SPL-02 Project Report

Dhaka University Swimming Pool Management System

Submitted by

Ibne Bin Rafid (BSSE1330)

Eftekhar Mahmud Efty (BSSE 1309)

Supervised by

Abdus Satter

Assistant Professor

Institute of Information Technology

University of Dhaka

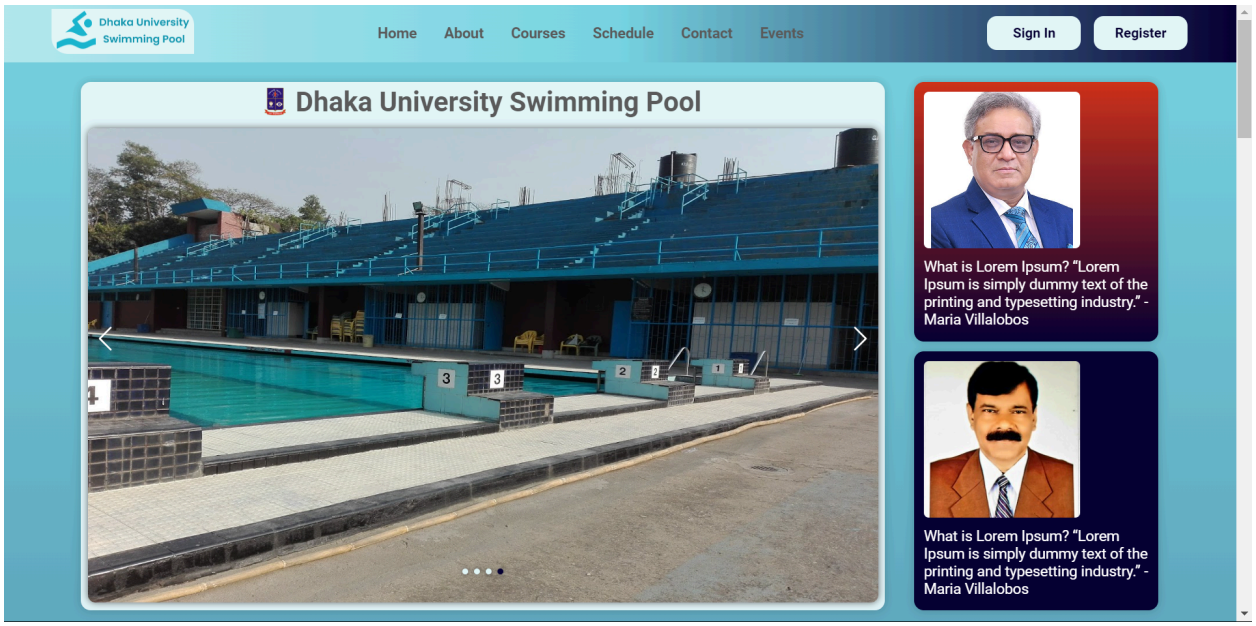
[13-06-2024]

Signature of Supervisor

Table of Contents

1. Introduction

In response to the growing need for efficient management of Dhaka University's swimming pools, we propose the development of a comprehensive Swimming Pool Management System. This system aims to streamline various administrative tasks, enhance user experience, and improve overall operational efficiency.



2. Background of the Project

2.1. Swimming Pool Management System

The Swimming Pool Management System is designed to automate and optimize the management of swimming pool facilities at Dhaka University. This includes user registrations, scheduling, financial transactions, and communication with stakeholders.

2.2. Web Application Development

The project will be implemented as a web-based platform, utilizing modern web development technologies such as ReactJS for the frontend and NodeJS for the backend.

2.3. Database Management

A robust database management system, specifically MongoDB, will be employed

to securely store and manage user information, schedules, and financial data.

3. Description of the Project

3.1. User Management

The system will provide functionalities for user registration, login, and profile management. Each user will have a personalized account for accessing the swimming pool facilities.

3.2. Schedule Management

The system will allow administrators to create and manage schedules for various activities such as swimming lessons, training sessions, and maintenance periods. Users can view and register for these activities.

3.3. Financial Management

The system will facilitate online transactions, allowing users to make payments for various services. It will also provide financial insights for administrators, helping them track the pool's profit and loss.

3.4. Communication and Notification

Administrators will be able to send notifications and announcements to users. Users can also provide feedback and reviews, fostering engagement and transparency.

4. Implementation

4.1. Frontend

4.1.1. Styling

The frontend is styled using raw CSS to ensure a responsive and user-friendly interface. Consistent and responsive design principles are applied to ensure usability across various devices.

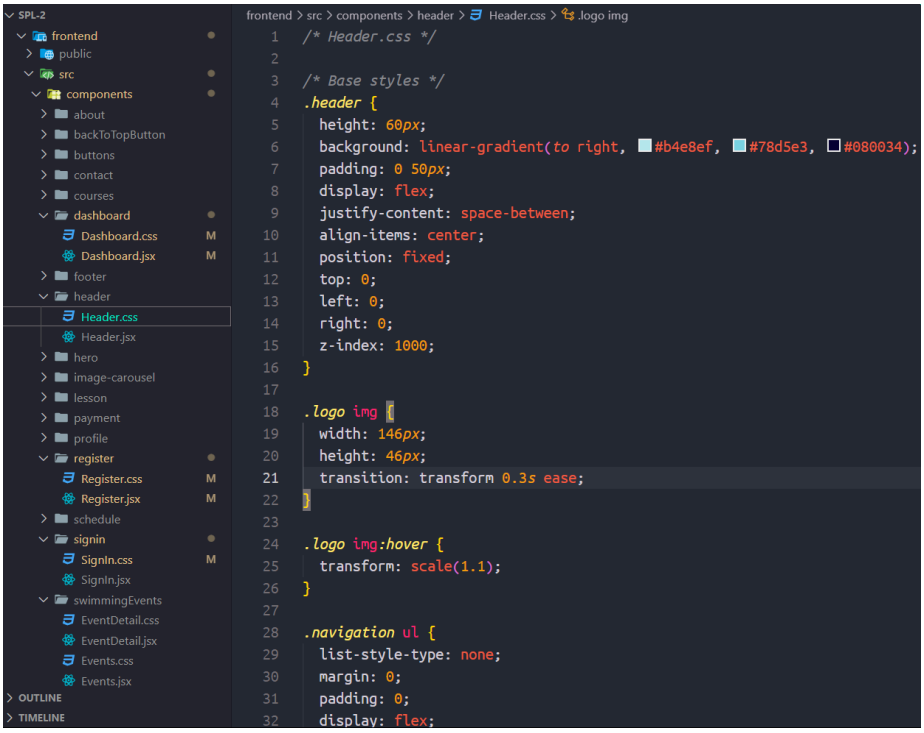


Figure: Example styling of the Header component

4.1.2. Components Structure

The application is modularized into reusable components, making the development process more efficient and the codebase easier to maintain.

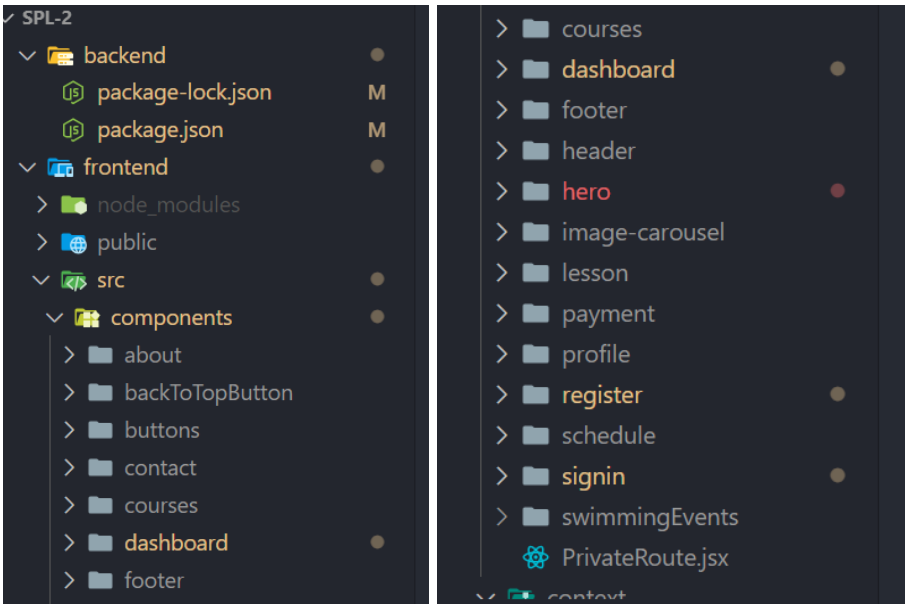


Figure: Components used in frontend

4.1.3. User Interaction

The system is providing interactive elements such as authentication forms, schedules, transaction, swimming events, swimming courses, swimming lessons interfaces to enhance user engagement and usability.

Authentication Forms: The authentication forms, including login and registration, are built using ReactJS and handle user input through controlled components. Form data is managed using state hooks and is submitted via secure API calls to the backend for authentication and user account creation.

Registration Code:

```
const Register = () => {
  const [swimmerData, setSwimmerData] = useState({
    name: "",
    email: "",
    number: "",
    category: "",
    gender: "",
    dob: "",
    registrationNumber: "",
    password: "",
    confirmPassword: "",
    courses: []
  });

  const navigate = useNavigate();

  const handleInputChange = (event) => {
    const { name, value } = event.target;
    setSwimmerData({ ...swimmerData, [name]: value });
  };
```

```
const handleCheckboxChange = (event) => {
  const { value, checked } = event.target;
  let updatedCourses = [...swimmerData.courses];
  if (checked) {
    updatedCourses.push(value);
  } else {
    updatedCourses = updatedCourses.filter((course) => course !== value);
  }
  setSwimmerData({ ...swimmerData, courses: updatedCourses });
};

const isValidEmail = (email) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email);
};
```

```
const signup = async () => {
  const { password, confirmPassword, email } = swimmerData;

  if (password.length < 8) {
    alert("Password must be at least 8 characters long.");
    return;
  }

  if (password !== confirmPassword) {
    alert("Passwords do not match.");
    return;
  }

  if (!isValidEmail(email)) {
    alert("Please enter a valid email address.");
    return;
  }

  try {
    console.log('Sending swimmer data:', swimmerData); // Log data being sent

    const response = await fetch('http://localhost:4003/register', {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(swimmerData),
    });
```

```
return (
  <div className="signup">
    <div className="signup-container">
      <h1>Swimmer Registration Form</h1>
      <div className="signup-fields">
        <label htmlFor="swimmerName">Name</label>
        <input type="text" value={swimmerData.name} id="swimmerName" name="name" onChange={handleInputChange}
        required />
        <label htmlFor="swimmerEmail">Email</label>
        <input type="email" value={swimmerData.email} id="swimmerEmail" name="email" onChange={handleInputChange}
        required />
        <label htmlFor="swimmerNumber">Number</label>
        <input type="text" value={swimmerData.number} id="swimmerNumber" name="number" onChange=
        {handleInputChange} required />
        <label htmlFor="swimmerCategory">Select a Swimmer Category:</label>
        <select id="swimmerCategory" name="category" value={swimmerData.category} onChange={handleInputChange}
        required>
          <option className='option' value="">Select Category</option>
          <option className='option' value="Dhaka University Student">Dhaka University Student</option>
          <option className='option' value="DU Teacher/Officer/Employee/Their Family Member">DU Teacher/Officer/
          Employee/Their Family Member</option>
          <option className='option' value="BUET/DMC Student">BUET/DMC Student</option>
          <option className='option' value="Associate Member">Associate Member</option>
          <option className='option' value="General Swimmer">General Swimmer</option>
        </select>
        <label htmlFor="swimmerDOB">Date of Birth</label>
        <input type="date" id="swimmerDOB" name="dob" value={swimmerData.dob} onChange={handleInputChange}
        required/>
        <label htmlFor="swimmerRegistrationNumber">Registration Number for DU Students</label>
        <input type="text" id="swimmerRegistrationNumber" name="registrationNumber" value={swimmerData
```

Figure: Registration Processes code

Sign In Code:

```
const SignIn = () => {
  const [formData, setFormData] = useState({
    email: '',
    password: '',
    showPassword: false,
  });

  const navigate = useNavigate();
  const { login } = useAuth();

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const togglePasswordVisibility = () => {
    setFormData({ ...formData, showPassword: !formData.showPassword });
  };
```

```
const handleLogin = async () => {
  console.log('Login function executed', formData);

  let responseData;
  await fetch('http://localhost:4003/login', {
    method: 'POST',
    headers: {
      Accept: 'application/form-data',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ email: formData.email, password: formData.password }),
  })
    .then((response) => response.json())
    .then((data) => (responseData = data));

  if (responseData.success) {
    localStorage.setItem('auth-token', responseData.token);
    login();
    navigate('/dashboard');
  } else {
    alert(responseData.errors);
  }
};
```

```
  return (
    <div className="login">
      <div className="login-container">
        <h1>Sign in to Dhaka University Swimming Pool</h1>
        <div className="login-fields">
          <label htmlFor="email">Email</label>
          <input
            type="email"
            value={formData.email}
            id="email"
            name="email"
            onChange={handleChange}
            required
          />
          <label htmlFor="password">
            Password
            <div className="password-input">
              <input
                type={formData.showPassword ? 'text' : 'password'}
                value={formData.password}
                id="password"
                name="password"
                onChange={handleChange}
                required
              />
              <button
                type="button"
                className="password-toggle"
                onClick={togglePasswordVisibility}
              >
                {formData.showPassword ? <FaEye /> : <FaEyeSlash />}
```

Figure: SignIn Processes code

Schedules: Schedule display available time slots for swimming classes, events, and maintenance. Users can view and register for these slots and also can download as PDF.

Code and interface:

```
const Schedule = () => {
  const scheduleRef = useRef();

  const downloadAsPDF = () => {
    const input = scheduleRef.current;
    html2canvas(input, { scale: 2 }) // Increase scale for better quality
      .then(canvas => {
        const imgData = canvas.toDataURL('image/png');
        const pdf = new jsPDF('p', 'mm', 'a4'); // Create PDF in A4 size
        const imgProps = pdf.getImageProperties(imgData);
        const pdfWidth = pdf.internal.pageSize.getWidth();
        const pdfHeight = (imgProps.height * pdfWidth) / imgProps.width;
        pdf.addImage(imgData, 'PNG', 0, 0, pdfWidth, pdfHeight);
        pdf.save('DU-Swimming-Pool-Schedule.pdf');
      })
      .catch(err => console.error('Error generating PDF:', err));
  };

  const downloadAsPNG = () => {
    const input = scheduleRef.current;
    html2canvas(input, { scale: 2 }) // Increase scale for better quality
      .then(canvas => {
        const link = document.createElement('a');
        link.href = canvas.toDataURL('image/png');
        link.download = 'DU-Swimming-Pool-Schedule.png';
        link.click();
      })
      .catch(err => console.error('Error generating PNG:', err));
  };
```

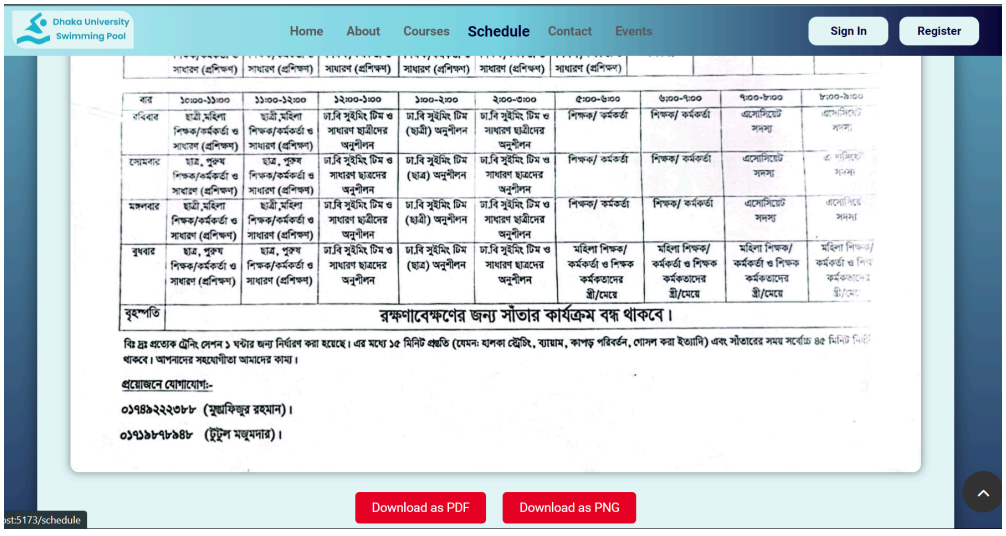


Figure: Schedule code and Interface

Transaction Interfaces: Secure transaction interfaces are provided for users to make online payments for various courses. These interfaces are integrated with payment gateways, ensuring secure and seamless financial transactions. Users can view transaction history and download receipts.

Code and Interface:

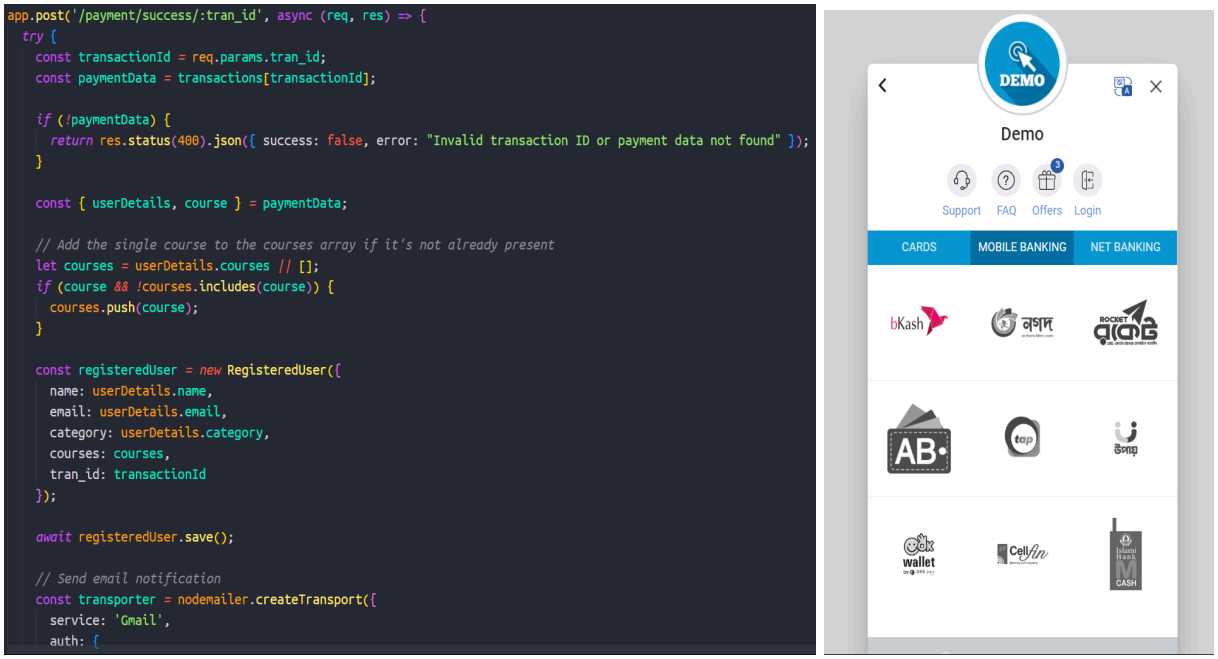


Figure: Transaction code and Interface

Courses: The courses module enables the creation and management of swimming courses offered at the pool. Users can browse available courses, view details such as schedules and fees, and register online.

Code and Interface:

```jsx
import React from 'react';
import "./CourseCard.css";
import Button1 from "../buttons/Button1";

const CourseCard = ({ className, imageSrc, title, description, buttonText, buttonLink }) => {

  return (
    <div className="courses">
      <div className={`section-1 ${className}`}>
        <div className="left-box-courses">
          <h1>{title}</h1>
          <p>{description}</p>
          <Button1 text={buttonText} to={buttonLink} className="course-btn" />
        </div>
        <div className="right-box-courses">
          <img
            src={imageSrc}
            alt="Course Image"
          />
        </div>
      </div>
    </div>
  );
};

export default CourseCard;
```

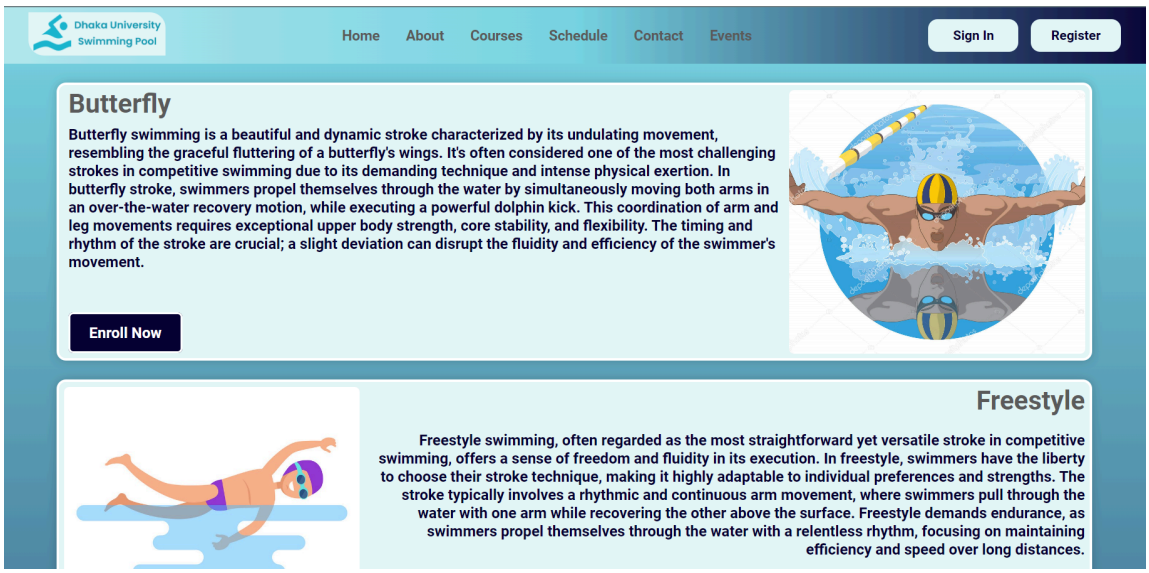


Figure: Course modules code and Interface

Swimming Events: The swimming events module allows administrators to create and manage events. Users can view upcoming events and take event tickets. The interface is designed to handle event details, participant lists for admin UI, and ticket PDF download.

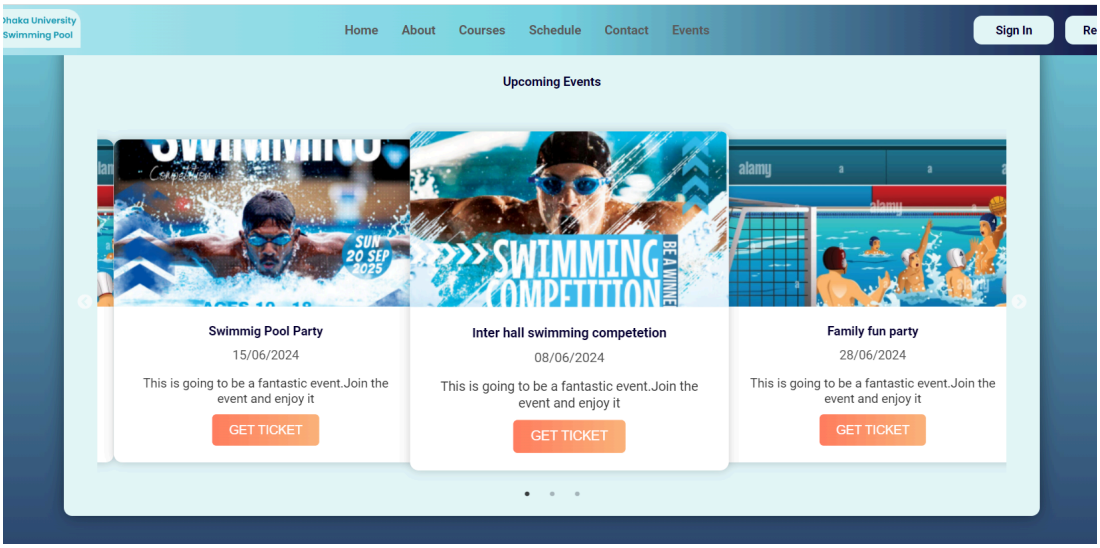


Figure: Event module UI

Swimming Lessons Interfaces: A comprehensive interface for swimming lessons includes video tutorials and guidelines to enhance the learning experience.

Code and Interface:

```
const SwimmingLessons = () => {
  return (
    <div className="swimming-lessons">
      <div className="swimming-lessons-container">
        <h1>Swimming Lessons Resources</h1>

        <div className="resources">
          <div className="resources-section">
            <h2>PDFs</h2>
            <ul>
              <li><a href="/path/to/pdf1.pdf" target="_blank" rel="noopener noreferrer">Swimming Basics</a></li>
              <li><a href="/path/to/pdf2.pdf" target="_blank" rel="noopener noreferrer">Advanced Techniques</a></li>
              <li><a href="/path/to/pdf3.pdf" target="_blank" rel="noopener noreferrer">Safety Guidelines</a></li>
            </ul>
          </div>

          <div className="resources-section">
            <h2>Photos</h2>
            <div className="photos">
              <img src="/path/to/photo1.jpg" alt="Swimming Lesson 1" />
              <img src="/path/to/photo2.jpg" alt="Swimming Lesson 2" />
              <img src="/path/to/photo3.jpg" alt="Swimming Lesson 3" />
            </div>
          </div>

          <div className="resources-section">
            <h2>Videos</h2>
            <div className="videos">
              <video controls>
                <source src="https://www.youtube.com/watch?v=p6ROh-M7S0k&ab_channel=GlobalTriathlonNetwork"
                type="video/mp4" />
```

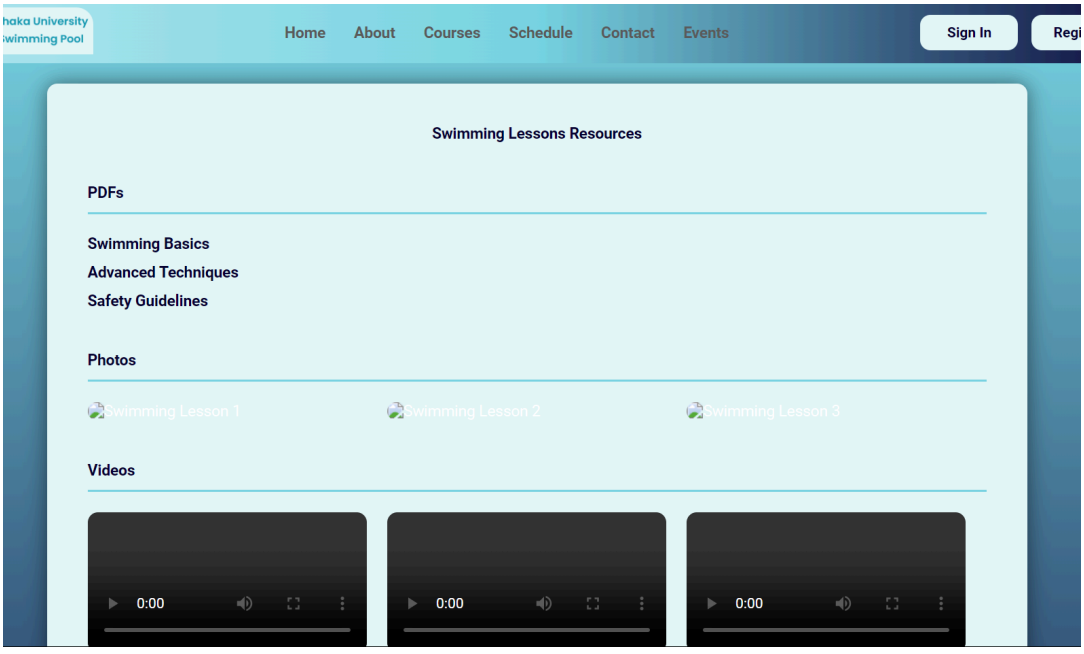


Figure: Lesson modules code and Interface

4.1.4. Testing

Extensive testing is and will be conducted to ensure the functionality, performance, and security of the application. This includes unit tests, integration tests, and user acceptance testing.

4.2. Backend

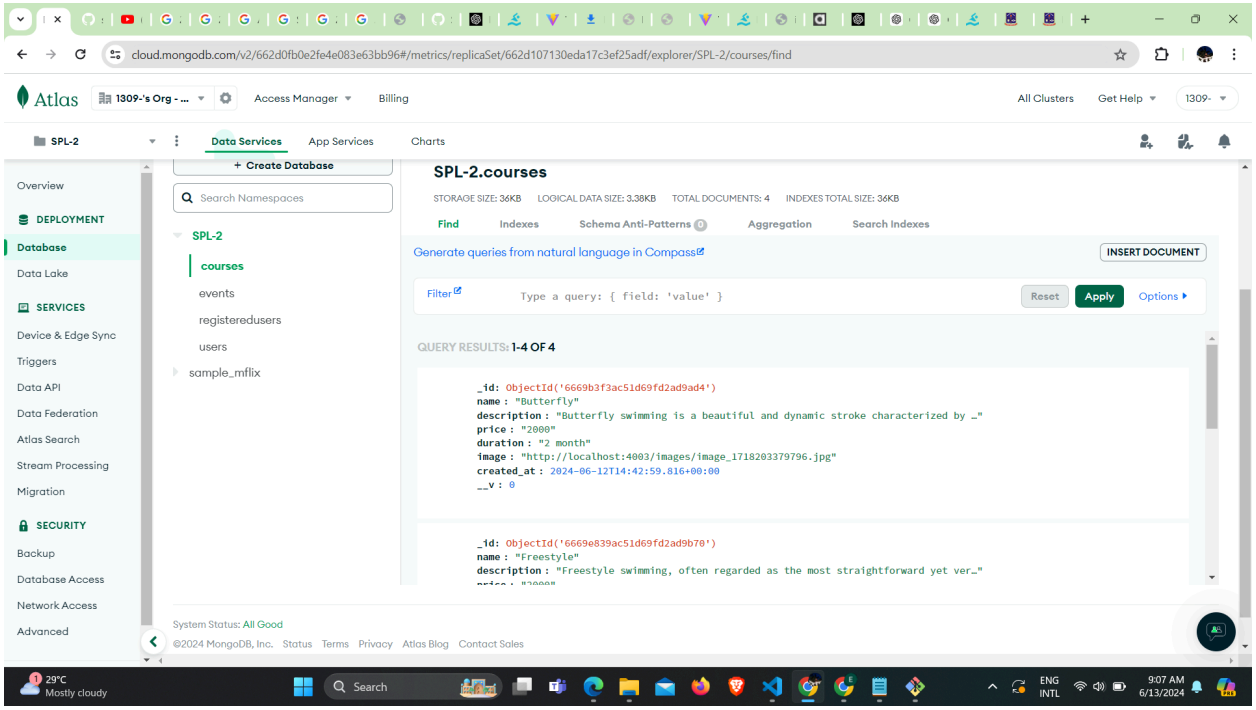
The backend is developed using NodeJS, with ExpressJS as the framework for building the API. It will handle user authentication, data management, and business logic.

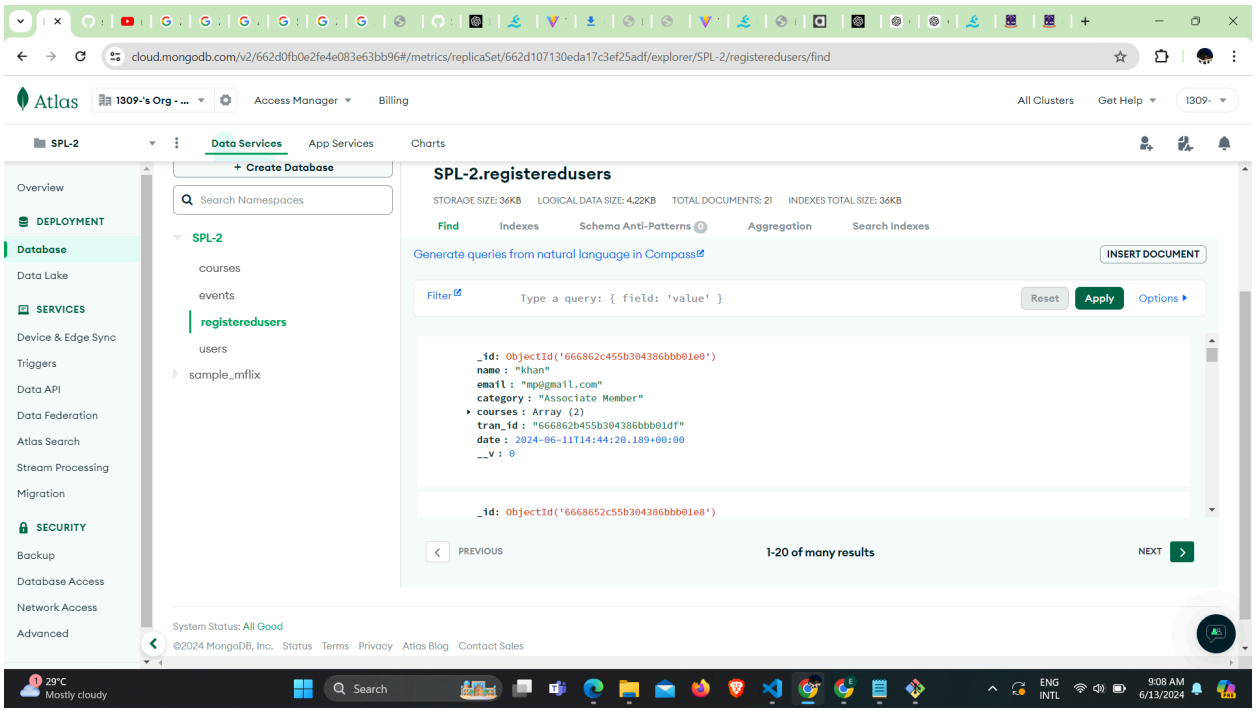
Figure: Backend dependencies

4.3. Database

4.3.1. Database Design

The database is designed to store user information, schedules, transactions, and other relevant data securely and efficiently.





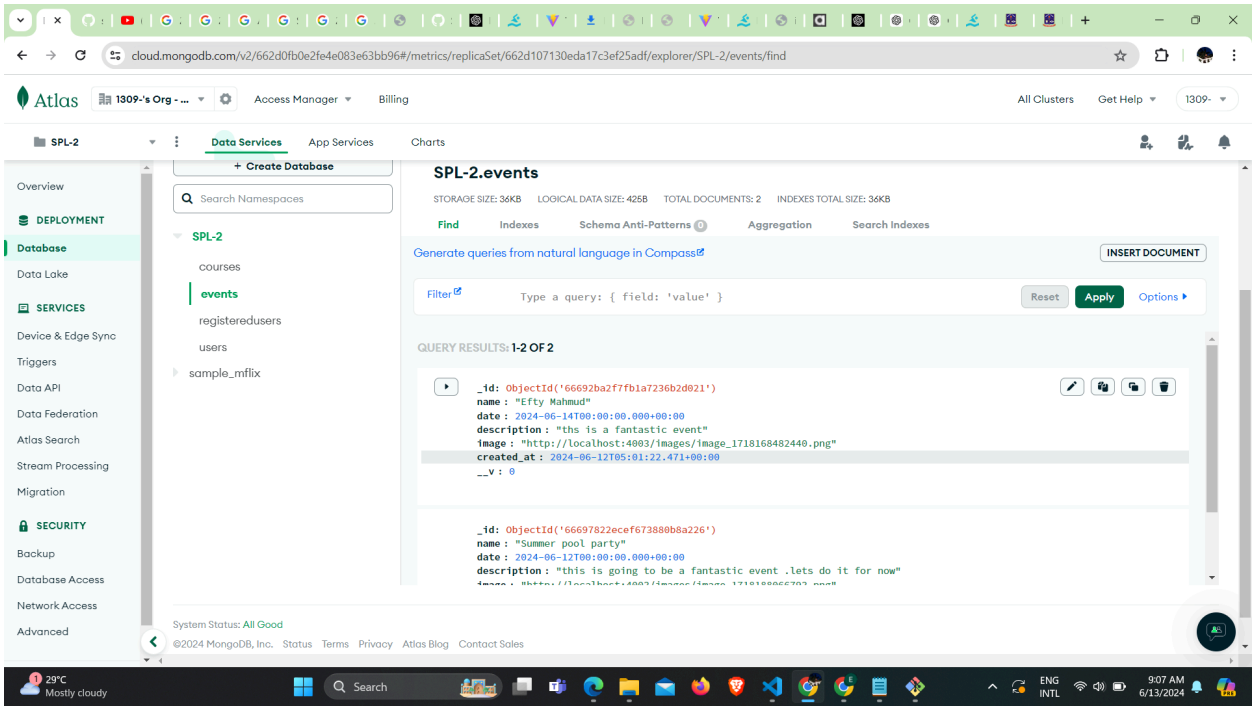


Figure: MongoDB Database implementation

4.3.2. Data Management

Data management practices is implemented to ensure data integrity, security, and availability. This includes regular backups and data validation procedures.

5. User Manual

5.1. Landing Page

The landing page will provide an overview of the system and guide users to the sign-in and registration pages.

Figure: Landing page

5.2. Sign In

Users can sign in using their credentials to access their personalized dashboard.

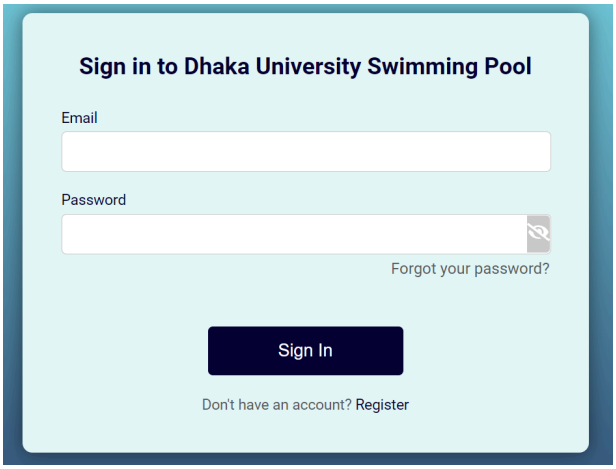


Figure: SignIn page

5.3. User Registration

New users can register for an account by providing necessary details and selecting a membership plan.

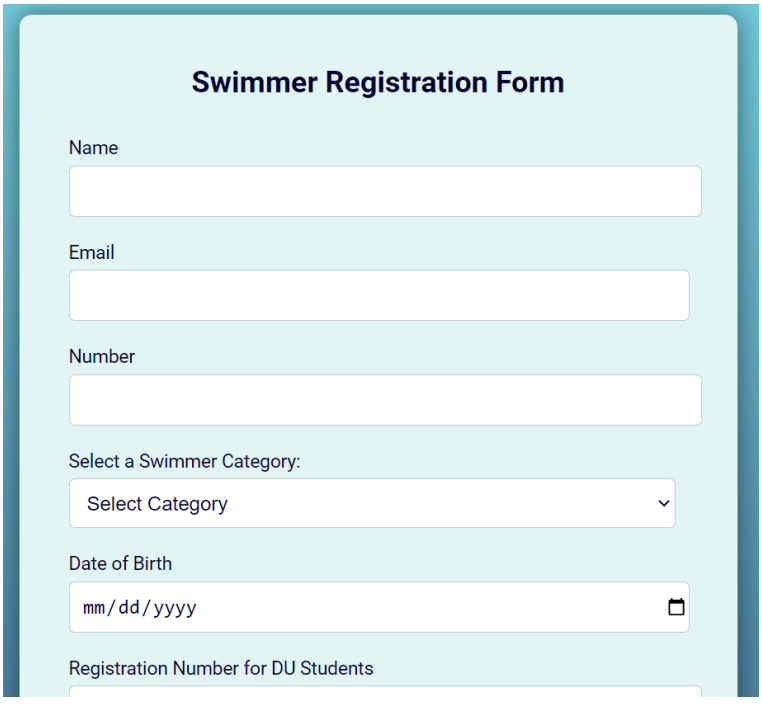


Figure: Registration Page

5.4. Schedule Management

Users can view the and download schedule of activities and register for courses at their convenience.

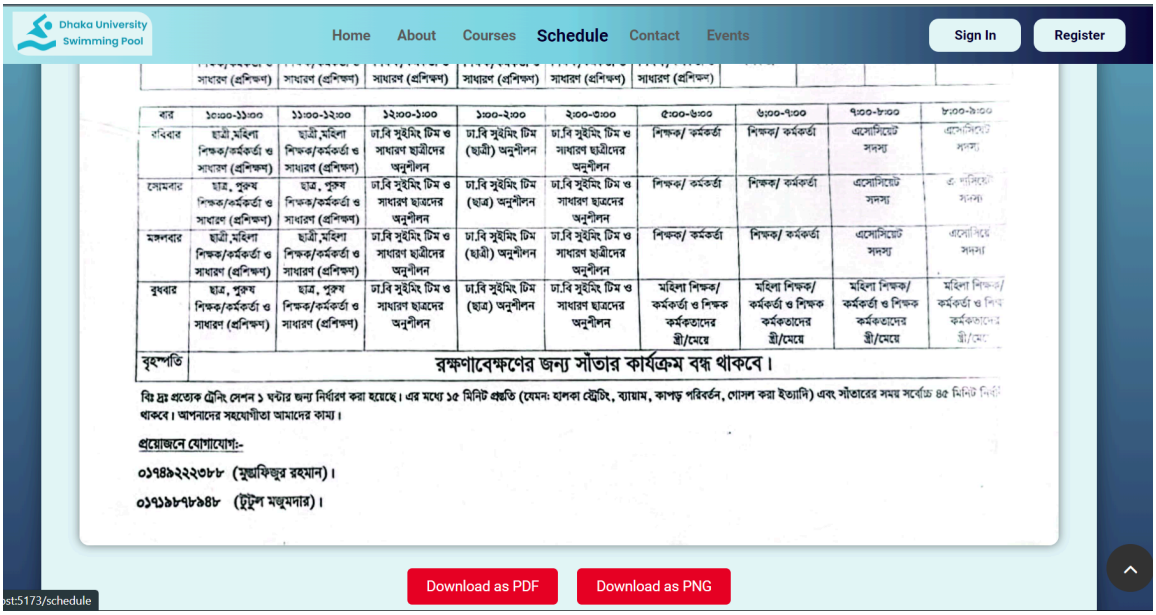
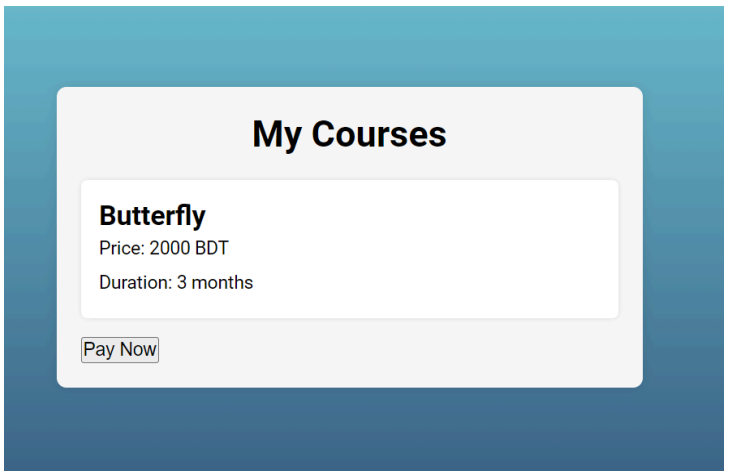
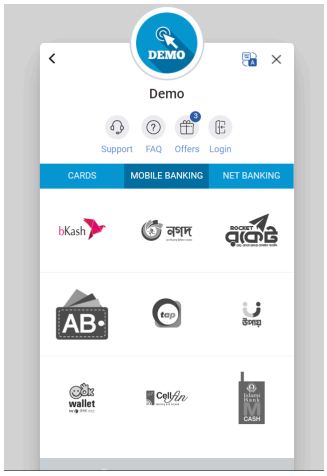


Figure: SchedulePage

5.5. Financial Management

Users can make payments for services that is enrollment of courses and view their transaction history. Administrators can access financial reports through admin panel.





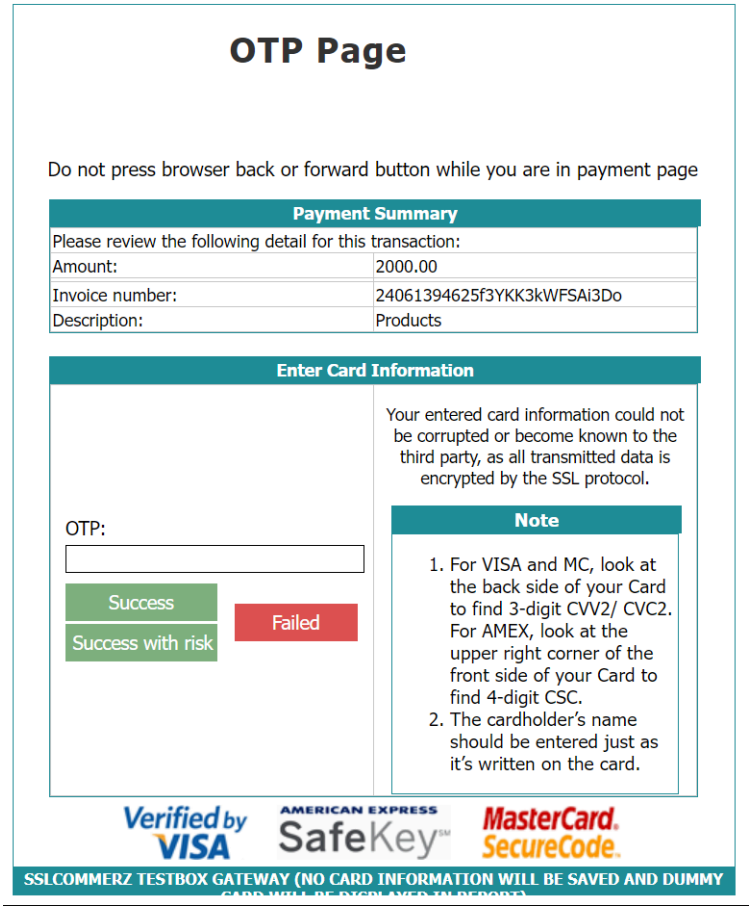


Figure: Financial management through SSLCOMMERZ

5.6. Event Management

Users can participate in swimming events which are created by Admin. Users can download ticket as PDF.

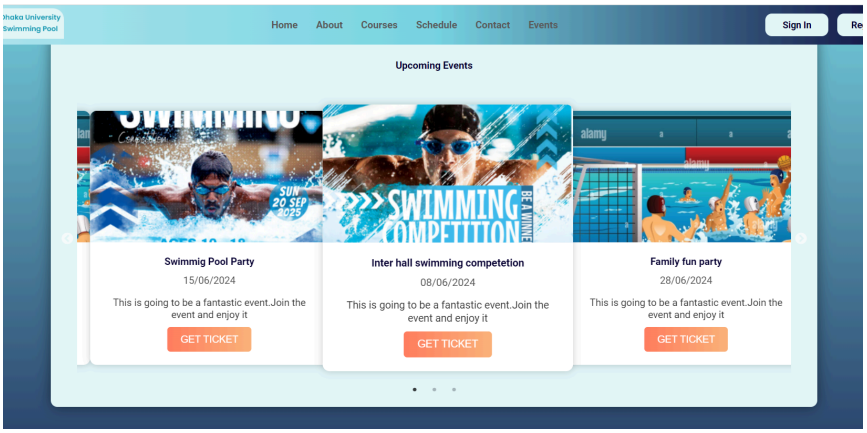


Figure: Event Module

5.6. Admin Panel

Admin panel is for to see registered users and their info, Admin can add different swimming events, can add swimming courses and can also check available courses. The UI's are given below:

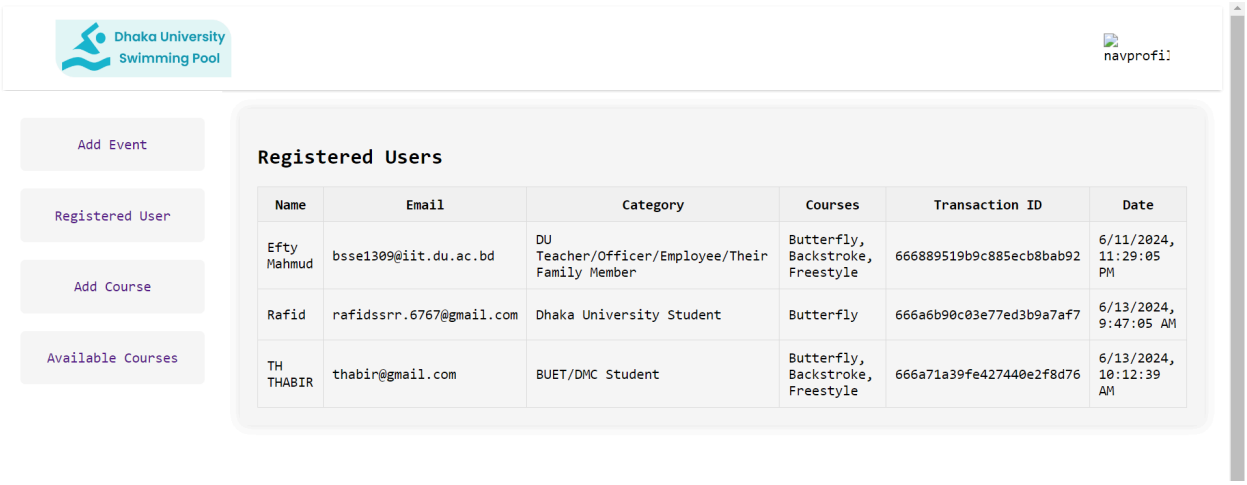


Figure: Admin can see registered users and their info

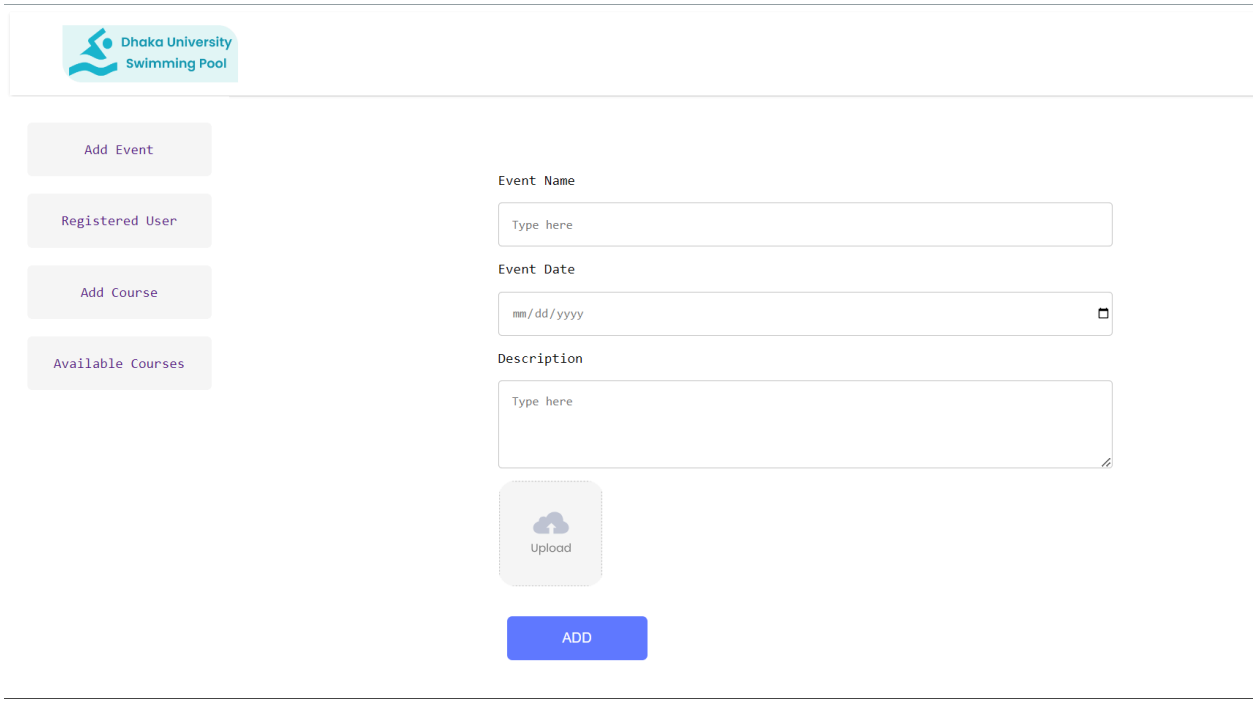


Figure: Admin can upload event

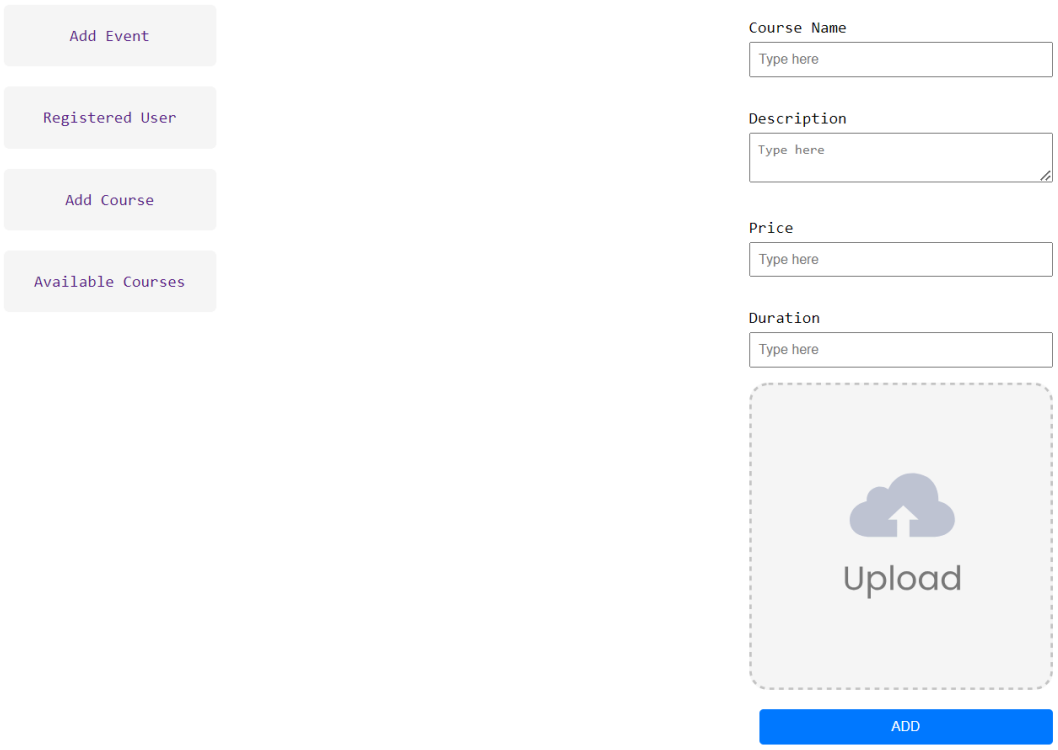


Figure: Admin can add swimming courses

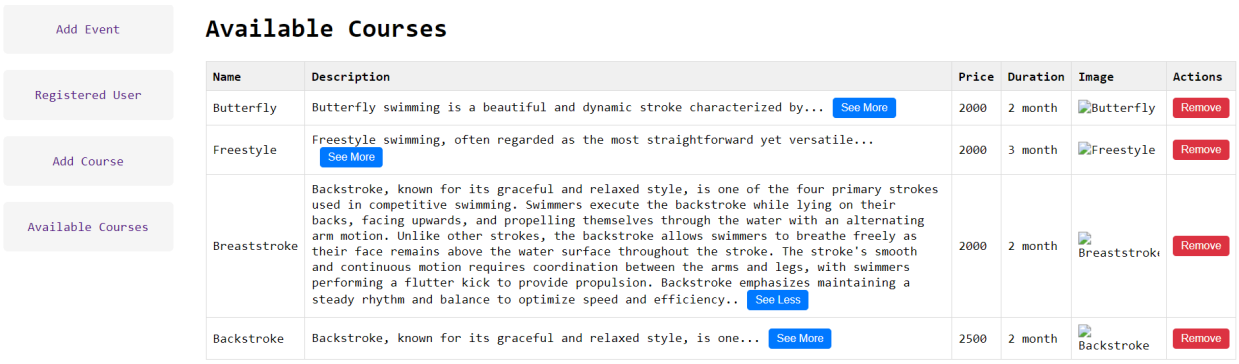


Figure: Admin can see current available courses

6. Challenges Faced

6.1. Dependency Related Issues

Managing dependencies and ensuring compatibility between different libraries and frameworks posed a challenge during development.

6.2. Integration of Components

Integrating various components and ensuring smooth communication between the frontend and backend required careful planning and testing.

6.3. Maintainability in Backend Code

Ensuring the maintainability of the backend code involved implementing best practices for code organization, documentation, and testing.

7. Conclusion and Future Work

The Dhaka University Swimming Pool Management System represents a significant step towards modernizing and optimizing the management of swimming pool facilities at Dhaka University. By implementing this system, we aim to enhance user experience, improve operational efficiency, and contribute to a safer and more enjoyable swimming experience for all members of the university community.

8. References

1. *W3schools react tutorial*. Retrieved from

   https://www.w3schools.com/REACT/DEFAULT.ASP

2. *SSLCOMMERZ intregation*. Retrieved from

   https://github.com/sslcommerz/SSLCommerz-NodeJS