2,
  "credits": 6
}
```

Success Response:
Status Code: 200 OK
Content-Type: application/json
Body: Course details in JSON format (as shown above).

Error Responses:
404 Not Found: If the course with the given id does not exist.
Message: "Course with id {id} not found"

500 Internal Server Error: For any unexpected errors during the request processing.
Message: "Internal Server Error"

Acceptance Criteria:

- When a valid course ID is provided in the request, the system should return the correct course details in a structured JSON format. The response should include the course ID, name, description, phase, credits, and associated lecturers. The lecturers array should contain the id and name of each lecturer.

- If the course ID does not exist in the database, the system should throw a NotFoundException and return a 404 Not Found response with the appropriate message: "Course with id {id} not found".

- If there is any unexpected error during the processing of the request, a 500 Internal Server Error should be returned with a generic error message: "Internal Server Error".

- The response body must be in a valid JSON format. The JSON should follow the structure mentioned above with the correct data types (string, integer, list).

Get all lecturers

User story:
As an admin,
I want to retrieve a list of all lecturers,
So that I can view the details of all lecturers, including their expertise and associated user information.

Acceptance Criteria:
- The GET request to '/lecturers' should return a list of all lecturers.
- The list should include the lecturer's expertise and associated user information such as username, first name, last name, and email.
- The response should not include fields that are marked with @JsonIgnore, such as the createdAt and updatedAt timestamps.
- If no lecturers exist, the response should return an empty list.

API Specification:

Endpoint: GET /lecturers

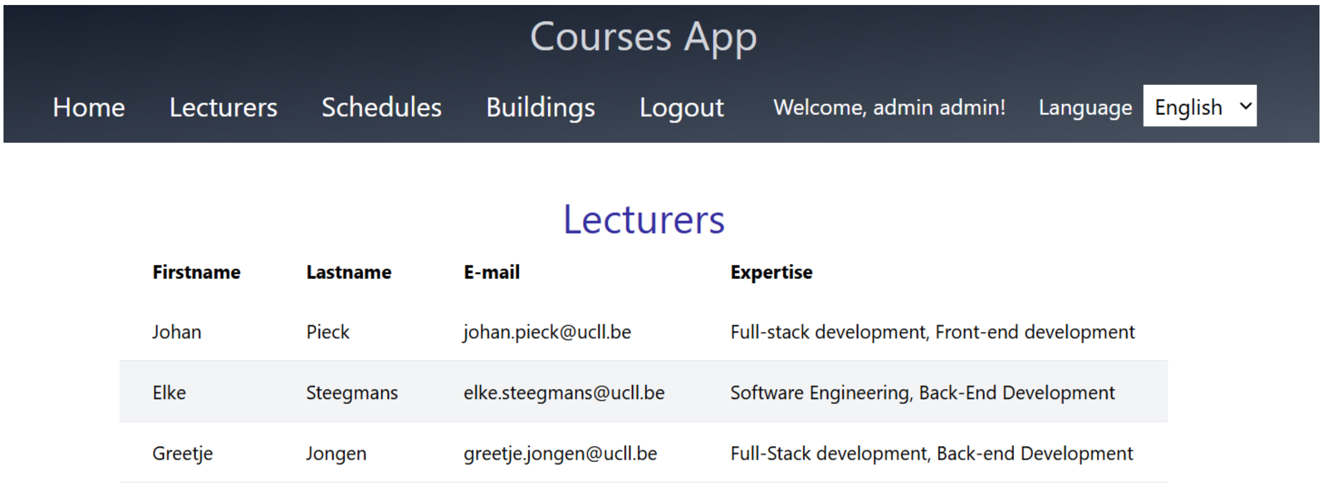
Description: Retrieves a list of all lecturers, including their expertise and associated user information.

Request:
- Method: GET
- URL: /lecturers
- Headers:
  - Content-Type: application/json

Success Response:
- Status: 200 OK
- Body:

```
  [
{
  "id": 1,
  "expertise": "Full-stack development, Front-end development",
  "user": {
    "id": 2,
    "username": "johanp",
    "firstName": "Johan",
    "lastName": "Pieck",
    "email": "johan.pieck@ucll.be",
    "role": "lecturer"
  },
  "courses": [
    {
      "id": 3,
      "name": "Front-End Development",
      "description": "Learn how to build a front-end web application.",
      "phase": 1,
      "credits": 6
    },
    {
      "id": 1,
      "name": "Full-stack development",
      "description": "Learn how to build a full stack web application.",
      "phase": 2,
      "credits": 6
    }
  ]
},
{
  "id": 2,
  "expertise": "Software Engineering, Back-End Development",
  "user": {
    "id": 3,
    "username": "elkes",
    "firstName": "Elke",
    "lastName": "Steegmans",
    "email": "elke.steegmans@ucll.be",
    "role": "lecturer"
  },
  "courses": [
    {
      "id": 1,
      "name": "Full-stack development",
      "description": "Learn how to build a full stack web application.",
      "phase": 2,
      "credits": 6
    },
    {
      "id": 2,
      "name": "Software Engineering",
      "description": "Learn how to build and deploy a software application.",
      "phase": 2,
      "credits": 6
    }
  ]
},
{
  "id": 3,
  "expertise": "Full-Stack development, Back-end Development",
  "user": {
    "id": 4,
    "username": "greetjej",
    "firstName": "Greetje",
    "lastName": "Jongen",
    "email": "greetje.jongen@ucll.be",
    "role": "lecturer"
  },
  "courses": [
    {
      "id": 1,
      "name": "Full-stack development",
      "description": "Learn how to build a full stack web application.",
      "phase": 2,
      "credits": 6
    },
    {
      "id": 4,
      "name": "Back-end Development",
      "description": "Learn how to build a REST-API in a back-end application.",
      "phase": 1,
      "credits": 6
    }
  ]
}
```

]

Error Response:
- Status: 500 Internal Server Error
- Body:
  {
    "error": "Unable to retrieve lecturers"
  }
Mocks:


Lecturers

Firstname	Lastname	E-mail	Expertise
Johan	Pieck	johan.pieck@ucll.be	Full-stack development, Front-end development
Elke	Steegmans	elke.steegmans@ucll.be	Software Engineering, Back-End Development
Greetje	Jongen	greetje.jongen@ucll.be	Full-Stack development, Back-end Development

Get Lecturer byI D

As a student,
I want to retrieve a lecturer by their unique ID,
So that I can view the details of a specific lecturer, including their expertise and user information.

Acceptance Criteria:
- The GET request to '/lecturers/{id}' should return the lecturer with the specified ID.
- If a lecturer with the given ID exists, the response should include:
  - The lecturer's ID
  - The lecturer’s expertise
  - The associated user object (with username, first name, last name, email)
- Fields marked with @JsonIgnore, such as createdAt, updatedAt, and password, must not appear in the response.
- If no lecturer with the given ID exists, the system should return a 404 Not Found response with an appropriate error message.

API Specification:

Endpoint: GET /lecturers/{id}

Description:
Retrieve a single lecturer by their ID, along with associated user information.

Request:
- Method: GET
- URL: /lecturers/{id}
- Path Parameter:
  - id: Long — ID of the lecturer to retrieve
- Headers:
  - Content-Type: application/json

Responses:

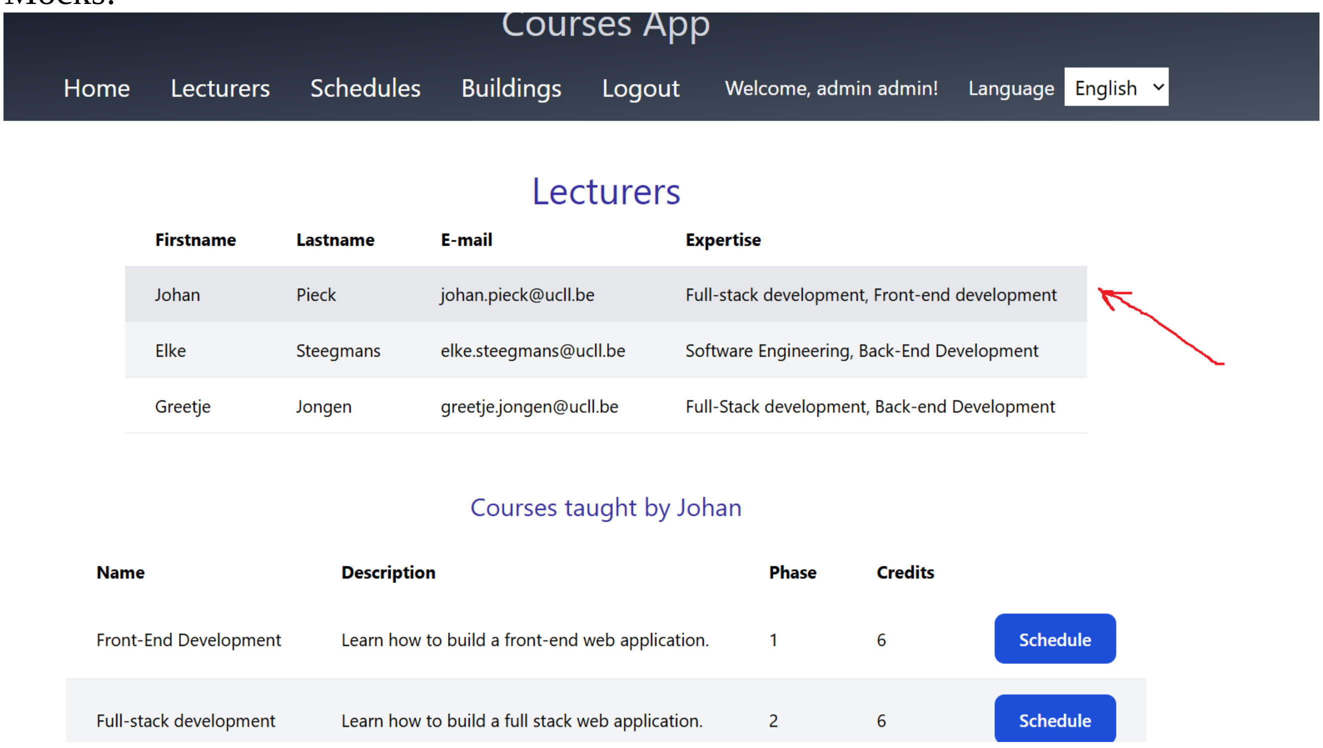
Success Response:
- Status: 200 OK
- Body:

```
 {
 "id": 1,
 "expertise": "Full-stack development, Front-end development",
 "user": {
   "id": 2,
   "username": "johanp",
   "firstName": "Johan",
   "lastName": "Pieck",
   "email": "johan.pieck@ucll.be",
   "role": "lecturer"
 },
 "courses": [
   {
     "id": 3,
     "name": "Front-End Development",
     "description": "Learn how to build a front-end web application.",
     "phase": 1,
     "credits": 6
   },
   {
     "id": 1,
     "name": "Full-stack development",
     "description": "Learn how to build a full stack web application.",
     "phase": 2,
    "credits": 6
   }
 ]
}
```

Error Response (Lecturer Not Found):
- Status: 404 Not Found
- Body:

```
 {
 "status": "not found",
 "message": "Lecturer with id 675 not found"
}
```

Mocks:



Lecturers

Firstname	Lastname	E-mail	Expertise
Johan	Pieck	johan.pieck@ucll.be	Full-stack development, Front-end development
Elke	Steegmans	elke.steegmans@ucll.be	Software Engineering, Back-End Development
Greetje	Jongen	greetje.jongen@ucll.be	Full-Stack development, Back-end Development

Courses taught by Johan

Name	Description	Phase	Credits	
Front-End Development	Learn how to build a front-end web application.	1	6	Schedule
Full-stack development	Learn how to build a full stack web application.	2	6	Schedule

Get schedules

As an authenticated user (Admin or Student),
I want to retrieve a list of schedules relevant to me,
So that I can view all available schedules (if I am an admin) or only the ones I'm enrolled in (if I am a student).

Acceptance Criteria:
- The GET request to '/schedules' should return a list of `Schedule` objects.
- If the user has the `ADMIN` role:

  - The endpoint should return **all** schedules in the system.
- If the user has the `STUDENT` role:
  - The endpoint should return only schedules where the lecturer's user matches the authenticated student's username.
- If the user has neither role, access must be denied with a 403 Forbidden status.
- Fields marked with `@JsonIgnore` must not be shown in the response.

API Specification:

Endpoint: GET /schedules

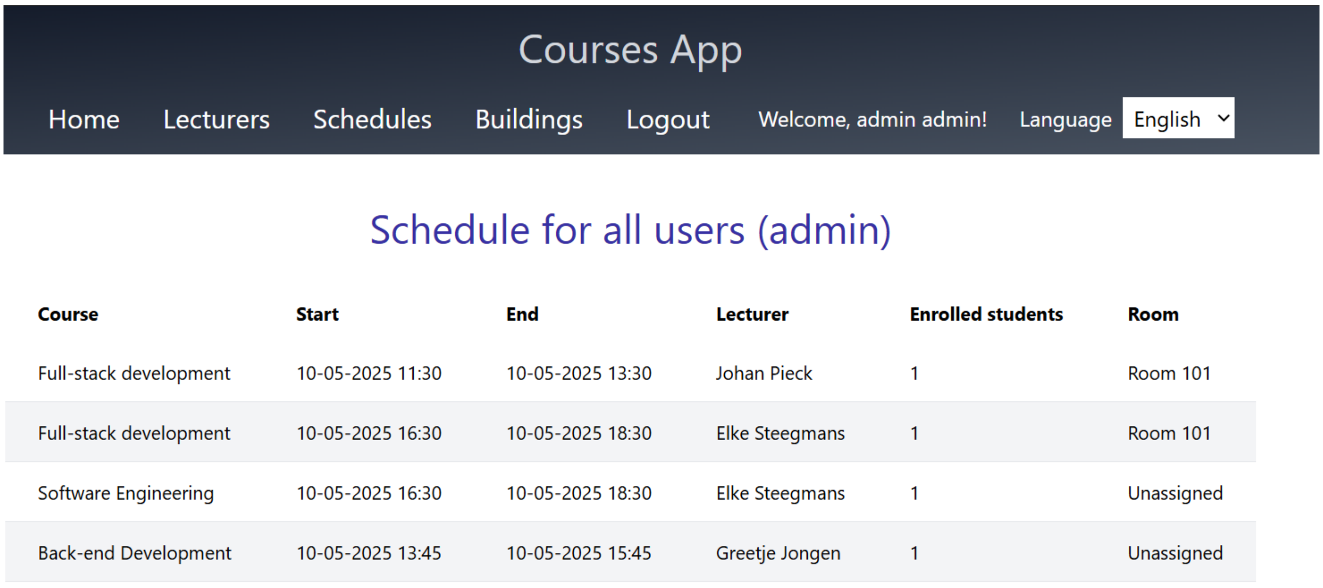
Description:
Retrieve all schedules for admins, or relevant schedules for students based on authentication.

Request:
- Method: GET
- URL: /schedules
- Headers:
  - Authorization: Bearer <JWT Token>
  - Content-Type: application/json

Responses:

 Success Response (Admin):
- Status: 200 OK
- Body:
```json
[
  {
    "id": 1,
    "start": "2025-04-06T05:30:00Z",
    "end": "2025-04-06T07:30:00Z",
    "course": {
      "id": 1,
      "name": "Full-stack development",
      "description": "Learn how to build a full stack web application.",
      "phase": 2,
      "credits": 6
    },
    "lecturer": {
      "id": 1,
      "expertise": "Full-stack development, Front-end development",
      "user": {
        "id": 2,
        "username": "johanp",
        "firstName": "Johan",
        "lastName": "Pieck",
        "email": "johan.pieck@ucll.be",
        "role": "lecturer"
      },
      "courses": [
        {
          "id": 3,
          "name": "Front-End Development",
          "description": "Learn how to build a front-end web application.",
          "phase": 1,
          "credits": 6
        },
        {
          "id": 1,
          "name": "Full-stack development",
          "description": "Learn how to build a full stack web application.",
          "phase": 2,
          "credits": 6
        }
      ]
    },
    "students": [
      {
        "id": 1,
        "user": {
          "id": 5,
          "username": "peterp",
          "firstName": "Peter",
          "lastName": "Parker",
          "email": "peter.parker@student.ucll.be",
          "role": "student"
        },
        "studentnumber": "r0785023"
      }
    ]
  }
]
```

error:
```
{
  "error": "You do not have permission to access this resource"
}
```

Mocks:


Schedule for all users (admin)

Course	Start	End	Lecturer	Enrolled students	Room
Full-stack development	10-05-2025 11:30	10-05-2025 13:30	Johan Pieck	1	Room 101
Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned

Create Schedule

As an administrator,
I want to create a new schedule by providing a course, lecturer, and time range,
So that I can assign a lecturer to teach a specific course at a specific time.

Acceptance Criteria:
- The POST request to `/schedules` should accept a JSON body representing a `ScheduleInput`.
- The `ScheduleInput` must include:
  - Start time (`start`)
  - End time (`end`)
  - Course reference (with at least `id`)
  - Lecturer reference (with at least `id`)
- If the course or lecturer does not exist, a `404 Not Found` should be returned.
- If a schedule already exists for the given course and lecturer, the system should reject it with a `400 Bad Request` and an appropriate error message.
- Upon success, the created schedule should be returned with:
  - `id`

  - `start` and `end`
  - Associated `course`
  - Associated `lecturer`
  - Empty student list
- `createdAt` and `updatedAt` fields must not appear in the response (`@JsonIgnore`).

API Specification:

Endpoint: POST /schedules
Description: Create a new schedule entry

Request:
- Method: POST
- URL: /schedules
- Headers:
  - Authorization: Bearer <JWT Token>
  - Content-Type: application/json
- Body:
```json
{
  "start": "2025-05-01T09:00:00Z",
  "end": "2025-05-01T11:00:00Z",
  "course": {
    "id": 1
  },
  "lecturer": {
    "id": 2
  }
}
```

Response:
Success (201)
Body:
```
{
  "id": 10,
  "start": "2025-05-01T09:00:00Z",
  "end": "2025-05-01T11:00:00Z",
  "course": {
    "id": 1,
    "title": "Physics 101",
    "description": "Introduction to Physics"
  },
  "lecturer": {
    "id": 2,
    "expertise": "Physics",
    "user": {
      "id": 5,
      "username": "dr_einstein",
      "firstName": "Albert",
      "lastName": "Einstein",
      "email": "albert.einstein@example.com"
    }
  },
  "students": []
}
```

Errors:

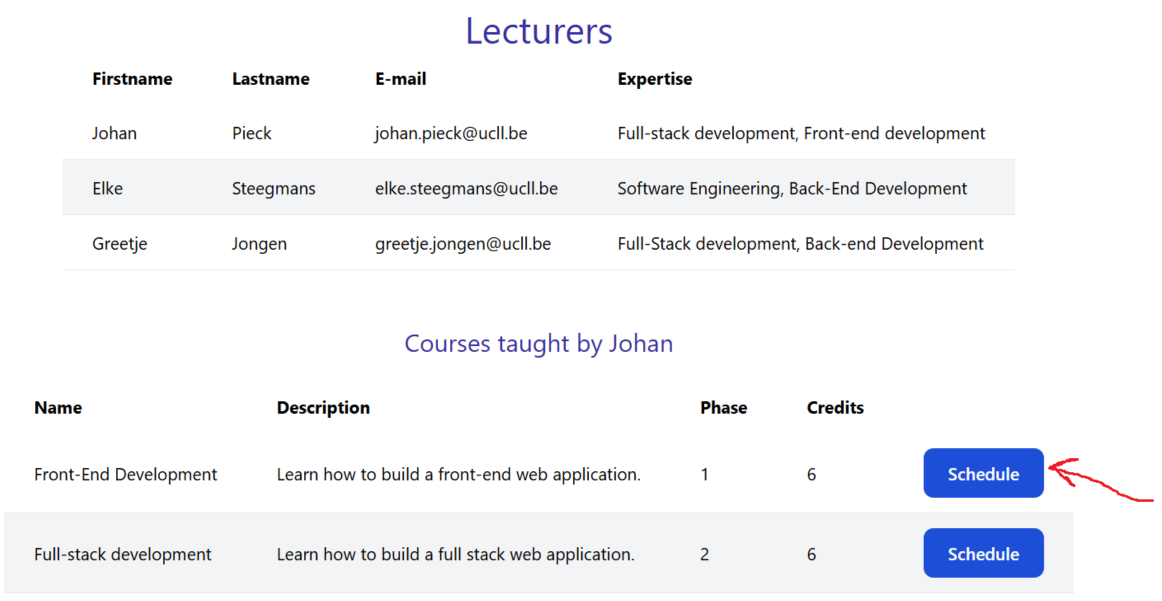
Status: 404 Not Found
Body:

```
{
  "error": "Course with id 1 not found"
}
```

Status: 400 Bad Request
Body:

```
{
  "error": "Schedule already exists for course with id 1 and lecturer with id 2"
}
```

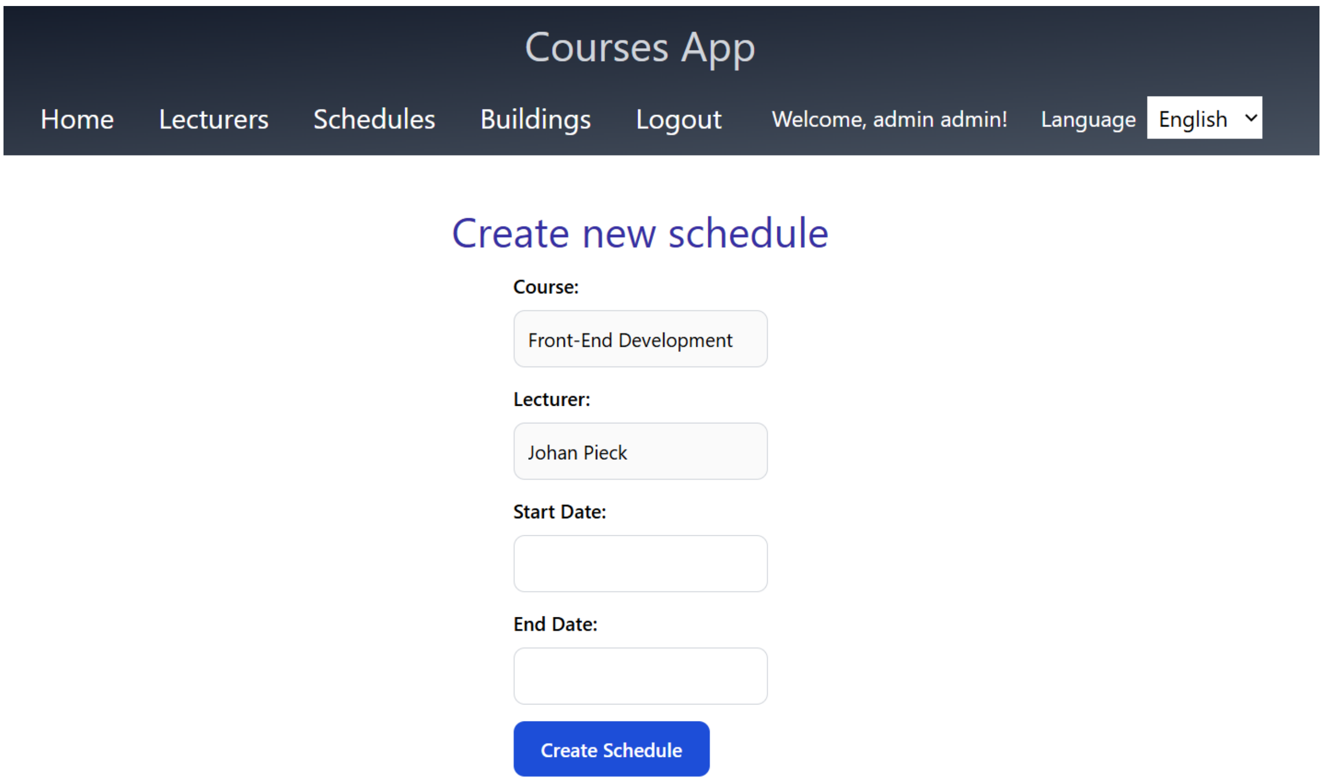
Mocks:


Lecturers

Firstname	Lastname	E-mail	Expertise
Johan	Pieck	johan.pieck@ucll.be	Full-stack development, Front-end development
Elke	Steegmans	elke.steegmans@ucll.be	Software Engineering, Back-End Development
Greetje	Jongen	greetje.jongen@ucll.be	Full-Stack development, Back-end Development

Courses taught by Johan

Name	Description	Phase	Credits	
Front-End Development	Learn how to build a front-end web application.	1	6	Schedule
Full-stack development	Learn how to build a full stack web application.	2	6	Schedule



Create new schedule

Course: Front-End Development

Lecturer: Johan Pieck

Start Date:

End Date:

Enroll schedule

As an administrator,
I want to enroll multiple students into a specific schedule,
So that they can attend the scheduled course with a specific lecturer.

Acceptance Criteria:

- The POST request to `/schedules/enroll` accepts a JSON body representing `EnrollmentInput`.
- The `EnrollmentInput` must include:
  - A valid `schedule` object with at least its `id`.
  - A list of `students`, each with at least their `id`.
- If the schedule does not exist, the system should respond with `404 Not Found`.
- If any student in the list does not exist, the system should respond with `404 Not Found` and a clear message.
- Students should be added to the `students` list of the schedule (if not already enrolled).
- The updated schedule is saved and returned in the response.
- `createdAt` and `updatedAt` should not appear in the response (`@JsonIgnore`).

API Specification:

Endpoint: POST `/schedules/enroll`
Description: Enroll one or more students into a given schedule.

Request:
- Method: POST
- Headers:
  - Authorization: Bearer <JWT Token>
  - Content-Type: application/json
- Body:

```json
{
  "schedule": {
    "id": 10
  },
  "students": [
    { "id": 1 },
    { "id": 2 },
    { "id": 3 }
  ]
}
```

Success Response:
Status: 200 OK
Body:

```
{
  "id": 10,
  "start": "2025-05-01T09:00:00Z",
  "end": "2025-05-01T11:00:00Z",
  "course": {
    "id": 1,
    "title": "Physics 101"
  },
  "lecturer": {
    "id": 2,
    "user": {
      "id": 5,
      "username": "dr_einstein"
    }
  },
  "students": [
    { "id": 1, "firstName": "John", "lastName": "Doe" },
    { "id": 2, "firstName": "Jane", "lastName": "Smith" },
    { "id": 3, "firstName": "Alice", "lastName": "Brown" }
  ]
}
```

Error:
Status: 404 Not Found
Body:

```
{
  "error": "Schedule with id 10 not found"
}
```

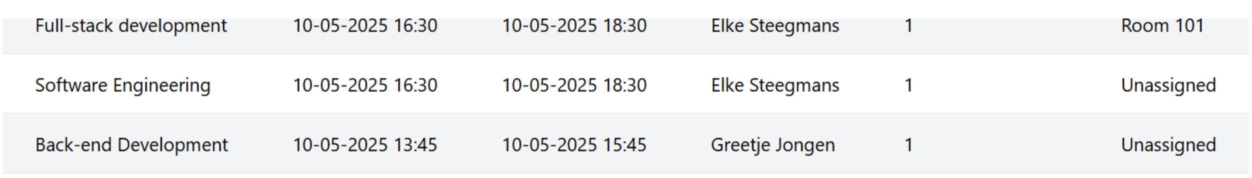
Error: One of the students not found

Status: 404 Not Found

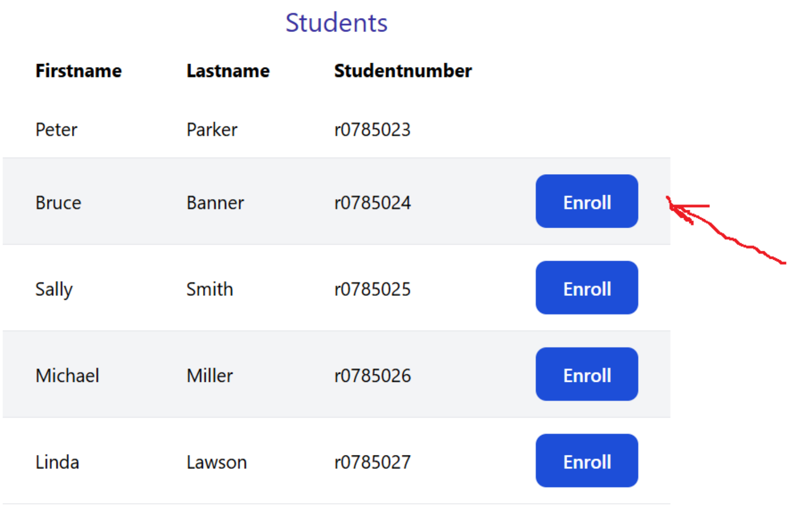
Body:

```
{ "error": "Student with id 2 not found" }
```

Same steps as in Get Students mocks and then click enroll



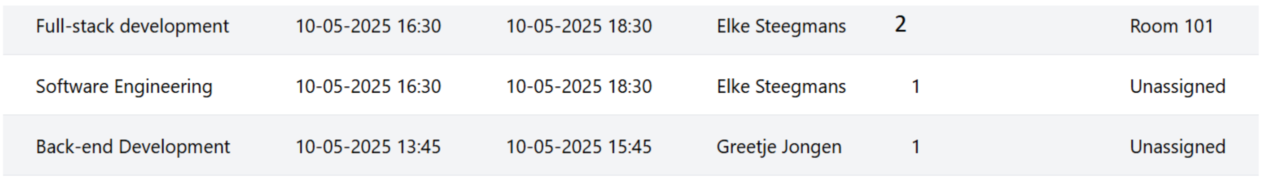
Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned



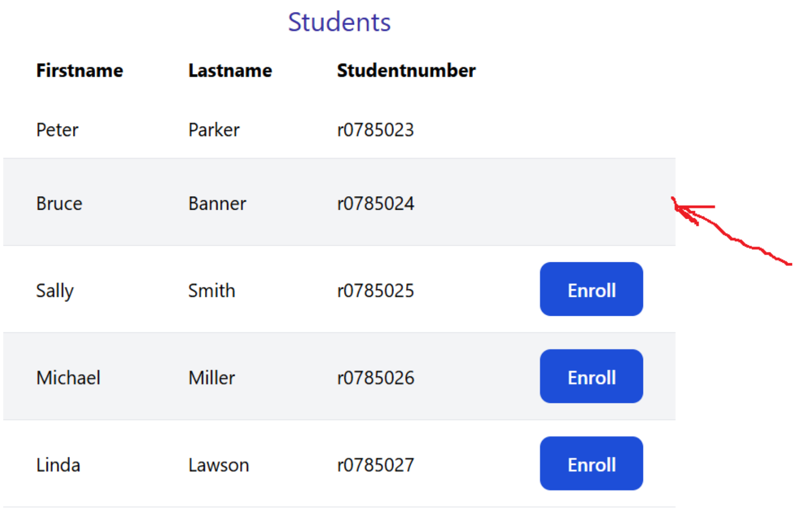
Students

Firstname	Lastname	Studentnumber	
Peter	Parker	r0785023	
Bruce	Banner	r0785024	Enroll
Sally	Smith	r0785025	Enroll
Michael	Miller	r0785026	Enroll
Linda	Lawson	r0785027	Enroll

Then:



Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	2	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned



Students

Firstname	Lastname	Studentnumber	
Peter	Parker	r0785023	
Bruce	Banner	r0785024	
Sally	Smith	r0785025	Enroll
Michael	Miller	r0785026	Enroll
Linda	Lawson	r0785027	Enroll

Assign room to schedule

User story:

As a lecturer, I want to assign rooms to schedules so that classes have designated locations.

API Specification:

Endpoint: Assign a room to a schedule
Method: POST
URL: /schedules/{scheduleId}/assign-room

Request Body:

```
{
  "roomId": 1
}
```

Response (Success):

Status: 200 OK

```
{
  "id": 10,
  "start": "2024-04-01T10:00:00Z",
  "end": "2024-04-01T12:00:00Z",
  "course": {
    "id": 5,
    "name": "Software Engineering"
  },
  "lecturer": {
    "id": 2,
    "name": "Dr. Smith"
  },
  "room": {
    "id": 1,
    "name": "Room 101",
    "building": {
      "id": 3,
      "name": "Main Building"
    }
  }
}
```

Response (Failure - Room Not Found):

Status: 404 Not Found

```
{
   "error": "Room not found"
}
```

Response (Failure - Schedule Not Found):

Status: 404 Not Found

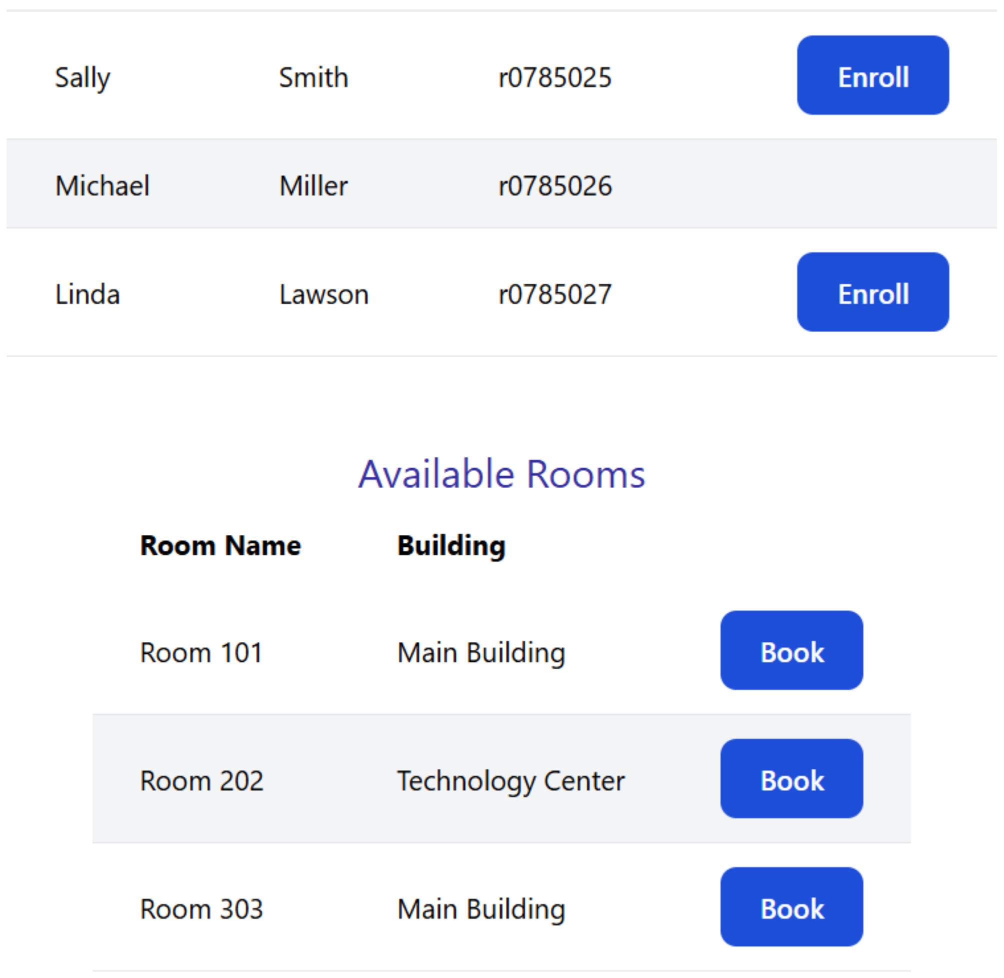
```
{
   "error": "Schedule not found"
}
```

Response (Failure - Room Already Assigned):

Status: 400 Bad Request

```
{
   "error": "Room is already assigned to another schedule at this time"
}
```

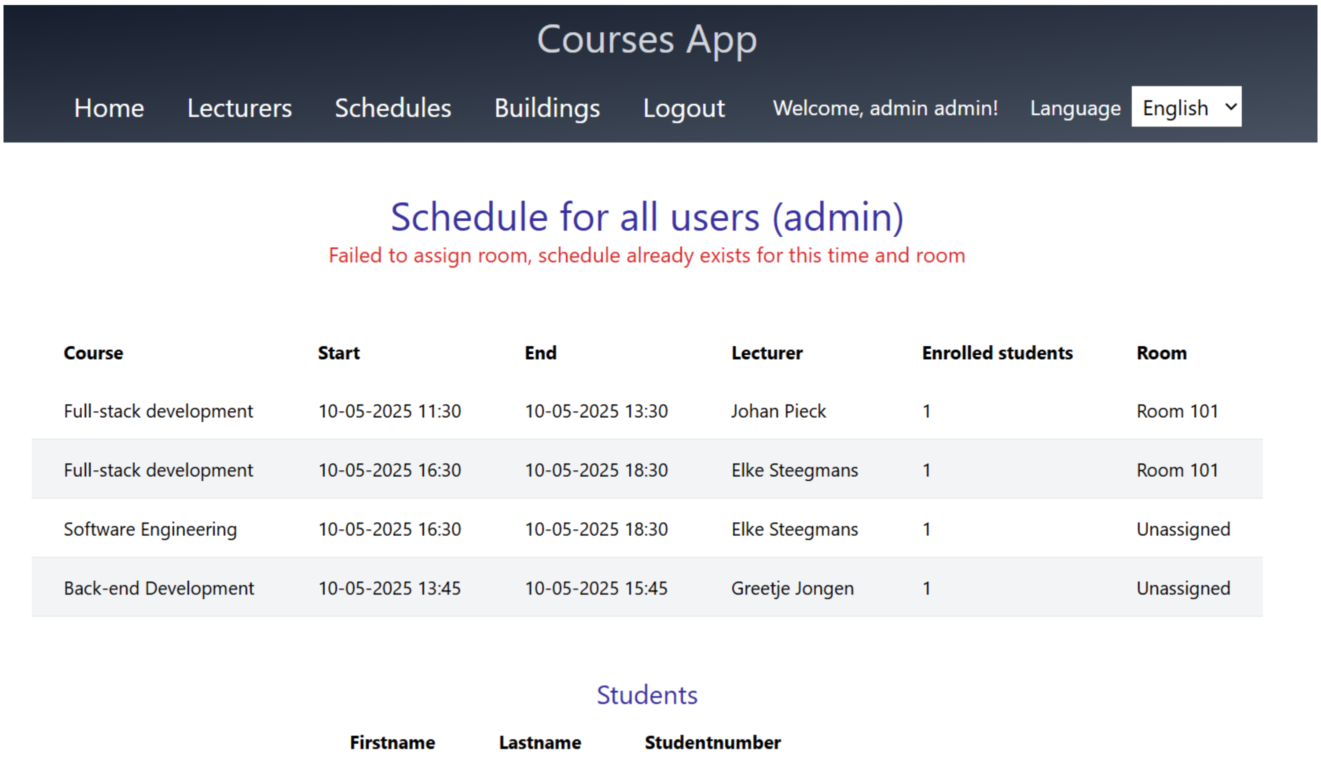
Mocks:
Same steps as in get students (you need to click on schedule. Below on the page you see available rooms. If schedule already assigned to rooms, buttons should not be visible. Otherwise you should see a button Book)



Sally	Smith	r0785025	Enroll
Michael	Miller	r0785026	
Linda	Lawson	r0785027	Enroll

Available Rooms

Room Name	Building	
Room 101	Main Building	Book
Room 202	Technology Center	Book
Room 303	Main Building	Book



Schedule for all users (admin)

Failed to assign room, schedule already exists for this time and room

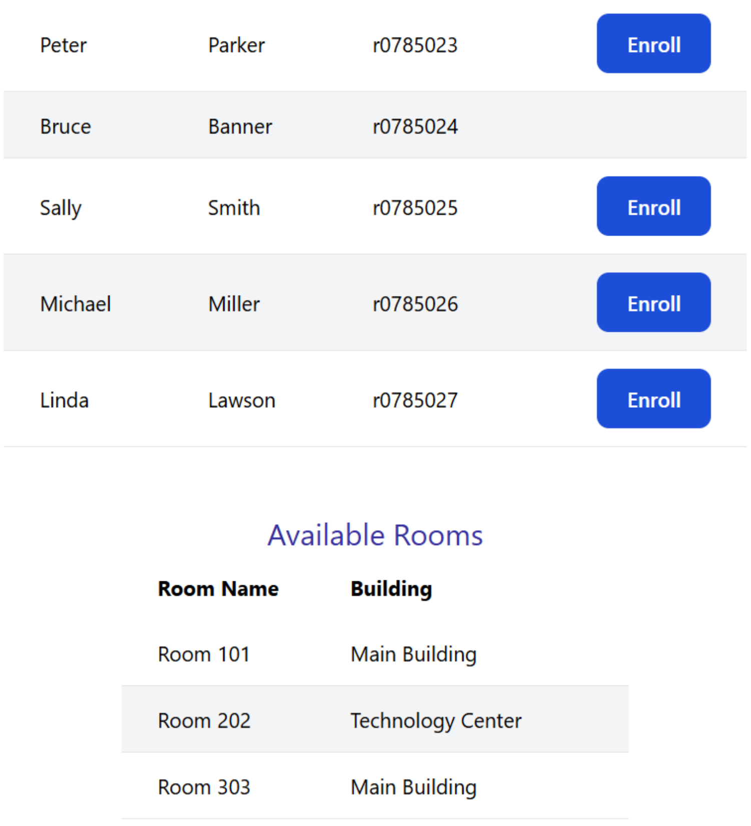
Course	Start	End	Lecturer	Enrolled students	Room
Full-stack development	10-05-2025 11:30	10-05-2025 13:30	Johan Pieck	1	Room 101
Full-stack development	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Room 101
Software Engineering	10-05-2025 16:30	10-05-2025 18:30	Elke Steegmans	1	Unassigned
Back-end Development	10-05-2025 13:45	10-05-2025 15:45	Greetje Jongen	1	Unassigned

Students

Firstname	Lastname	Studentnumber

If you try to assign room to the schedule, and room already has a schedule

And finally, if schedule assigned to room successfully you should see appropriate message and buttons should not be visible anymore:

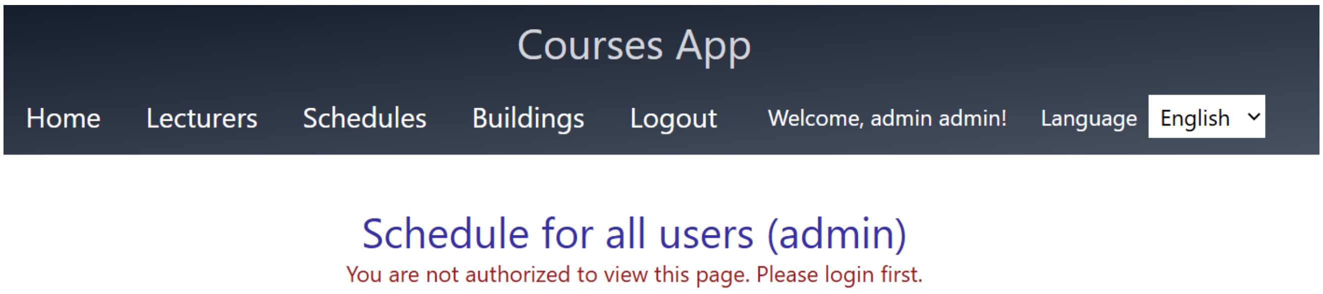


Peter	Parker	r0785023	Enroll
Bruce	Banner	r0785024	
Sally	Smith	r0785025	Enroll
Michael	Miller	r0785026	Enroll
Linda	Lawson	r0785027	Enroll

Available Rooms

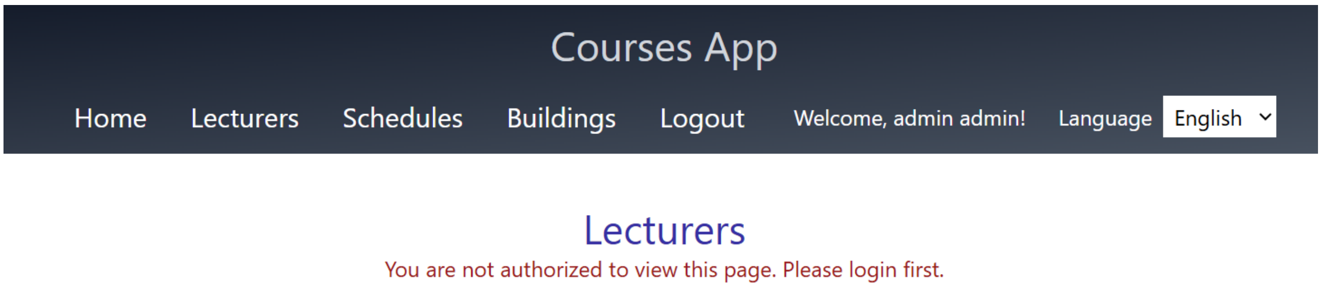
Room Name	Building
Room 101	Main Building
Room 202	Technology Center
Room 303	Main Building

Not authorized mocks:


Courses App

Schedule for all users (admin)

You are not authorized to view this page. Please login first.


Courses App

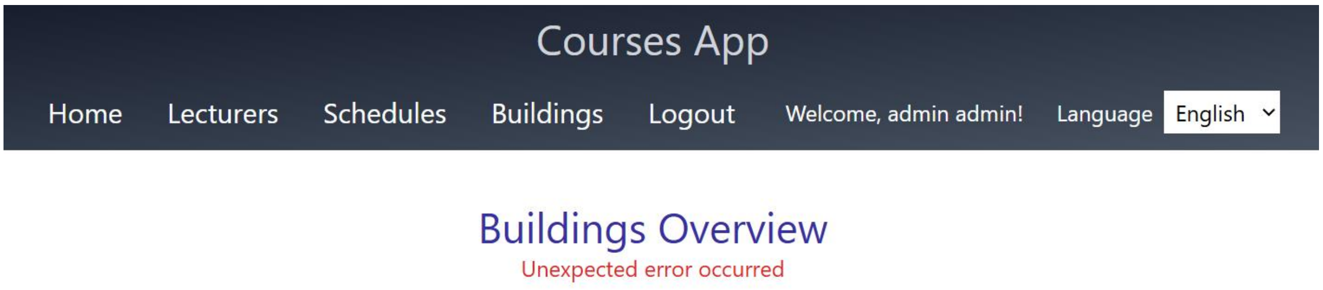
Lecturers

You are not authorized to view this page. Please login first.

Building Overview Story

Story:

As a user, I want to view a list of all buildings along with their rooms and associated schedules so that I can see which rooms are in use and when.


Courses App

Buildings Overview

Unexpected error occurred

API Specification

GET /buildings

Description:

Returns all buildings with their rooms and each room’s associated schedules.

Request:

No parameters required.

Response:

- **Status Code:** 200 OK
- **Content-Type:** application/json
- **Body:**

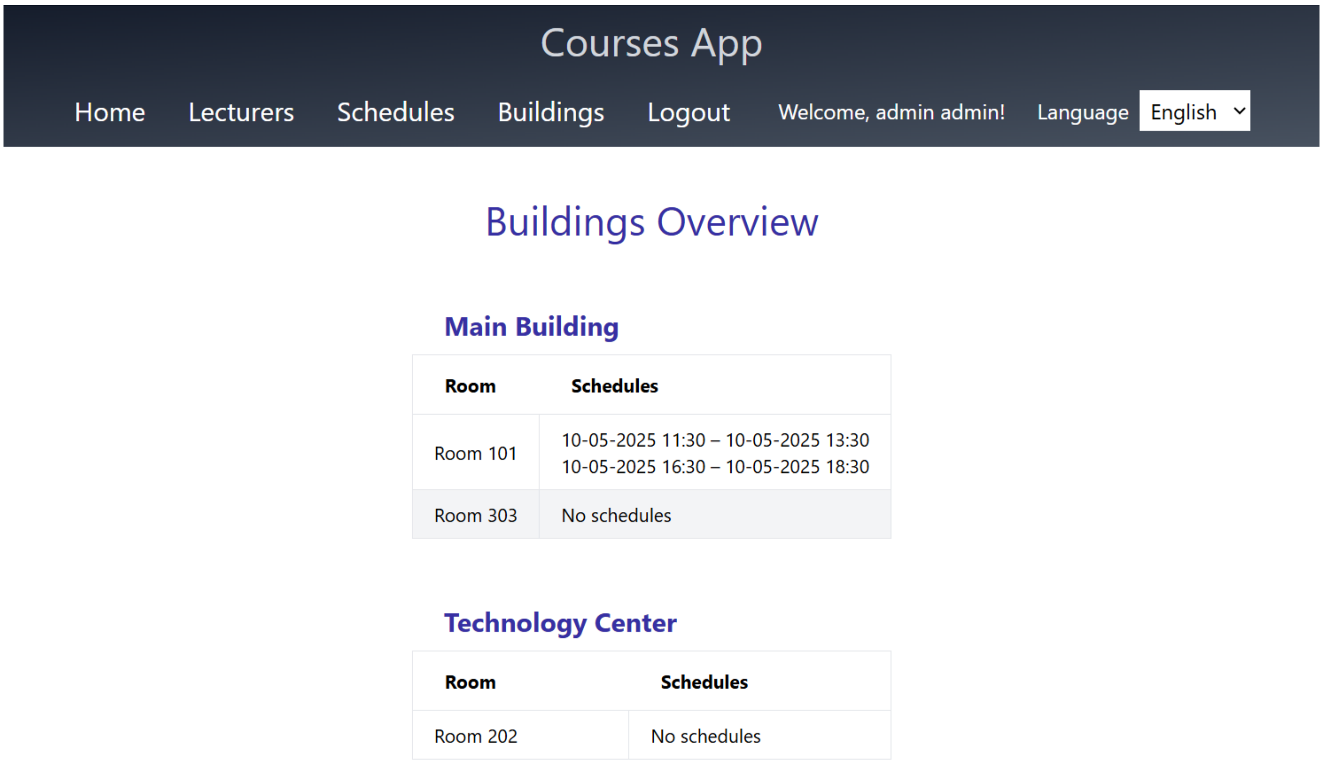
```
[
  {
    id: number,
    name: string,
    rooms: [
      {
        id: number,
        name: string,
        schedules: [
          {
            id: number,
            start: string (ISO 8601),
            end: string (ISO 8601),
            courseName: string
          }
        ]
      }
    ]
  }
]
```

1. Sending a GET request to /buildings returns a list of all buildings.
2. Each building includes:
    - id
    - name
    - a list of associated rooms.
3. Each room includes:
    - id
    - name
    - a list of associated schedules.
4. Each schedule includes:
    - id
    - start (date-time)

    - end (date-time)
    - courseName
5. The response is in JSON format.
6. No authentication is required to access this endpoint (unless configured otherwise).
7. The output matches the following structure:

```
[
  {
    "id": 9007199254740991,
    "name": "Main Building",
    "rooms": [
      {
        "id": 1,
        "name": "Room 101",
        "schedules": [
          {
            "id": 1,
            "start": "2025-05-11T08:30:00.000Z",
            "end": "2025-05-11T10:30:00.000Z",
            "courseName": "Physics"
          }
        ]
      }
    ]
  }
]
```

Mocks:



Buildings Overview

Main Building

Room	Schedules
Room 101	10-05-2025 11:30 – 10-05-2025 13:30 10-05-2025 16:30 – 10-05-2025 18:30
Room 303	No schedules

Technology Center

Room	Schedules
Room 202	No schedules