GROUP-6

Conference Management System

Drisya Damodharan- 22250267
MSc in Computer Science
(Software Engineering)
Department of Computer Science
Maynooth University

Suramya Suresh Babu- 22251388
MSc in Computer Science
(Software Engineering)
Department of Computer Science
Maynooth University

Tianze An- 22250240
MSc in Computer Science
(Software Engineering)
Department of Computer Science
Maynooth University

***Abstract*— This report represents an overview of the Conference Management System, a web application that manages and tracks various papers submitted for review. The reviewer can sign up and log in as necessary. This website can be easily managed by keeping track of the papers that are assigned for review, reading those papers, and only look at the entries provided by other reviewers after their reviews have been submitted. The papers are automatically assigned, giving the reviewer some degree of system control. They only need to complete the reviewer form, save it as a draft for later revision, and then submit the review to give their feedback on the submitted paper. The reviewer is unable to make any more changes after submitting. In order to determine whether the score and comments provided are consistent with those of other reviewers, they can later check the review provided by other reviewers for the same paper only after submitting their review. MERN stack development was used to implement the website.**

Keywords*— *Conference Management System, Web Application, Reviewer, MERN stack

I. INTRODUCTION

Organizing and managing conferences can be a daunting task for many individuals. With the increasing number of academic conferences held each year, managing the submissions and review process can become even more tedious. In recent years, conference management systems have emerged as an essential tool for facilitating this process. The Conference Management System described in this report provides a user-friendly platform that enables reviewers from around the world to review papers and view feedback from other reviewers. The system offers numerous benefits, including streamlined communication, efficient data storage, and automation of complex and time-consuming tasks. Overall, the Conference Management System is an indispensable website for simplifying and optimizing the conference organization and management process.

II. PROBLEM STATEMENT

The problem statement is divided into the following user stories:

1) A reviewer of the conference logs into their online account and can view the list of papers assigned to them to review. For each paper they see: paper title, authors, keywords, abstract, pdf attachment, and name of author that submitted the paper.

2) The reviewer has the option to enter a review for a paper. Details to enter are: overall score (2 accept to -2 reject), review details, private comments to other reviewers and conference organizers. The reviewer can save their review in draft format for future editing. When the reviewer clicks submit on their review, they cannot make further edits to the review.

3) When a reviewer submits their review for a paper, they can see the other reviews already submitted for the paper. Their review is also visible to other reviewers who have already submitted a review; but hidden from those who have not yet submitted their review.

III. METHODOLOGY

The MERN stack, which includes MongoDB as the database, Express as the web application framework, React as the front-end library, and Node.js as the back-end runtime environment, is the structure around which our conference management system is developed. We have developed the system as a collection of pages, each of which is in charge of managing a particular aspect of conference system from the perspective of reviewers, such as reviewer registration, login, paper dashboard, which contains papers assigned for review, and review form, where reviewers can offer overall score, review details and private comments (visible only to other reviewers and organizers) on the papers assigned to them. To gain a complete picture of their assessment, they can examine other evaluations that other reviewers provided for the same paper only after their own review has been submitted. The RESTful API is used to execute the API endpoints, which helps the frontend in interacting with the database. Through this RESTful API, each of the front-end pages communicates with the database, allowing for the flexible and scalable integration of many services. React and Redux are implemented to create a responsive user interface that enables users to engage with the system from any device with a web browser.

IV. ARCHITECTURE

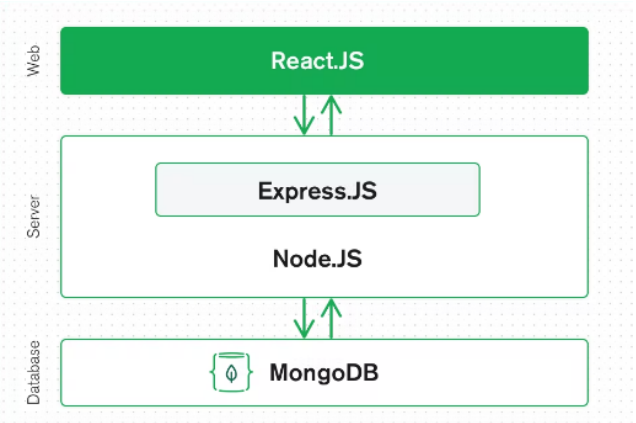

Fig-1: Architecture of MERN Stack Development

Full-stack web applications can be deployed more quickly and easily with the MERN Stack, a JavaScript stack. The MERN Stack is made up of the following 4 technologies: MongoDB, Express, React, and Node.js. It is intended to simplify and streamline the development process. Each of these four strong technologies plays a significant role in the development of web applications and offers developers an end-to-end framework to work on. The Architecture of MERN Stack Development is as shown in Fig-1.

A. MongoDB- Cross-platform Document-Oriented Database

MongoDB is a NoSQL database where each record is a document made up of key-value pairs that resemble JSON (JavaScript Object Notation) objects. Due to MongoDB's flexibility, users can create schemas, databases, tables, etc. very easily. Users can use Mongo Shell after installing MongoDB. Users can communicate and perform operations such as querying, updating records, and deleting records through the JavaScript interface provided by the Mongo Shell.

B. Express- Back-end Framework

Express is a Node.js framework. Express makes it simpler and easier to develop the back-end code than using Node.js and developing multiple Node modules. Great web apps and APIs can be created with the aid of Express. It offers a wide range of middlewares, which results in quicker and simpler code.

C. React – Front-end Library

React is a JavaScript library that is used for building user interfaces. Since React can manage rapidly changing data, it is used to create single-page applications and mobile applications. React allows users to code in JavaScript and create UI components.

D. Node.js - JS Runtime Environment

The JavaScript Environment offered by Node.js enables users to run their code on the server (outside of the browser). The user can select from thousands of free packages (also known as node modules) to download using the node pack manager, or npm.

V. SYSTEM REQUIREMENTS

1) Operating System: Windows 7 and above
2) Language: HTML, CSS, JavaScript
3) Technology stack: MongoDB, ExpressJS, ReactJS, NodeJS
4) IDE: Visual Studio Code
5) API Platform: Postman
6) Browser: Any browser, IE 8 and above
7) Processor: A single-core 2GHz processor
8) RAM: 512 Mb and above

VI. TECHNICAL APPROACH

The Conference Management System is made up of a number of modular components, each of which is in charge of various aspects of organizing conferences. We decided to use the MERN stack because it offers an advanced and adaptable framework for developing web-based applications. We have put in place a variety of security features, like user authentication and authorization, to ensure the security and privacy of our users. To safeguard user credentials, we also implemented industry-standard security procedures like password hashing.

The front-end is developed in React, which offers a component-based design that enables the creation of reusable and responsive user interface elements. To handle state management, routing, and styling, we used a range of React tools and frameworks, including Redux, React-Toastify, React-icons, and React Router.

The back-end is developed using Node.js and Express, two web application frameworks that are quick and scalable. We utilized MongoDB, a NoSQL document-oriented database that enables adaptable and scalable data storage, as our database. Using Express, we created a RESTful API that makes it simple to combine various services and parts. To manage data modeling and validation, we also used Mongoose, an Object-Document Mapping (ODM) package that offers a higher-level abstraction layer over MongoDB. The MongoDB cloud is used to manage the database entirely. The objects in the application code map to the JSON-like document data model.

A. Registration

For registration, the useState hook is utilized, and the Register component defines a state variable called formData. Name, email, password, and password confirmation have been included. Additionally, it makes use of React hooks like useSelector to retrieve information from the Redux store, useDispatch to send actions to the store, and useNavigate to move to the next page, which is the Login page.

B. Login

We have used JWT (JSON Web Tokens) and bcrypt to securely encrypt and save user credentials in the database for user authentication. The encrypted tokens are generated here. Node.js and Express are utilized for developing the server-side component, while React, React Router, and Redux can be used to implement the client-side component.

C. Papers Dashboard

The server-side code is responsible for handling incoming requests and returning appropriate responses. It has two route handlers that are in charge of retrieving assigned papers from the database: getPapers and getPaperById. These route handlers are exported as an object, which other application components use. The dashboard page that displays the user's assigned papers is displayed by the client-side code, which is a React component. When the component mounts, it utilizes the useEffect hook to send an action (getPapers) that retrieves the papers from the database. The papers are extracted from the Redux store and rendered in a table using the useSelector hook.

D. Review Form

The component displays a form with a number of input fields, such as a slider input for the final score, a text input for the review's details, and a text input for the private comments. Two buttons, one to save the review as a draft and the other to submit it as a final submission, are also rendered by the component. The component sends an action to the Redux store to save the review data as a draft when the user clicks the "Save as Draft" button. The component sends the appropriate action to the Redux store to submit the review data as a final submission when the user clicks the "Submit" button. Depending on whether the user is present, the component employs a variety of conditional rendering techniques to show and hide specific items. The component uses various conditional rendering techniques to show and hide certain elements based on whether the user has already submitted a review, whether the form data is valid, and whether the review is a draft or a final submission.

E. Reviews Dashboard

To access the navigation object and URL parameters, the component first calls the useNavigate and useParams hooks. Additionally, it calls the useSelector hook to retrieve the data from the Redux store's state and the useDispatch hook to obtain a reference to the dispatch function. When the component mounts, the useEffect hook is then used to dispatch an action creator called getReviewsbyId to retrieve review data using the id URL parameter. When review data is available, it is shown as a table. Additionally, it displays a message stating that there are no reviews if the reviews array is empty or an error message if a retrieval problem occurred.

VII. CONCLUSION

The Conference Management System is a full-featured web application developed to make managing and following up on submitted papers easier. The platform makes it simple for reviewers to register, log in, and carry out a number of actions, like reading the papers that have been assigned for review, submitting the reviews, saving them as drafts, or submitting the review and looking at the comments left by other reviewers for the same paper. The website, which was created utilizing the MERN stack, has a user-friendly interface and effective review management features. Overall, the Conference Management System is a useful tool for organizing and monitoring the papers that have been requested for review.

ACKNOWLEDGMENT

We are extremely thankful to our Professor Liadh Kelly and Head Demonstrator Eduardo Avila for their constant support and guidance in the completion of this project and the documentation.

REFERENCES

[1] https://www.geeksforgeeks.org/mern-stack/
[2] https://react-redux.js.org/introduction/getting-started
[3] https://www.mongodb.com/mern-stack
[4] https://www.w3schools.com/REACT/default.asp
[5] https://www.w3schools.com/nodejs/nodejs_intro.asp
[6] https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js
[7] https://www.freecodecamp.org/news/how-to-sign-and-validate-json-web-tokens/
[8] Murali Kanthi, Darmapuri Pranay, Mohammad Shanawaz, Sode Saideep, “Project Tracking System using MERN Stack”, JETIR Journal, vol. 8, Issue 6