


SERVICE LAYER

MARYVILLE HUB

The service layer in the Maryville Hub web application serves as the intermediary between the user interface and the underlying database. It summarizes the complexities of data storage and retrieval, providing a clean set of endpoints that the user interface can interact with.

This separation of concerns ensures that changes in the database schema or the user interface design do not affect each other as long as the service layer's contract remains consistent.

1. **RESTful Principles:** The service layer adheres to RESTful design principles, using standard HTTP methods like GET, POST, PUT, and DELETE to perform CRUD (Create, Read, Update, Delete) operations on resources.
2. **Structured Endpoint Paths:** Each service endpoint has a clear and logical URL path that indicates the resource it operates on and its actions.
3. **User Goals Alignment:** The service endpoints are designed with the user in mind, ensuring that each endpoint directly supports the goals and tasks that users aim to accomplish within the application.

- 
4. **Request and Response Validation:** The service layer handles data validation, ensuring that requests contain all the necessary and correctly formatted information before processing. It also constructs appropriate responses, including error messages when submitting invalid data.
 5. **Error Handling:** The service layer handles errors, providing error messages and HTTP status codes to the client in case of issues like invalid input, resource not found, or server errors.
 6. **Scalability and Performance:** The service layer is designed to be scalable, handling an increasing number of requests as the user base grows. It optimizes database interactions to ensure quick and responsive performance.

Specifications

I'm in the process of setting up the backend services and designing the necessary service endpoints for the web application. I will be using a placeholder API URL: **<https://placeholder-url.com/api/endpoint>**. Once I decide on the deployment service, I will update the API URL to the backend services.

Create User

- Method: **POST**
- Url: **https://placeholder-url.com/api/users**
- Purpose: Allows new users to register on Maryville Hub.
- Example Request:

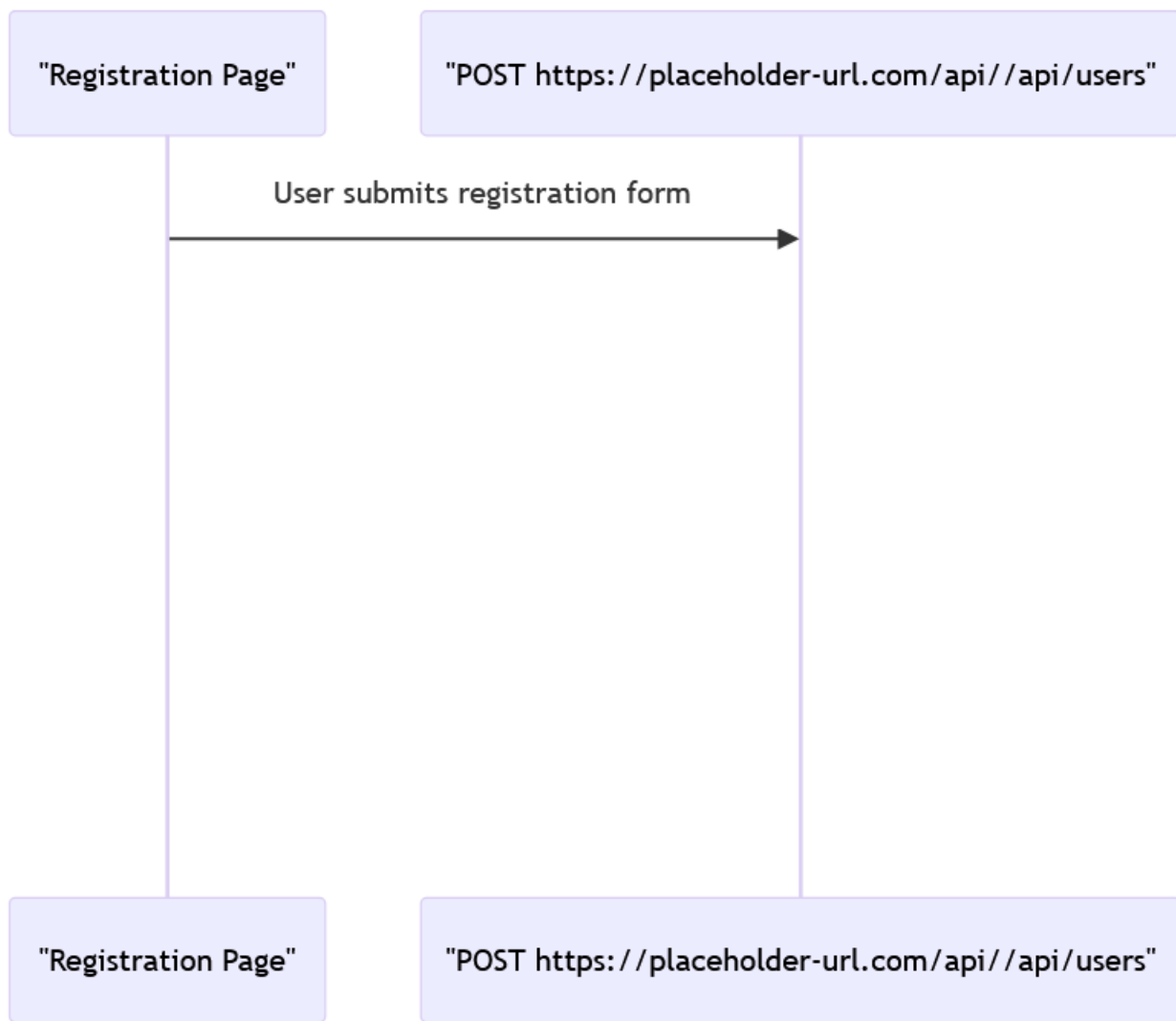
```
{
  "username": "jdoe",
  "email": "jdoe@example.com",
  "password": "securepassword",
  "fullName": "John Doe",
  "major": "Computer Science",
  "profileImg": "path_to_profile_image.jpg"
}
```

Example Response:

```
{
  "success": true,
  "message": "User registered successfully."
}
```

Error Response:

```
{
  "success": false,
  "message": "User already exists."
}
```



User Login

- Method: **POST**
- Url: **https://placeholder-url.com/api/users/login**
- Purpose: Authenticates users and provides them with access to their accounts.
- Example Request:

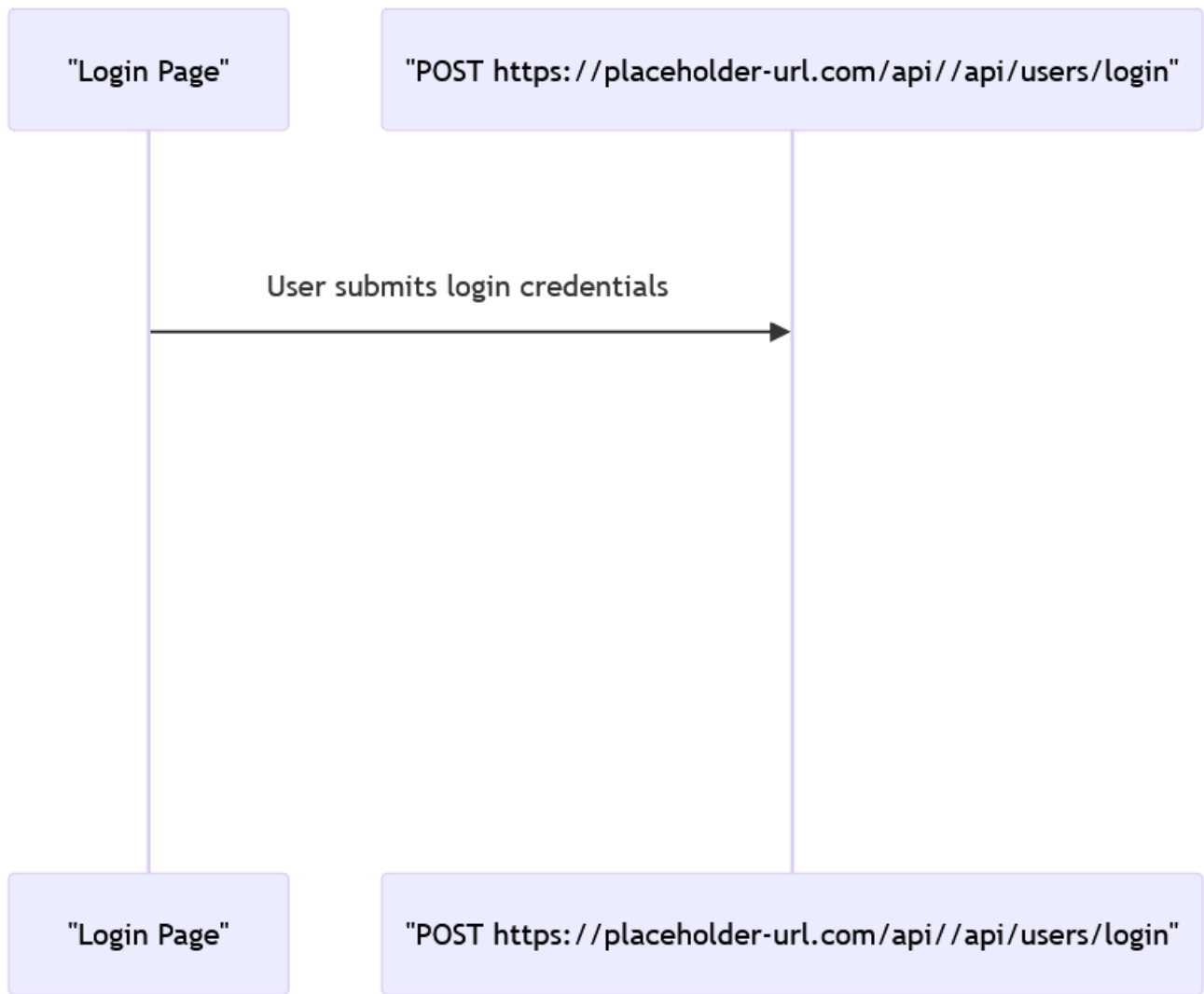
```
{  
  "username": "jdoe",  
  "password": "securepassword"  
}
```

Example Request:

```
{  
  "success": true,  
  "token": "JWT_token_here"  
}
```

Example Error:

```
{  
  "success": false,  
  "message": "Invalid username or password."  
}
```



Enroll in Course

- Method: **POST**
- Url: **<https://placeholder-url.com/api/courses/enroll>**
- Purpose: Allows students to enroll in courses.
- Example Request:

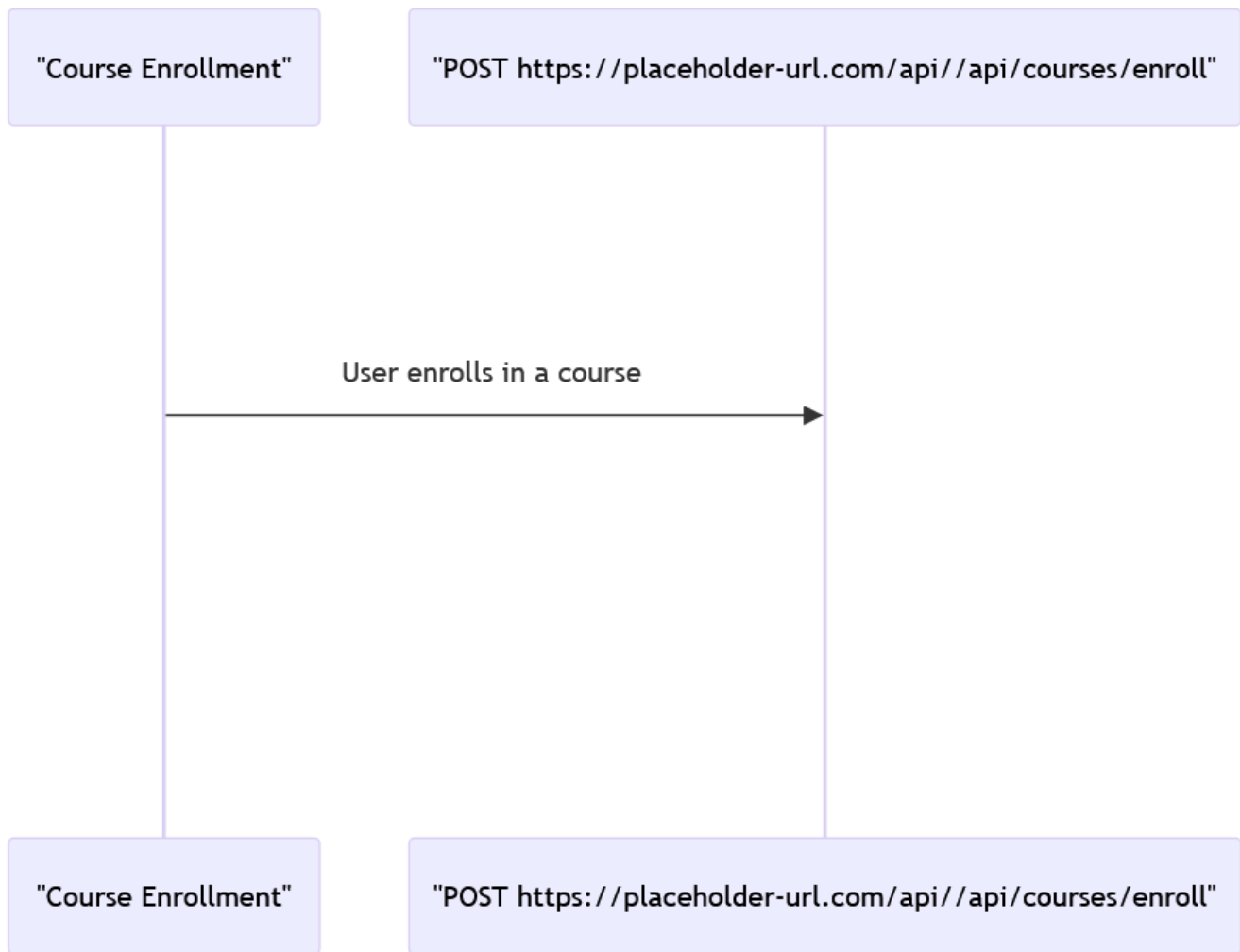
```
{  
  "courseId": "course123",  
  "userId": "user456"  
}
```

Example Response:

```
{  
  "success": true,  
  "message": "Enrollment successful."  
}
```

Error Response:

```
{  
  "success": false,  
  "message": "Course not found or already enrolled."  
}
```



Create Post

- Method: **POST**
- Url: **<https://placeholder-url.com/api/community/posts>**
- Purpose: Allows users to create new posts in the community forum.
- Example Request:

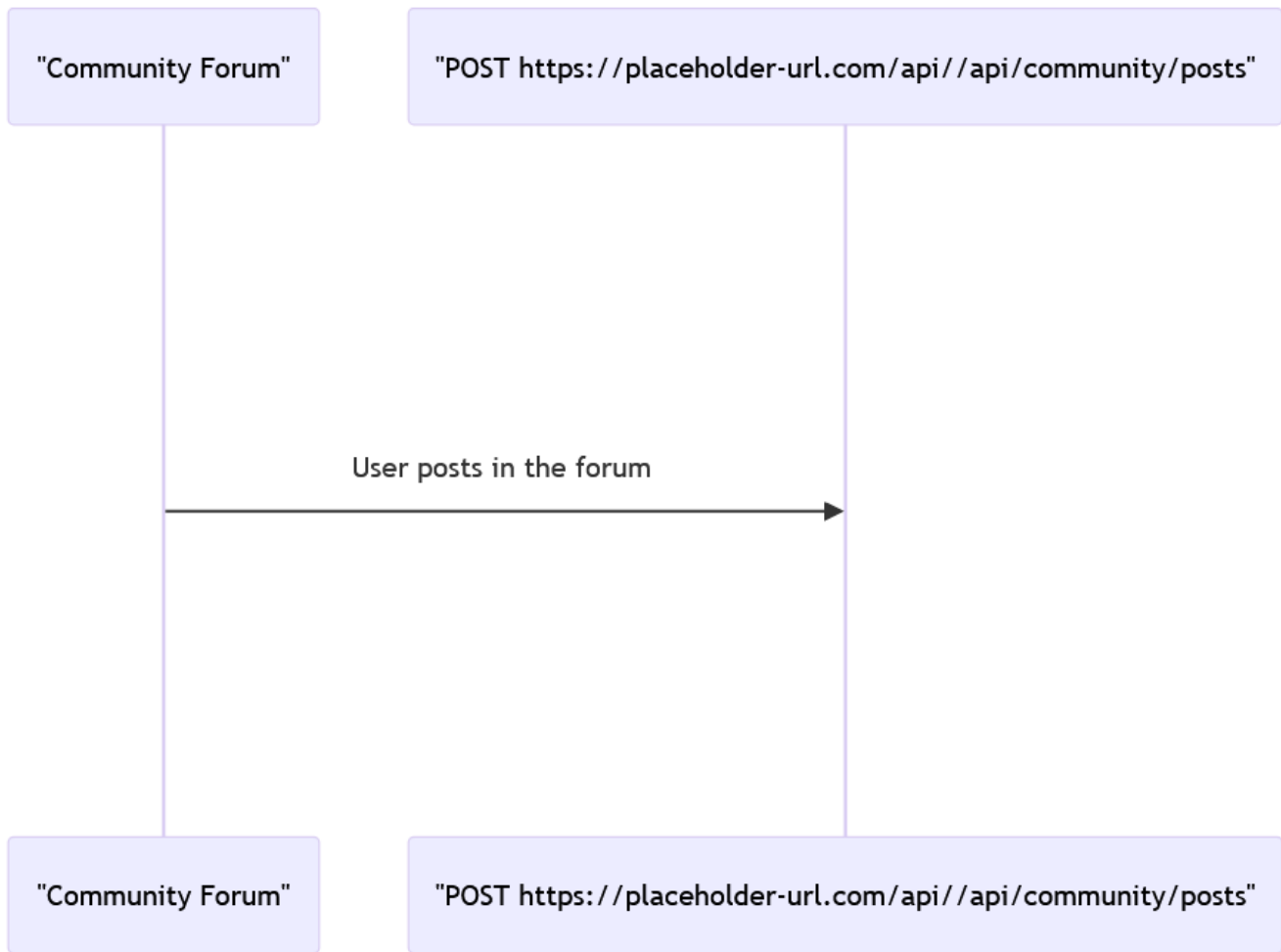
```
{  
  "title": "Study Group",  
  "content": "Looking for group members for CS101.",  
  "authorId": "user456"  
}
```

Example Response:

```
{  
  "success": true,  
  "message": "Post created successfully."  
}
```

Example Error:

```
{  
  "success": false,  
  "message": "Invalid data provided."  
}
```



Add Calendar Event

- Method: **POST**
- Url: **<https://placeholder-url.com/api/community/events>**
- Purpose: Allows users to create new posts in the community forum.
- Example Request:

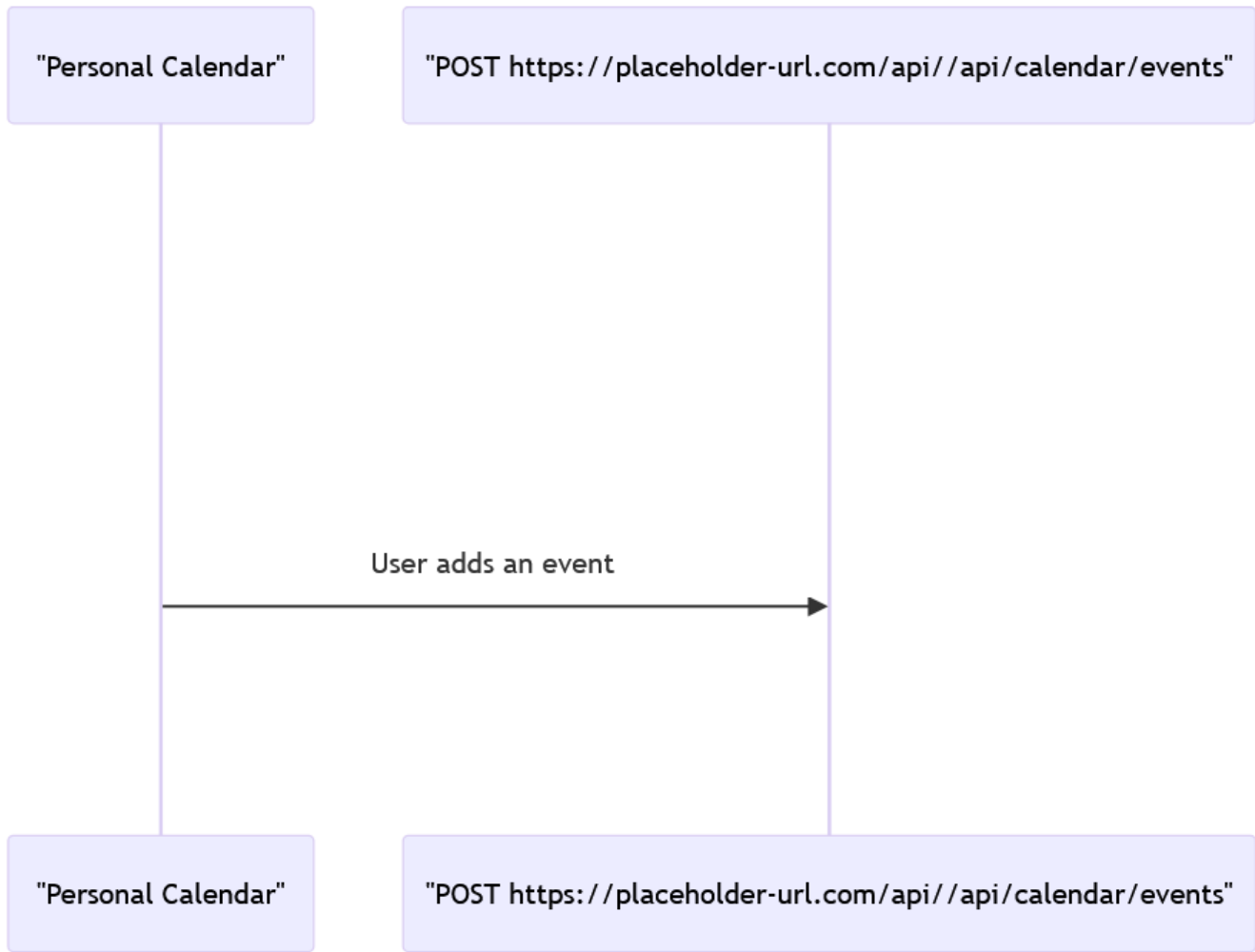
```
{  
  "title": "Midterm Exam",  
  "description": "CS101 Midterm",  
  "date": "2023-11-10",  
  "userId": "user456"  
}
```

Example Response:

```
{  
  "success": true,  
  "message": "Event added successfully."  
}
```

Example Error:

```
{  
  "success": false,  
  "message": "Invalid date or missing information."  
}
```



Attend Event

- Method: **POST**
- Url: **<https://placeholder-url.com/api/events/attend>**
- Purpose: Allows users to mark their attendance for university events.
- Example Request:

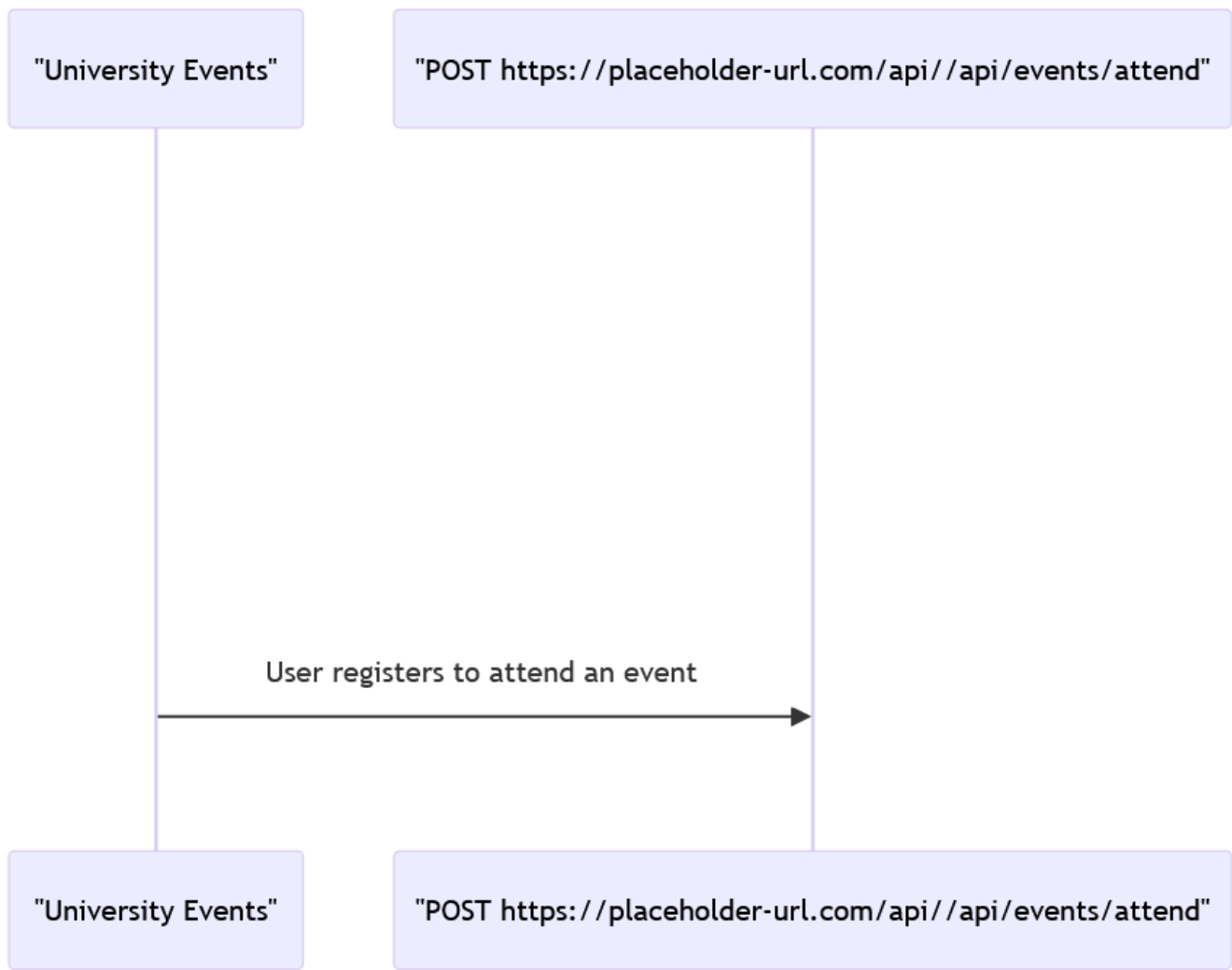
```
{  
  "eventId": "event789",  
  "userId": "user456"  
}
```

Example Response:

```
{  
  "success": true,  
  "message": "Attendance marked successfully."  
}
```

Example Error:

```
{  
  "success": false,  
  "message": "Event not found or already attending."  
}
```



Get User Profile

- Method: **GET**
- Url: **https://placeholder-url.com/api/users/{userId}/profile**
- Purpose: Retrieves the profile information of a user.
- Example Request:

```
{
  "success": true,
  "data": {
    "username": "jdoe",
    "fullName": "John Doe",
    "major": "Computer Science",
    "enrolledCourses": [
      {
        "courseId": "course123",
        "title": "Introduction to Computer Science"
      },
      {
        "courseId": "course456",
        "title": "Data Structures"
      }
    ],
    "upcomingEvents": [
      {
        "eventId": "event789",
        "title": "Career Fair",
        "date": "2023-12-05"
      }
    ]
  }
}
```

Error Response:

```
{
  "success": false,
  "message": "User not found."
}
```

