UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Web-based Recommender System for Scientific Conferences Management

Diploma Thesis

Spyridon Kavvathas

Supervisor: Michael Vassilakopoulos

June 2024

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Web-based Recommender System for Scientific Conferences Management

Diploma Thesis

Spyridon Kavvathas

Supervisor: Michael Vassilakopoulos

June 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Βασισμένο στον Ιστό Σύστημα Συστάσεων για τη Διαχείριση Επιστημονικών Συνεδρίων

Διπλωματική Εργασία

Σπυρίδων Καββαθάς

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Ιούνιος 2024

Approved by the Examination Committee:

Supervisor **Michael Vassilakopoulos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Aspasia Daskalopoulou**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Eleni Tousidou**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

First and foremost, Professor Michael Vassilakopoulos's mentorship has been a true privilege. His constant encouragement, support, and insightful feedback were essential in bringing this application to life and guiding me through the writing of this thesis.

Additionally, I am very thankful to Vaios Stergiopoulos for his helpful suggestions and contributions to both the development of this thesis and the recommendation system used in it. His outstanding help was crucial, and without his support, completing this thesis would have been very difficult.

I would like to extend my sincere appreciation to my family for their unwavering support throughout this endeavor. Their motivation and commitment have played a key role in this journey.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Spyridon Kavvathas

Diploma Thesis

Web-based Recommender System for Scientific Conferences Management

Spyridon Kavvathas

Abstract

In the world of organizing scientific conferences, there's a noticeable gap in user-friendly web tools that not only help manage submissions but also suggest suitable papers to reviewers. This thesis seeks to fill this gap by investigating the Conference Paper Assignment Problem (CPAP) in academia, examining current management software, and suggesting a complete solution. This solution will integrate an existing recommendation system with a new, user-friendly web platform for managing scientific conferences. Our aim was to develop a web-based platform specifically designed for efficiently managing scientific conferences. Users are empowered to create accounts, initiate new conferences, submit papers, and enlist reviewers seamlessly within the system.

Building this system was a complex task, starting from scratch. The decision to employ the technologies below was guided by their widespread adoption and proven success in similar web development projects. React.js stands out for its component-based architecture, facilitating modular design and efficient rendering of user interfaces. Node.js, paired with Express.js, offers a scalable and lightweight server-side solution, ideal for handling concurrent requests in real-time applications. Furthermore, SQL and MySQL provide robust data storage and retrieval capabilities, crucial for managing large volumes of conference-related information securely and efficiently. Using these well-known technologies, our system is designed to meet what's commonly expected in the industry and provide an easy, smooth experience for users.It's worth noting that our web system integrates a pre-existing paper recommendation system, streamlining the process of suggesting papers to reviewers. This existing system was seamlessly incorporated into our web platform, which was developed using different technologies.

Keywords:

Conference management system, Recommender system, Web applications, JavaScript based

technologies for web applications, Databases / SQL

Διπλωματική Εργασία

Βασισμένο στον Ιστό Σύστημα Συστάσεων για τη Διαχείριση Επιστημονικών Συνεδρίων

Σπυρίδων Καββαθάς

Περίληψη

Στον κόσμο της οργάνωσης επιστημονικών συνεδρίων, υπάρχει ένα εμφανή κενό σε φιλικά προς τον χρήστη εργαλεία που όχι μόνο θα βοηθούν στη διαχείριση των υποβολών νέων συνεδρίων αλλά και θα προτείνουν κατάλληλα άρθρα στους κριτές κάθε συνεδρίου. Αυτή η διπλωματική επιδιώκει να καλύψει αυτό το κενό, ερευνώντας το πρόβλημα ανάθεσης εργασιών σε επιστημονικά συνέδρια στον ακαδημαϊκό τομέα, εξετάζοντας τρέχοντα λογισμικά διαχείρισης συνεδρίων και προτείνοντας μια ολοκληρωμένη λύση. Αυτή η λύση θα ενσωματώσει ένα υπάρχον σύστημα προτάσεων με ένα νέο, φιλικό προς τον χρήστη σύστημα για τη διαχείριση επιστημονικών συνεδρίων. Ο στόχος μας είναι να δημιουργήσουμε ένα βασισμένο στον ιστό σύστημα ειδικά σχεδιασμένο για τη διοργάνωση επιστημονικών συνεδρίων.

Η κατασκευή αυτού του συστήματος ήταν μια πολύπλοκη εργασία, ξεκινώντας από το μηδέν. Η απόφαση να χρησιμοποιηθούν οι τεχνολογίες που αναφέρονται παρακάτω οφείλεται στην ευρεία αποδοχή και στην αποδεδειγμένη επιτυχία τους σε παρόμοια έργα ανάπτυξης ιστού. Πιο συγκεκριμένα οι τεχνολογίες και οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι: React,js, Node.js, Express.js, SQL, MySQL. Χρησιμοποιώντας αυτές τις τεχνολογίες, το σύστημά μας σχεδιάστηκε προκειμένου να προσφέρει μια εύκολη και ομαλή εμπειρία για τους χρήστες. Αξίζει να σημειωθεί ότι το βασισμένο στον ιστό σύστημα μας ενσωματώνει ένα υπάρχον σύστημα προτάσεων εργασιών, διευκολύνοντας τη διαδικασία προτάσεων άρθρων στους κριτές του εκαστοτε συνεδρίου.

Λέξεις-κλειδιά:

Σύστημα διαχείρισης συνεδρίων, Σύστημα συστάσεων, Εφαρμογές ιστού, Τεχνολογίες JavaScript για εφαρμογές ιστού, Βάσεις δεδομένων / SQL

Table of contents

Acknowledgements **ix**

Abstract **xii**

Περίληψη **xiv**

Table of contents **xv**

List of figures **xix**

List of tables **xxi**

Abbreviations **xxiii**

1 Introduction **1**

1.1 Subject of Thesis 1

1.1.1 Contribution 2

1.2 Thesis Structure 3

2 Related work **5**

2.1 COMS - Conference Management Software 6

2.2 EasyChair 6

2.3 Converia - Conference Management Software 6

2.4 Recommender Systems for the Conference Paper Assignment Problem . . . 7

2.5 The AI Conference Paper Assignment Problem 7

2.6 An Academic Recommender System on large citation data based on clustering, graph modeling and deep learning 8

3 Technologies used **9**

3.1 Type of the application . . . 9

3.1.1 Types of web-applications . . . 9

3.1.2 Selection of Type . . . 12

3.2 Front-end technologies . . . 12

3.2.1 JavaScript . . . 12

3.2.2 CSS . . . 13

3.2.3 HTML . . . 14

3.2.4 React.js . . . 14

3.2.5 Chakra UI . . . 15

3.3 Back-end Technologies . . . 15

3.3.1 Node.js . . . 16

3.3.2 Express.js . . . 16

3.3.3 SQL . . . 17

3.3.4 MySQL . . . 17

3.4 Authentication . . . 18

3.4.1 JSON Web Tokens . . . 18

4 Implementation **19**

4.1 System design . . . 19

4.2 Requirements Analysis . . . 20

4.3 Database Design . . . 21

4.4 Web server . . . 23

4.4.1 server.js file . . . 24

4.4.2 middleware folder . . . 25

4.4.3 db folder . . . 26

4.4.4 controllers folder . . . 27

4.4.5 routes folder . . . 32

4.5 Recommendation System for Scientific Papers . . . 35

4.5.1 Requirements . . . 36

4.5.2 main.py . . . 36

4.5.3 db_utils.py . . . 38

4.5.4 r2p_utils.py . . . 38

4.5.5 reviewer.py 39
4.5.6 data/alphabetical_researchers 39
4.6 Web Client 41
4.6.1 Folder Structure and Files 41
4.6.2 hooks folder 45
4.6.3 Components Folder 46
4.6.4 pages folder 47
4.6.5 context folder 48

5 Graphical User Interface **49**

6 Usage example **59**

7 Conclusions **65**
7.1 Summary and Conclusions 65
7.2 Future work 66

Bibliography **69**

APPENDICES **73**

A Code **75**

List of figures

4.1 The design of the system 19
4.2 The Entity Relationship Diagram Diagram 23
4.3 The server file 25
4.4 The folder that contains the code of the recommendation system 35
4.5 The structure of BERT model 37
4.6 The client folder 42
4.7 The pages folder 47

5.1 The first page of the app 51
5.2 The login page 51
5.3 The register page 52
5.4 The initial page of the app (home page), where basic information about it exists 52
5.5 The page where the user can select what functionality wants 53
5.6 The page where the user can view the personal conferences. The names of conferences here is some test names 53
5.7 The page where the user create a new conference 54
5.8 The page where the user can see the papers and the reviewers of his conference 54
5.9 The page where the user add a single paper, giving the title and the abstact . 55
5.10 The page where the user can upload his paper 55
5.11 The page where the user upload multiple papers with their title and abstract, via Excel 56
5.12 The page where users can upload multiple reviewers at once through an Excel file containing their names and emails. 56
5.13 The page where the user can upload reviewers with their name and email by hand one each time 57

5.14 The page where the user can upload multiple reviewers with their name and email via a CSV file 57

6.1 Conference form example 59
6.2 Example of creation of a conference 60
6.3 The conference before the upload of papers and reviewers 60
6.4 The upload of paper_test.csv file. 61
6.5 The upload of reviewers_test.csv file. 61
6.6 The assignment results from the recommendation system. 62
6.7 The user can download both the assignments and the paper IDs. 62
6.8 The recommendation system when starts to create the profiles. 63

List of tables

4.1 Summary of controllers in the controllers folder 27

Abbreviations

κ.λπ.	και λοιπά
CSS	Cascading Style Sheets
JS	JavaScript
SQL	Structured Query Language
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
JSX	Javascript Syntax Extension
CORS	Cross-Origin Resource Sharing
JWT	JSON Web Token
CPAP	Conference Paper Assignment Problem
PCO	Professional Conference Organizers
PWAs	Progressive Web Applications

Chapter 1

Introduction

The ongoing progress of recommendation systems mirrors the swift advancements seen in machine learning recently. These developments have not only sped up the adoption of recommendation systems across many industries but also highlighted their growing importance and flexibility. At the same time, academia has been vital in coming up with new methods to improve recommendation techniques.

As a result, industries are actively embracing these academic breakthroughs to refine their recommender systems for real-world use. Professionals in these fields are carefully applying the latest findings from academia to create top-notch products that meet customer needs.

This thesis focuses on integrating a web-based system (something accessible to everyday users) with the knowledge produced in academia, particularly in the Conference Paper Assignment Problem (CPAP). The goal is to develop a strong solution that caters to a wide range of users, including professors, students, and tech enthusiasts. This ensures its usefulness across different fields and for various purposes.

1.1 Subject of Thesis

This thesis endeavors to contribute a significant and unique perspective to the rapidly evolving field of recommendation systems research and industry applications. Building upon existing academic work in recommendation systems [1], this study aims to develop a web-based platform tailored for managing scientific conferences. The platform will use recommendation algorithms to facilitate the assignment of papers to reviewers, streamline the review process, and enhance overall conference management efficiency. By bridging the gap

between academic research and practical application, this thesis seeks to address real-world challenges and optimize the scholarly conference experience.

More specifically, this thesis aims to address the challenge of integrating web-based scientific conference creation with user-friendly management of conference data. The primary focus is on automating the assignment of papers to reviewers and providing users with convenient access to recommendation results through a digital interface.

It is important to be mentioned that the web-based recommendation system for scientific conferences is developed as a Single Page Application (SPA), leveraging the dynamic capabilities of React.js for a seamless user experience. Through its integration with a backend server powered by Node.js and Express.js, alongside a RESTful API for efficient data transfer, the system ensures real-time interaction and responsiveness. By using MySQL database for data persistence, the application guarantees reliability and scalability, meeting the evolving needs of users and conference organizers alike.

1.1.1 Contribution

The aim of this thesis is to add to the ongoing advancements in applications using recommendation systems, as previously stated. The findings presented here stem from a particular approach. The contribution of this thesis can be outlined as follows:

1. Existing conference web-based systems were thoroughly studied.

2. Appropriate web technologies were carefully selected.

3. Solutions that already exist for the Conference Paper Assignment Problem were examined [2] [3] .

4. The web system was meticulously designed and planned, with each step outlined. The recommendation system, which had been previously published [1], was studied and integrated into the web system.

5. The system was implemented step by step through coding.

6. The system was tested carefully with different combination of user input.

1.2 Thesis Structure

Following the introductory Chapter 1, the remainder of this thesis is organized as follows: Chapter 2 provides a complete review of related work from the international literature, accompanied by an in-depth analysis of various conference management software solutions. In Chapter 3, an analysis of the technologies employed in the web-based system is provided, aimed at users who may not be familiar with these technologies. Chapter 4 looks into the application's structure, conducting an analytical examination of its integration, the functions utilized, and the connections among various components and technologies. Chapter 5 showcases visual snapshots of the graphical interface within the web-based recommendation system for scientific conferences, offering readers a firsthand glimpse into its user experience. In Chapter 6, an example of the application's usage is provided. It illustrates how a user can create a conference, add papers and reviewers, and then retrieve the results. Finally, Chapter 7 synthesizes the findings and conclusions drawn from the study, while also highlighting promising directions for further research and exploration.

Chapter 2

Related work

The aim in this chapter is to explore how recommender systems have been implemented for the Conference Paper Assignment Problem, the insights gained from past endeavors, and the landscape of existing conference management web apps. This review is really valuable, because provides important lessons from past efforts, pinpoints challenges from previous work, and inspires new ideas.

To the best of our knowledge, here are the 3 web conference management applications that already exist, 2 papers related to the topic that the thesis is about and also a summary of the paper [1] that explains the functionality and the implemenation of the recommendation system that was used for the completion of the web system:

1. COMS - Conference Management Software [4]

2. EasyChair [5]

3. Converia - Conference Management Software [6]

4. Recommender Systems for the Conference Paper Assignment Problem [2]

5. The AI Conference Paper Assignment Problem [3]

6. An Academic Recommender System on large citation data based on clustering, graph modeling and deep-learning [1]

2.1 COMS - Conference Management Software

COMS is a complete conference management system designed for academic and scientific events, as its website says [4]. Provides meaningful support to organizers as well as a tool to manage the key requirements of their conference in one place. COMS supporting academic conferences since 2008. COMS is a web based system, but also is available for Android and iOS. It integrates the handling of abstracts and papers, from the submission and review process to the creation and display of the conference agenda and proceedings. It handles registrations and payments, onsite check-in, the use of multiple languages, website content and many more functions that are useful for the management of academic events. It is important to mention that COMS is not free to the average user, but require payment. The reader can learn more about this specific system by visiting its website [4].

2.2 EasyChair

EasyChair [5] is a conference management system designed to simplify the complexities of organizing conferences across various models. EasyChair is really famous among the academic community. Its features include facilitating call for submissions and participation, managing abstract and paper submissions with flexible forms and multiple file upload options, overseeing reviewer assignments and monitoring the program committee, conducting the reviewing process with online discussions and author rebuttal phases, facilitating communication and monitoring through email functionalities, providing analytics, editing and publishing programs, preparing conference proceedings, and handling attendee registration and online payments in multiple currencies. Its flexibility extends its utility beyond traditional conferences, as it can also be used for evaluating project proposals, teaching students paper writing and peer reviewing, and also generating program web pages for large conferences.

2.3 Converia - Conference Management Software

Converia [7] is the trird conference management software solution that examined in this thesis. Its main goal is to simplify the organization of conferences and congresses. According to its developers, it’s suitable for all kinds of companies, regardless of their size. It is said to be useful for professionals in various industries, including universities and higher ed-

ucation, professional conference organizers (PCOs) and organizations, as well as business conferences. Generally, the clients of Converia software is really famous universities and companies.

2.4 Recommender Systems for the Conference Paper Assignment Problem

Conry, Koren and Ramakrishnan [2] in their paper introduce a new approach in the conference paper assignment problem (CPAP). The primary input in CPAP, in this paper, is a papers × reviewers matrix of 'bids', expressing interest or disinterest of reviewers to review specific papers. The paper introduces a recommender systems approach for assigning conference papers to reviewers, addressing both reviewer-paper preferences modeling and assignment optimization. It proposes a method to integrate multiple information sources to learn preference models due to limited data availability. The approach is evaluated on real data from a conference, showing improved assignment quality using linear programming-based optimization. Different normalization strategies are suggested to balance reviewer rating scales, leading to significant enhancements in assignment quality compared to baseline methods. The study highlights the importance of integrating recommendation and optimization for effective conference paper assignment, with potential applications beyond academia.

2.5 The AI Conference Paper Assignment Problem

Goldsmith and Sloan [3] try to give an overview of work in progress in the CPAP problem. They mention that while the problem is solvable in polynomial time when considering only reviewers' preferences, it becomes more complex with the inclusion of expertise ratings. Various optimization criteria, such as maximizing total bid values or minimizing undesirable assignments, are explored in their paper. Feasibility testing for paper assignments is addressed using network flow techniques. Despite several algorithms being employed, including Integer Linear Programming (ILP) and constraint solvers, none guarantee optimal solutions. The paper also discusses the addition of expertise ratings to the bidding process, suggesting modeling the problem as a variant of the stable marriage problem. Open questions remain regarding the complexity of paper assignment problems under different constraints

and optimality criteria.

2.6 An Academic Recommender System on large citation data based on clustering, graph modeling and deep learning

This paper is a fundamental component in the implementation of the web system that is presented in this thesis because it explains the attempt to create a recommender system that will recommend specific reviewers for particular papers. Here, we will sum up the paper and in the section 4.5 we will dive into how the recommender system is implemented. The recommender system referred to here is utilized to make recommendations within the web application being developed. Utilizing data sourced from the DBLP AMiner Citation-network dataset, which comprises 5.3 million papers [8], the paper introduces a novel approach. It proposes a hybrid recommendation system, blending Content-based Filtering and Collaborative Filtering techniques within a multi-staged framework.The multi-staged framework involves three key steps: clustering analysis, the construction of a non-directed graph, and the incorporation of a sophisticated Deep Learning Recommender System named CATA++ [9]. These steps together improve the system's ability to offer personalized recommendations for both papers and reviewers.

Chapter 3

Technologies used

3.1 Type of the application

The type of web application chosen depends on the specific objectives it aims to achieve, tailored to meet the needs of its target audience and business requirements. Various categories of web applications exist, each serving distinct purposes. In this section, a brief overview of these categories will be presented, followed by an analysis of the selected type for our web-based recommendation system for scientific conferences.

3.1.1 Types of web-applications

Static Web Applications

Static Web Applications [10] are the most basic forms of web applications. They are web applications that can be sent directly to a user's browser without any changes made to the HTML, CSS, or JavaScript content on the server side. Creating and hosting static web applications is simple, as they do not need complex server-side processing. This makes them a budget-friendly choice for individuals or small businesses looking for a straightforward online presence. However, due to their simplicity, they offer limited functionality.

Dynamic Web Applications

Dynamic Web Applications [10] are more intricate and interactive compared to Static Web Applications. They employ client-side and server-side scripts (such as JavaScript, PHP, ASP, or JSP) to generate content in real-time. These web applications are linked to a database,

enabling them to offer customized experiences based on user interactions and preferences. They are well-suited for businesses, particularly if prioritizing user engagement and content diversity. However, due to their complexity, dynamic web applications require more effort in development and maintenance. They also necessitate a robust hosting environment and incur higher web development costs.

Single-Page Applications (SPAs)

Another type of web application is the Single-Page Application [10]. This type of application load a single HTML page and dynamically update content as users interact with the app. These categories of web applications are ideal for platforms where user experience and speed are critical, such as: social media platforms, email clients, cloud-based software. The benefit is that this web app type avoids reloading the entire page with each user action, leading to a smoother and faster user experience. However, they also come with challenges, particularly in Search Engine Optimization and initial load times, as the entire application must be loaded simultaneously. SEO, or Search Engine Optimization, refers to the process of optimizing a website to improve its visibility and ranking in search engine results pages. Single-Page Applications are built using JavaScript frameworks like React.js, Angular or Vue.js, which handle the dynamic loading of content and user interface elements.

Multi-Page Web Applications (MPAs)

In contrast to Single-Page Applications, Multi-Page Web Applications [10] (MPAs) require the complete reloading of pages from the server upon user interaction. These traditional web application architectures are better suited for platforms with extensive content and diverse functionalities, such as eCommerce sites and educational platforms. Unlike SPAs, MPAs efficiently manage complex structures and large databases. Moreover, they exhibit enhanced search engine optimization capabilities as individual pages can be indexed separately. However, MPAs may experience slower page transitions and increased resource consumption due to the necessity of server requests and page reloads for each new interaction.

The development of MPAs typically involves a more intricate back-end process to handle multiple pages and server interactions. Collaborating with proficient website development companies can ensure the seamless establishment of an effective online presence.

Progressive Web Applications (PWAs)

Another application that should be referred here is the Progressive Web Applications (PWAs) [11]. PWAs blend regular web pages or sites with mobile apps, letting users install them straight from their device's home screen, without having to go through app stores. PWAs stand out for using service workers or scripts in the background, which enable features like offline mode, push notifications, and syncing data in the background. Additionally, PWAs are responsive and shareable via URLs, ensuring they work well on different devices. With smooth animations and seamless scrolling, PWAs offer top-notch performance, which is particularly helpful for users with unreliable internet connections. By using standard web technologies like HTML, CSS, and JavaScript, developers can create PWAs efficiently and without hassle.

eCommerce Web Applications

In general, eCommerce Web Applications expedite online buying and selling [12]. They're complex systems that integrate various functionalities, including: product displays or catalogs, product search and filtering, shopping carts, payment processing, customer account and order management, customer service tools. These websites need to offer a smooth, easy-to-use interface to boost sales and keep customers coming back. Additionally, they must be able to handle fluctuations in website traffic and sales volumes. Also, security is paramount to protect sensitive customer data, including payment information.

Platforms like Shopify, Wix, Square, Magento, and WooCommerce [12] are famous examples, offering customizable templates and various plugins to enhance functionality. The advent of eCommerce web applications has revolutionized the retail industry, empowering businesses to reach broader audiences and operate seamlessly round-the-clock.

Content Management Systems (CMS)

A Content Management System (CMS) [13] facilitates the creation and modification of digital content, accommodating multiple users in a collaborative setting. The features of CMS platforms span a wide spectrum, encompassing web-based publishing, format management, version control, indexing, and search capabilities. These systems find applicability across various domains, including blogging, e-commerce, and news websites, catering to scenarios

where frequent content updates are necessary without demanding extensive technical expertise.

Popular CMS platforms such as WordPress, Joomla, and Drupal offer a plethora of templates and plugins for customization, eliminating the need for coding from scratch. Additionally, CMSs boast user-friendly interfaces, simplifying the process of content management and updates.

3.1.2 Selection of Type

The choice of web application type was meticulously determined by prioritizing factors such as user experience, application speed, and overall fluidity. Crucial considerations were given to the seamless integration between front-end, back-end, and database components. After comprehensive evaluation, it became evident that the Single Page Application (SPA) model best aligned with the objectives of our system. Leveraging the React.js framework for front-end implementation further reinforced our commitment to delivering an intuitive and responsive user interface.

3.2 Front-end technologies

Front-end development serves as the visual gateway to the user experience, shaping interactions and impressions. In the project that is presented, we used CSS and HTML for foundational styling and structure. JavaScript played a central role in enhancing interactivity, enabling dynamic content updates and user-driven functionalities. Using the powerful React.js library, we smoothly organized these elements into unified, adaptable interfaces. Additionally, integration of Chakra UI provided a streamlined design system, ensuring consistency and efficiency throughout development. This amalgamation of front-end technologies has led to the creation of an engaging and intuitive web-based recommendation system for scientific conferences.

3.2.1 JavaScript

JavaScript is a programming language used primarily by Web browsers to create a dynamic and interactive experience for the user. Most of the functions and applications that make the Internet indispensable to modern life are coded in some form of JavaScript [14].

The earliest incarnations of JavaScript were developed in the late 1990s for the Netscape Navigator Web browser. At the time, Web pages were static, offering little user interaction beyond clicking links and loading new pages. For the first time, JavaScript enabled animation, adaptive content and form validation on the page.

For many years, JavaScript was only compatible with a limited number of browsers. Microsoft's Internet Explorer, which had the largest user base, didn't support JavaScript until much later. Instead, Microsoft developed its own proprietary client-side script named JScript. In the early stages of Web development, developers aiming to create dynamic websites often had to pick one browser family over another. This situation was less than ideal because it hindered universal accessibility on the Internet. JavaScript did not become standardized and widely adopted until 1999. Even after standardization, browser compatibility remained an issue for over a decade.

JavaScript has evolved significantly since its inception, with the introduction of features like asynchronous programming, which allows for non-blocking execution and improved performance. This has facilitated the development of complex web applications that can handle multiple tasks simultaneously without causing delays. Moreover, JavaScript now supports a wide range of functionalities such as handling events, manipulating the DOM (Document Object Model), making HTTP requests, and interacting with APIs (Application Programming Interfaces). With the advent of frameworks and libraries like React, Angular, and Vue.js, developers can build sophisticated single-page applications (SPAs) and progressive web apps (PWAs) more efficiently, further expanding the capabilities of JavaScript in modern web development.

3.2.2 CSS

CSS is standing for Cascading Style Sheets [15]. Is a style sheet language that describes the style of the HTML document. It describes how HTML elements are displayed on a page. CSS saves a lot of work as it may control the layout of multiple pages at once. Style formatting is usually saved in a separate .css file. Thanks to this, someone can modify the layout of the whole site by changing only one file. Typically, everyday users opt for CSS in their projects for several straightforward reasons: it streamlines webpage design, speeds up page loading, and is user-friendly.

3.2.3 HTML

HTML stands for Hypertext Markup Language, and it is a standardized markup language system for displaying documents on web browsers. HTML defines the structure of a website while other CSS their appearance and JavaScript their behaviour. ”Hypertext” describes the links between pages of content on the web that someone can access immediately by clicking on said link. ”Markup” refers to the method by which text, images, and other content is annotated to then be displayed. Examples of these markup elements include <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <video>, <ul>, <ol>, <li> and many more.

3.2.4 React.js

React.js [16], often referred to as ReactJS, is a JavaScript library renowned for its flexibility and efficiency in crafting interactive user interfaces (UIs) for both web and native applications. Its component-based architecture empowers developers to create UI elements like buttons or search bars, which can then be reused throughout an application. This not only simplifies the development process but also improve maintainability. One of React’s distinguishing features is JSX (JavaScript XML), which resembles HTML and allows developers to write markup directly within their JavaScript code. This unique capability simplifies coding by providing a more natural way to structure components. React’s influence extends beyond the browser; it also plays a crucial role in mobile app development with React Native.

Above all else, one of the key attractions of using React is its declarative nature — it makes code easier to understand and debug because it describes what your application should look like in various states rather than detailing step-by-step instructions on achieving that look. This makes your code more predictable and easier for both you and others on your team to work with.

What does React.js offer that sets it apart from other frameworks? React.js has some core features that have give him an edge compare to other frontend libraries and frameworks. Firstly, React is known for its component-based architecture. Components are independent, reusable pieces of code that function as building blocks of a React application. The combination of these components creates complex user interfaces. Moreover, each component can manage its own state, which means it can hold and manipulate data within itself.

Another significant feature is JSX (JavaScript XML) [17], a syntax extension to JavaScript

that allows developers to write HTML-like code within their JavaScript. This bridges the gap between JavaScript and HTML, making coding more intuitive and efficient.

React also uses a virtual DOM (Document Object Model) [18] , a programming concept where an ideal or ”virtual” representation of UI is kept in memory and synced with the ”real” DOM by React’s reconciliation process. This feature brings about high efficiency in rendering web pages, making applications faster and more responsive.

State management in React is flexible; while there is built-in support through ’state’ in class components or ’useState’ hook in functional components, React also accommodates third-party libraries like Redux for larger applications.

Finally, as stated above, besides being excellent for building user interfaces on the web with its main library **react-dom**, it also supports mobile platform development using react-native. This makes it possible to create both web apps and native mobile apps using largely similar coding patterns. All the above reasons, make React.js perfect for the implementation of our system.

3.2.5 Chakra UI

Chakra UI [19] is a component-based library. It’s made up of basic building blocks that can help you build the front-end of your web application. It is customizable and reusable, and most importantly it supports React.js (this is why it is selected), along with some other libraries too.

3.3 Back-end Technologies

Back-end refers to the server-side of a web application responsible for handling data processing, logic, and database management. It acts as the backbone of the system, facilitating seamless communication between the front-end and the database. In the web-based recommendation system for scientific conferences, we employed a combination of powerful technologies to ensure robustness, scalability, and efficiency. The technologies utilized in the back-end include Node.js, Express.js, SQL, and MySQL and will be analyzed below.

3.3.1 Node.js

Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser. It is worth mentioning that Node.js is written in C, C++, and JavaScript. Node.js [20] is asynchronous, so requests can be handled without having any dependency on one another, which improves efficiency and throughput. This design pattern is known as non-blocking code execution. Also, Node.js is single-threaded and as a result is able to handle multiple concurrent clients without creating multiple threads.

Some important features of Node.js are [21]:

1. **User-Friendly**: Node.js is beginner-friendly and easy to get started with. Thanks to abundant tutorials and a vibrant community, diving in is hassle-free.

2. **Scalability**: It offers extensive scalability for applications. Despite being single-threaded, Node.js can manage a large number of simultaneous connections efficiently.

3. **Speediness**: Non-blocking thread execution enhances the speed and efficiency of Node.js.

4. **Abundant Packages**: The NPM ecosystem boasts a vast array of open-source Node.js packages, simplifying development tasks. There are over a million packages available today.

5. **Robust Backend**: Node.js, written in C and C++, ensures speed and incorporates features like networking support.

6. **Cross-Platform**: Node.js supports multiple platforms, enabling the creation of SaaS websites, desktop apps, and even mobile apps.

7. **Ease of Maintenance**: Node.js is favored by developers as both frontend and backend can be managed with JavaScript, streamlining the development process.

3.3.2 Express.js

Express.js [22] [23] stands out as the top choice for Node.js backend frameworks, deeply integrated into the JavaScript ecosystem. It's a lightweight, rapid, and reminiscent of Sinatra framework, offering a robust toolkit for building scalable backend applications. With its

routing system and simplified features, Express.js empowers users to extend the framework, crafting powerful components tailored to specific application needs. Offering a comprehensive suite of tools for web applications, handling HTTP requests and responses, managing routing, and integrating middleware, Express.js is ideal for constructing and deploying large-scale, enterprise-ready applications. Additionally, it comes with a convenient command-line interface tool (CLI) through Node Package Manager (NPM), simplifying package management for developers.

3.3.3 SQL

Structured Query Language (SQL) ia a really famous language. Is a standardized programming language that is used to manage relational databases and perform various operations on the data in them. Initially created in the 1970s, SQL is regularly used not only by database administrators, but also by developers writing data integration scripts and data analysts looking to set up and run analytical queries.

3.3.4 MySQL

MySQL [24] is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is an organized collection of data. It can range from a basic shopping list to a photo album or a repository for the extensive information within a business network. Specifically, a relational database is a digital storage system that gathers data and arranges it based on the relational model. In this model, tables are made up of rows and columns, and connections between data items adhere to a clear logical structure. An RDBMS (Relational Database Management System) is essentially the suite of software tools employed to create, oversee, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, data-driven B2B services. Its open-source nature, stability, and rich feature set, paired with ongoing development and support from Oracle, have meant that internet-critical organizations such as Facebook, Flickr, Twitter, Wikipedia, and YouTube all employ MySQL backends.

3.4 Authentication

The authentication has a paramount important in the applications that respect themselves. Authentication [25] is the process of verifying the identity of a user or information. Authentication is used to prove that some fact or some document is genuine, true or valid. The process involves users confirming their identity by providing their credentials. This piece of information is shared between the user and the service or system where authentication is required.

3.4.1 JSON Web Tokens

The application use JSON Web Tokens for authentication. JSON Web Tokens (JWTs) is one of the most common ways to implement token-based authentication. JWTs are an open standard that defines a self-contained way to transmit information between parties as JSON objects securely. A JSON [26] web token is JSON (JavaScript object notation) with some extra structure. JWTs include a header and payload that use the JSON format. Optionally, the tokens can be encrypted or signed with a message authentication code (MAC). Signed tokens are commonly referred to as JSON web signatures (JWS) and encrypted tokens as JSON web encryption (JWE).

Chapter 4

Implementation

4.1 System design

The first step of the implementation is the theoretical design of the system in order to understand in detail how the system will work. In the image below the system is depict with 4 different entities. The front-end which include the user interface, the client-side routing and the integration with external service, the back-end which include the routing and middleware and the data handling. Also it depicts the database which receive data from the back-end and query data to. Last but not least the recommendation system which communicates mainly with the database in order to take the needed papers and reviewers and send the recommendations/assignments.

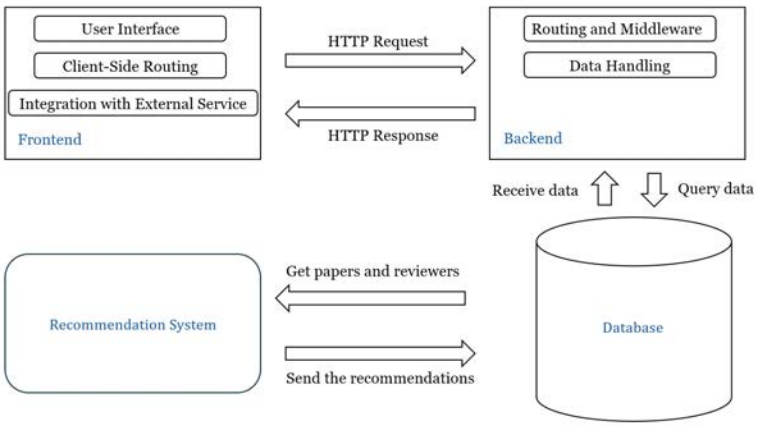

Figure 4.1: The design of the system

4.2 Requirements Analysis

Before looking into technical aspects, it's crucial to outline the various capabilities that users have when utilizing this web-based recommendation system for scientific conferences. Below is an analytical list of user possibilities:

1. Users can create a new account using the registration page.
2. Users can log in to the application using the credentials provided during the registration process.
3. Users have the option to log out of the application at any time after registering or logging in.
4. Once logged in, users can navigate through the application's features and functionalities.
5. Users can create new scientific conferences, providing all the necessary information required for a scientific conference.
6. Users can view the conferences that have been created
7. Users possess the ability to add scientific papers in multiple ways:
    - Adding papers one by one, providing only the title and abstract.
    - Uploading papers individually, where the system extracts the title and abstract automatically.
    - Importing papers via Excel or CSV files, allowing multiple papers to be added simultaneously.
8. Users also have the capability to add reviewers for their conferences using various methods:
    - Adding reviewers one by one, providing their email and name.
    - Uploading reviewers individually, with the system extracting details from the provided information.
    - Importing reviewers via Excel or CSV files, facilitating the addition of multiple reviewers at once.

By offering these functionalities, the system aims to provide users with a comprehensive platform for managing scientific conferences efficiently and effectively.

4.3 Database Design

A basic step of the implementation is the design of the database. In the database that has been created there are the below tables:

- **users**: represents every user of the application. The fields that exist inside the table are:
    1. **userId**: The id of user. Every user has a different id.
    2. **firstName**: The first name of the user.
    3. **lastName**: The last name of the user.
    4. **username**: The username of the user that will be used for the application.
    5. **password**: The hashed password is saved here.
- **conferences**: represents every conference that the user create. Contains the fields:
    1. **conferenceId**: The id of the conference. Every conference has a different id.
    2. **userId**: The id of the user that created the conference.
    3. **author**: The name or identity of the individual or organization responsible for organizing or hosting the conference.
    4. **title**: The title of the conference.
    5. **acronym**: The unique acronym of the conference used for identification purposes.
    6. **webpage**: The conference's webpage. It is a string.
    7. **city**: The city of the conference.
    8. **country**: The country where the conference takes place, indicating the geographical region of the event.
    9. **firstday**: The date marking the commencement of the conference, indicating when the event begins.

10. **lastday**: The date marking the conclusion of the conference, indicating when the event ends.
11. **primaryarea**: The main focus or topic area of the conference, highlighting the primary subject matter of the event.
12. **secondaryarea**: Additional focus or topic areas covered in the conference, supplementing the primary subject matter.
13. **organizer**: The individual, group, or organization responsible for planning and coordinating the conference.
14. **start_recommendation**: The date or time when recommendations or proposals for the conference begin to be accepted.
15. **finished_recommendation**: The date or time when the period for submitting recommendations or proposals for the conference ends.

- **paper**: represents every paper that a user import in the application. Contains the fields:
  1. **paperId**: The unique identifier assigned to each paper, ensuring distinction between papers.
  2. **title**: The title of the paper.
  3. **abstract**: The abstract of the paper. It is useful for the recommendation system.
- **reviewers**: represents the reviewers of the paper. Contains the fields:
  1. **reviewerId**: Unique identifier for every reviewer.
  2. **email**: The email of the reviewer.
  3. **name**: The name of the reviewer.
- **conferencetopapers**: Connects the conferences with the paper. The fields that are contained:
  1. **conferenceId**: Foreign key. The unique id of a conference.
  2. **paperId**: Foreign key. The unique id of a paper.
- **recommendations**: Here are saved the assignments for every reviewer. The fields that are contained:

1. **reviewerId**: Foreign key in the table recommendations.
2. **conferenceId**: Foreign key in the table recommendations.
3. **assignment1**: The fist assignment that recommendations system will send for the specific reviewer
4. **assignment2**: The second assignment that recommendations system will send for the specific reviewer
5. **assignment3**: The third assignment that recommendations system will send for the specific reviewer

Below (Figure 4.2), you'll find the Entity Relationship Diagram diagram of our database, exported from MySQL Workbench and used with the MySQL server. This diagram clearly shows the different parts of our database, like users, conferences, papers, reviewers, conferencetopapers, and recommendations. It's a visual map that helps us see how these parts are connected to each other, giving us a clear picture of how our database is structured.

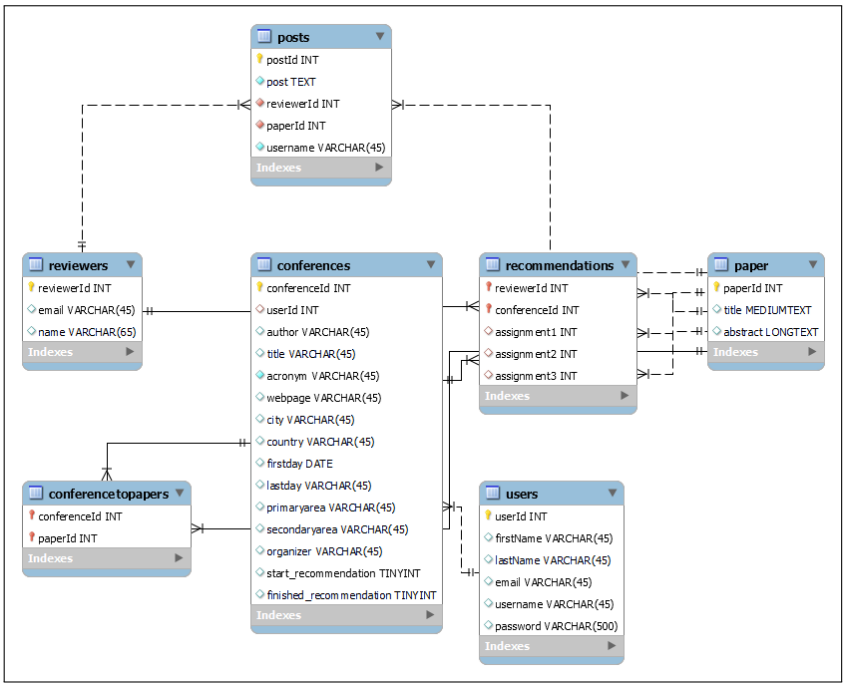

Figure 4.2: The Entity Relationship Diagram Diagram

4.4 Web server

For the web server, the technologies that were used include Node.js and the Express.js framework. To set up the project, Node.js and Express.js were installed. Node.js provides the

runtime environment for JavaScript code, while Express.js simplifies the process of building web applications and APIs in Node.js.

The project was initialized with the basic setup for any regular Express application. The installation process involved creating a new directory for the project, here named **server**, navigating into it, and initializing a new Node.js project using `npm init -y`. This command generated a `package.json` file to manage project dependencies.

Express.js was then installed as a dependency for the project using `npm install express`. This step ensured that the project had access to the Express.js framework for building the web server.

4.4.1 server.js file

The structure of the server folder is displayed in the 4.3. The most significant file and the entry point of the application is the `server.js` file, where the Express.js application is initialized and configured. In the provided `server.js` code, the Express.js app is created using `const app = express();`. Additionally, necessary dependencies such as `path`, `mysql`, and `express.json()` middleware are imported and utilized.

Middleware functions are then employed to log request paths and methods using `app.use((req, res, next) => {...});`. This allows for the monitoring of incoming requests and their corresponding HTTP methods.

Furthermore, the application defines various routes using `app.use()` to handle different API endpoints such as user authentication, conference management, reviewer assignments, paper submissions, posts, and recommendations. Each route is prefixed with `/api/` to organize the API endpoints:

- `/api/user`: Handles user authentication.
- `/api/conference`: Manages conference-related operations.
- `/api/reviewer`: Assigns reviewers to conferences.
- `/api/paper`: Manages paper submissions.
- `/api/post`: Handles posts.
- `/api/recommendation`: Manages recommendations.

Finally, the application listens on port 5000 using `app.listen(5000, function() {...});`, indicating that it is ready to accept incoming HTTP requests.

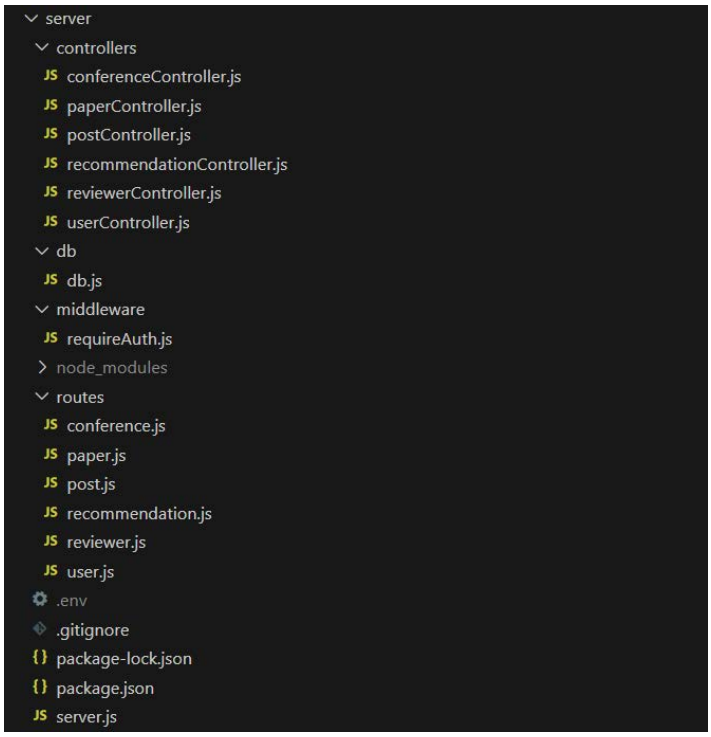

Figure 4.3: The server file

4.4.2 middleware folder

The middleware folder within the server directory contains essential components for implementing middleware functionalities in the application. This folder serves as a dedicated location for storing middleware-related scripts, allowing for better organization and maintenance of middleware logic.

One crucial middleware file located within this folder is `requireAuth.js`. This file defines a middleware function named **requireAuth**, responsible for enforcing authentication requirements for incoming requests. Upon receiving a request, requireAuth validates the authorization header to ensure the presence of a valid authentication token. If the token is missing or is invalid, the middleware returns a 401 Unauthorized error response.

To verify the authenticity of the token, **requireAuth** extracts the token value from the authorization header and utilizes the jsonwebtoken package to decode and verify its validity against the application's secret key. If the token is successfully verified, the middleware proceeds to retrieve the associated user's email from the database. If a user with the pro-

vided email exists, their email is attached to the request object as req.user, granting access to authenticated user data in subsequent request handlers.

However, if the token verification fails or the associated user is not found in the database, **requireAuth** responds with a 401 Unauthorized error, indicating that the request lacks proper authorization. By encapsulating authentication logic within this middleware, `requireAuth.js` promotes code reusability and maintains a separation of concerns, contributing to a more modular and maintainable codebase.

4.4.3 db folder

The db folder contains a critical file named `db.js`, which facilitates the connection between the development environment and the MySQL database. This file encapsulates essential information required for establishing the connection, such as the MySQL server host, username, database name, and password. It utilizes the **mysql2** package to create a connection pool, ensuring efficient management of database connections.

The connection pool configuration includes parameters such as the maximum number of connections, the maximum number of connections waiting in the queue, and whether to wait for connections when all connections are busy. These parameters can be adjusted based on specific project requirements.

The pool object, once created, is exported from the `db.js` file, allowing other modules within the application to utilize the established database connection for executing queries and interacting with the MySQL database. By centralizing database connection logic in the `db.js` file, the application promotes modularity, maintainability, and separation of concerns.

4.4.4 controllers folder

The controllers folder is a key part of the project, holding all the essential functions and setups for our backend operations. It's arguably the most important folder of the web server. In Table 4.1, the reader can find brief rundown of each controller, before explained in detail a little further down. This makes it easier for readers to get a general idea of how every controller works before we get into the specifics.

Name of controller	Description
`conferenceController.js`	Manages operations related to conferences, such as creating, updating, and deleting conferences.
`paperController.js`	Handles paper-related functionalities, including submission, review assignment, and retrieval.
`postController.js`	Responsible for managing posts, including creation, deletion, and retrieval of posts.
`recommendationController.js`	Manages recommendations, including generating recommendations based on user preferences and history.
`reviewerController.js`	Handles reviewer-related operations, such as assigning reviewers to papers or conferences.
`userController.js`	Manages user-related functionalities, including authentication, registration, and profile management.

Table 4.1: Summary of controllers in the controllers folder

More analytically we will dive into every controller file below and every function inside every controller will be explained.

- `conferenceController.js`: This file contains all the functions that are related to the conference management. This file is a substantial part of our project implementation. The functions that contains are:

    1. **insertConference**: This function is responsible for inserting a new conference into the database. It takes various parameters such as the userId, author, title,

acronym, webpage, city, country, first day, last day, primary area, secondary area, and organizer from the request body. It first checks if the conference name already exists in the database. If it does not exist, it inserts the conference details into the database and then retrieves the inserted conference's information. Finally, it returns the inserted conference as a JSON response.

2. **getConference**: The purpose of this function is to retrieve information about a conference based on its name. It takes the name of the conference from the request body and queries the database to fetch the conference details. If a conference with the given name is found, it returns the conference information as a JSON response. If no conference is found, it returns a JSON response indicating an error.

3. **getConferencesByAuthor**: This function retrieves all conferences authored by a specific author. It takes the author's ID as a parameter from the request URL and queries the database to fetch all conferences associated with that author. It then returns the list of conferences as a JSON response.

4. **getConferenceById**: This function is used to fetch the details of a conference based on its ID. It takes the conferenceId as a parameter from the request URL and queries the database to retrieve the conference information. If a conference with the given ID is found, it returns the conference details as a JSON response. If no conference is found, it returns a JSON response indicating an error.

- `paperController.js`: this controller is responsible for every functionality that is related to paper management and communicates the most of the time with the paper table. The functions that exist inside the file are:

  1. **insertPaper**: This function is responsible for inserting a new paper into the database. It takes parameters such as the conferenceId, title, and abstract from the request body. First, it checks if a paper with the given title already exists in the database. If it does, it returns an error response. If the paper does not exist, it inserts it into the paper table. Then, it retrieves the inserted paper from the database and inserts an entry into the **conferencetopapers** table to associate the paper with the conference. Finally, it returns the inserted paper as a JSON response.

  2. **getPapersByConferenceId**: This function retrieves all papers associated with a

specific conferenceId from the database. It takes the conferenceId as a parameter from the request URL. It performs a query to join the **paper** and **conferencetopapers** tables based on the conferenceId to fetch all papers linked to that conference. It then returns the list of papers as a JSON response.

3. **getPaper**: The purpose of this function is to retrieve details of a paper based on its ID from the database. It takes the paperId from the request body. It queries the **paper** table to fetch the paper information using the given paperId. If a paper with the given ID is found, it returns the paper details as a JSON response. If no paper is found, it returns a JSON response indicating an error.

- `recommendationController.js`: this file contains a function that is related with the recommendation of papers on a specific reviewer

1. **getAssignmentByReviewer**: This function retrieves assignments for reviewers associated with a specific conference ID from the database. It takes the conferenceId from the request body. The function first attempts to fetch assignments from the database by querying the **recommendations** table and joining it with the **reviewers** table based on the reviewerId. It selects the conferenceId, assignment1, assignment2, assignment3, reviewer email, and reviewer name for each assignment. The WHERE clause filters the assignments based on the provided conferenceId. If no assignments are found for the specified conferenceId, the function returns a 404 status with a JSON response indicating that no assignments were found. If assignments are found, it returns a 200 status with a JSON response containing the assignments data

- `reviewerController.js`: This file contains functions that are related with the reviewers management (create and get). More specifically these functions are:

1. **insertReviewer**: serves to add a new reviewer to the system for a specific conference. Upon receiving a request containing the conferenceId, reviewer's email, and name, it first checks if a reviewer with the provided email already exists in the database. If a reviewer with the given email is found, it returns the existing reviewer's information. If not, it proceeds to insert the new reviewer into the **reviewers** table. Following the insertion, it retrieves the inserted reviewer's details from the database. Subsequently, it adds an entry into the **recommendations** table

to associate the newly added reviewer with the provided conferenceId. Finally, it responds with a JSON object containing the details of the inserted reviewer. In case of any errors during this process, it returns a 400 status along with an error message. This function ensures that reviewers are uniquely identified by their email addresses and can be assigned to conferences for reviewing purposes, maintaining the integrity of the reviewer and conference data within the system.

2. **getReviewersByConferenceId**: retrieves the list of reviewers associated with a specific conferenceId from the database. Upon receiving the conferenceId from the request parameters, it constructs a SQL query to join the **reviewers** table with the **recommendations** table based on the reviewerId. It then selects all attributes from the **reviewers** table where the conferenceId in the **recommendations** table matches the provided conferenceId. Upon successful execution, it returns a JSON response containing the list of reviewers associated with the conferenceId. In case of any errors during the database query, it responds with a 400 status along with an error message. This function facilitates the retrieval of reviewers assigned to a particular conference, aiding conference organizers in managing and overseeing the review process efficiently.

- `userController.js`: This file encompasses all functions related to user management, including login and registration functionalities. It serves as the central hub for handling user-related operations within the system.

  1. **getByUsername**: This function retrieves user information from the database based on the username provided. It queries the **users** table to fetch a user with the given username. If a user is found, it returns the user object. If no user is found, it returns null. It handles any errors that occur during the database query.

  2. **checkUsername**: The purpose of this function is to check if a username already exists in the database. It queries the **users** table to find if there is any user with the given username. If a user with the username exists, it returns 0, indicating that the username is already in use. If no user with the username exists, it returns 1, indicating that the username is available. It handles any errors that occur during the database query.

  3. **loginUser**: This function handles the login process for a user. It takes the user-

name and password from the request body. It first calls the **getByUsername** function to retrieve the user information based on the provided username. If a user is found, it compares the provided password with the hashed password stored in the database using bcrypt. If the passwords match, it generates a JWT token using the user's email and sends both the user object and the token in the response. If the user is not found or the passwords do not match, it returns an error response.

4. **registerUser**: This function handles the user registration process. It takes the user's first name, last name, email, username, and password from the request body. It first calls the **checkUsername** function to ensure that the username is not already in use. If the username is available, it hashes the password using bcrypt, inserts the user's information into the **users** table, and retrieves the inserted user's information from the database. It then generates a JWT token using the user's email and sends both the user object and the token in the response. If the username is already in use, it returns an error response.
5. **getUserByUsername**: This function checks if a user is logged in by checking if the 'user' object is present in the session. If the user is logged in, it sends a response indicating that the user is logged in along with the user object. If the user is not logged in, it sends a response indicating that the user is not logged in.
6. **allUsers**: This function retrieves user information from the database based on the username provided in the request parameters. It queries the **users** table to fetch the user with the given username. If a user is found, it returns the user object. If no user is found, it returns an error response. It handles any errors that occur during the database query.

4.4.5 routes folder

The routes folder connects the server.js file with the controllers folder. More specifically the routes folder contains these files:

- `user.js`
- `conference.js`
- `reviewer.js`
- `recommendation.js`
- `paper.js`
- `recommendation.js`

Below, we'll conduct a detailed analysis of every function within the routes folder, examining how each function in the controllers folder is connected to specific routes

- `user.js`:
  - `/api/user/loginUser`: POST route for user login functionality. It calls the function `loginUser` from the `userController.js` file, handling user login logic.
  - `/api/user/register`: POST route for user registration functionality. It calls the `registerUser` function from the `userController.js` file, managing user registration logic.
  - `/api/user/token`: GET route for checking user authentication token. It calls the `token` function from the `userController.js` file to validate user authentication.
  - `/api/user/edit`: PUT route for editing user information. It calls the `editUser` function from the `userController.js` file, handling user profile editing.
  - `/api/user/getUserByUsername/:id`: GET route for retrieving user information by username. It calls the `getUserByUsername` function from the `userController.js` file, fetching user details based on the provided username.

- `paper.js`:
  - `/api/paper/insertpaper`: POST route for inserting a new paper into the system. It calls the `insertPaper` function from the `paperController.js` file, handling the logic to insert the paper into the database.
  - `/api/paper/:id`: GET route for retrieving all papers associated with a specific conference ID. It calls the `getPapersByConferenceId` function from the `paperController.js` file, which fetches and returns the papers related to the specified conferenceId.
  - `/api/paper/`: POST route for retrieving details of a paper based on its ID. It calls the `getPaper` function from the `paperController.js` file, which retrieves and returns the details of the paper specified in the request body.
- `reviewer.js`:
  - `/api/reviewer/insertReviewer`: POST route for inserting a new reviewer into the system. It calls the `insertReviewer` function from the `reviewerController.js` file, managing the logic to add a new reviewer to the database.
  - `/api/reviewer/:id`: GET route for retrieving reviewers associated with a specific conference ID. It calls the `getReviewersByConferenceId` function from the `reviewerController.js` file, fetching and returning the list of reviewers related to the provided conferenceId.
- `conference.js`:
  - `/api/conference/insertConf`: POST route for inserting conference information into the database. It calls the `insertConference` function from the `conferenceController.js` file, managing the logic to add conference details to the database.
  - `/api/conference/getConf`: POST route for retrieving conference information. It calls the `getConference` function from the `conferenceController.js` file, handling the logic to fetch conference details.

  - `/api/conference/:id`: GET route for retrieving conferences by authorId. It calls the `getConferencesByAuthor` function from the `conferenceController.js` file, fetching and returning conferences associated with the provided authorId.
  - `/api/conference/id/:id`: GET route for retrieving conference by conferenceId. It calls the `getConferenceById` function from the `conferenceController.js` file, fetching and returning the conference details based on the provided conferenceId.

Combining all these routes forms an API, which serves as the interface for interacting with the application's functionality. Below, it is summarized each route within the API for better understanding for the reader:

- `/api/paper/insertpaper`
- `/api/paper/:id`
- `/api/paper/`
- `/api/user/loginUser`
- `/api/user/register`
- `/api/user/token`
- `/api/user/edit`
- `/api/user/getUserByUsername/:id`
- `/api/reviewer/insertReviewer`
- `/api/reviewer/:id`
- `/api/conference/insertConf`
- `/api/conference/getConf`
- `/api/conference/:id`
- `/api/conference/id/:id`

4.5 Recommendation System for Scientific Papers

This section explains the recommendation system developed by Vaios Stergiopoulos [1], detailing its functionality for selecting and recommending papers to reviewers.

The recommendation system is primarily developed in Python and consists of four main .py files: `db_utils.py`, `main.py`, `r2p_utils.py` and `reviewer.py`, along with a directory that contains essential files named `data/alphabetical_researchers`.

The core idea of the recommendation system is to create a profile for each reviewer. After generating these profiles, the system compares each reviewer's profile with every paper available, utilizing BERT models for preprocessing and encoding the text of both the reviewer profiles and the papers. This process results in a similarity matrix for each reviewer-paper pair. The system then selects the top three papers from this matrix for each reviewer to review. Also, it is really important to mention that the recommendation system has communication with the database, and especially with the tables `paper`, `reviewers`, `recommendations`.

Moreover, the `db_utils.py` file handles database operations, ensuring seamless data retrieval and storage. The `main.py` orchestrates the overall workflow of the system, while `r2p_utils.py` contasins utilities for processing research-to-reviewer mappings. Additionally, the `reviewer.py` module manages reviewer profiles and preferences, crucial for fine-tuning the recommendation process.

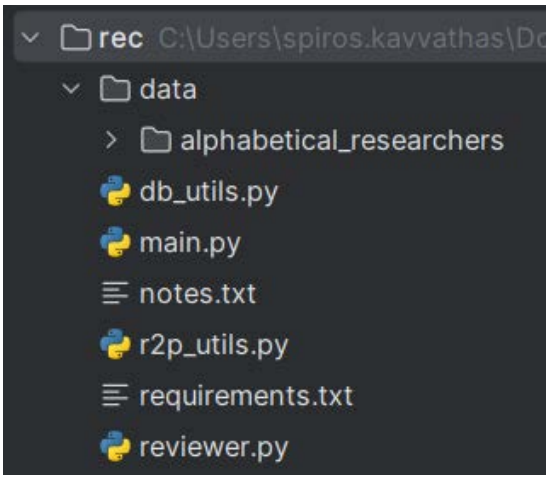

Figure 4.4: The folder that contains the code of the recommendation system

Furthermore, the `data/alphabetical_researchers` directory houses curated datasets of researchers, organized alphabetically, which serve as foundational resources for matching papers to appropriate reviewers.

4.5.1 Requirements

Before analyzing each file, it's important to outline the necessary requirements. To ensure the application displays and functions correctly, several dependencies must be in place. The `pandas` library is essential for data manipulation and analysis, providing data structures like DataFrames. The regex module is utilized for advanced string matching and manipulation using regular expressions. The `tensorflow` framework is crucial for building and deploying machine learning models, particularly neural networks. Complementing this, tensorflow-text extends TensorFlow with natural language processing (NLP) capabilities. The `scikit-learn` library offers a wide range of machine learning algorithms for tasks such as classification, regression, and clustering. For database interactions we need to use `mysql-connector-python` that provides a robust interface for connecting to MySQL databases. The `fuzzywuzzy` package is employed for fuzzy string matching, which is useful in data cleaning and record linkage. Finally, `numpy` is a fundamental package for numerical computations, offering support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

4.5.2 main.py

The `main.py` script contains the basic flow in order to assign scientific papers with suitable reviewers for peer review. The script begins by initializing necessary libraries and components, such as TensorFlow for natural language processing tasks using BERT models.

The BERT model [27], introduced by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," revolutionized natural language processing by pre-training bidirectional representations using masked language modeling (MLM) and next sentence prediction (NSP) objectives. BERT leverages a large corpus, including the Toronto Book Corpus and Wikipedia, to jointly condition on both left and right context across all layers. This allows it to be fine-tuned for various tasks such as question answering and language inference with minimal modifications. BERT achieved state-of-the-art results on eleven NLP tasks, notably improving scores on GLUE, MultiNLI, and SQuAD benchmarks. Conceptually simple and empirically powerful, BERT uses absolute position embeddings and recommends right-padding inputs. The model predicts masked tokens efficiently but is less suited for text generation. During pre-training, BERT randomly masks 15% of tokens, and it must predict

the original tokens and determine if two given sentences are consecutive. Additionally, PyTorch's scaled dot-product attention (SDPA) operator is employed for improved performance on compatible hardware, with users able to explicitly request SDPA in the from_pretrained() function for torch>=2.1.1.

More specifically, the BERT models that are used and explained here are the models: `bert_en_uncased_preprocess/3` and `bert_en_uncased_L-2_H-128_A-2/1`. The first one erves as a preprocessing layer specifically designed for BERT models trained on English text (bert_en_uncased). It standardizes and tokenizes input text into a format suitable for BERT embeddings and the second one is designed for more specific tasks requiring lower computational resources and memory compared to larger BERT variants. After the BERT models the script, establishes connections to the database where all the informations are saved, ensuring it can retrieve and update relevant information seamlessly.

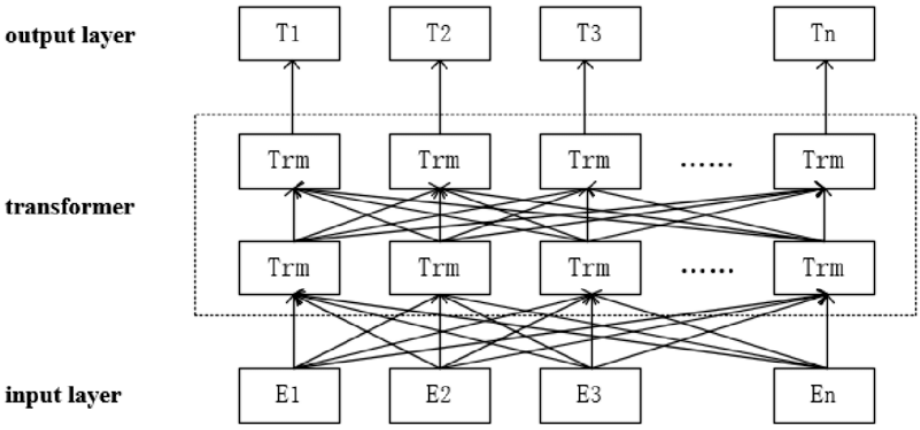

Figure 4.5: The structure of BERT model

The script runs continuously, checking if there are any new conferences that need reviewer assignments. Once it finds a conference that's active and needs reviewers (by checking if the start_recommendation column has a value of 1), it gathers information about the papers submitted to that conference. This includes details like the title and abstract of each paper. This information is stored and organized to make it easier to work with.

Next, the script looks up for potential reviewers for each conference from the database (`reviewers` table in database). It fetches the details about these reviewers (name and email). This information is important because it helps in matching papers to reviewers who have relevant knowledge in the same field.

To decide which papers should go to which reviewers, the script uses the BERT em-

beddings. More specifically this means that it compares the content of each paper with the research interests of each reviewer to find similarities. Papers that closely match a reviewer's expertise are prioritized for that reviewer.

Finally, the script assigns up to three papers to each reviewer based on these similarities. It updates the database (`conferencetopapers` table) with these assignments and moves on to the next conference that needs attention. This process repeats continuously, ensuring that conferences get reviewed efficiently and that reviewers are matched with papers that they are best suited to evaluate.

In essence, `main.py` automates and streamlines the process of matching scientific papers with reviewers, using advanced tools to improve the quality and efficiency of academic peer review.

4.5.3 db_utils.py

The `db_utils.py` script contains a class, DButils, designed to manage interactions with the MySQL database. This script includes methods for connecting to the database, inserting data into the tables, updating records, and retrieving information needed for assigning reviewers to conference papers. For instance, it can load paper details from a CSV file into the database, insert reviewer information, and update tables that link conferences to the papers and recommendations.

The methods in DButils cover a wide range of database operations. There are methods for connecting to the database, which include error handling to ensure the script handles connection issues gracefully. The script also includes methods to insert data into the tables `paper`, `reviewers`, `conferences`, and `conferencetopapers`, ensuring that all relevant data is stored correctly. Additionally, it provides update functionalities, such as updating recommendations for reviewers and marking conferences as finished once all assignments are made. Select methods are used to retrieve data, such as fetching paper IDs for a specific conference or getting reviewer IDs from the recommendations table, which are crucial for the logic implemented in main.py.

4.5.4 r2p_utils.py

The `r2p_utils.py` script is designed to facilitate the handling and processing of data related to academic papers and researchers, primarily for the purpose of assigning reviewers

to papers based on their research interests. The script includes functions for loading data from CSV files, cleaning and processing text data, and preparing it for further analysis. For example, the `load_reviewers` function loads and processes a CSV file containing reviewer information, converting all text to lowercase for consistency. Similarly, the `load_data` function loads paper data from a CSV file, cleans the titles and abstracts by converting them to lowercase and removing special characters, and merges these cleaned fields into a single column for easier processing.

Additionally, the script includes functions for generating BERT embeddings for text data, which can be used to measure the similarity between paper abstracts and reviewer research interests. The `get_bert_embeddings` function creates a BERT model to generate these embeddings. Another key function, `create_paper2researcher`, matches papers to researchers by calculating similarity scores and creating lists that link each paper to potential reviewers along with their similarity scores. This helps in the recommendation process, ensuring that papers are assigned to the most suitable reviewers based on their expertise. Some functions, like `get_user_query` and parts of `create_paper2researcher`, were used for testing purposes and are not part of the final project implementation.

4.5.5 reviewer.py

The `reviewers.py` script defines a ‘Reviewer‘ class used for managing reviewer information and creating research profiles based on fuzzy name matching. Each reviewer has an ID and a name. The class includes methods to calculate a fuzzy score to compare the reviewer’s name with a researcher’s name from a dataset using the FuzzyWuzzy library, to add keywords from a publication to the reviewer’s research profile, and to create a comprehensive research profile by matching the reviewer’s name with names in a CSV file containing researcher information. The profile is built by aggregating research texts associated with names that match the reviewer’s name with a high fuzzy score, ensuring relevant research interests are compiled accurately.

4.5.6 data/alphabetical_researchers

The `data/alphabetical_researchers` directory plays a crucial role in the system’s workflow. This directory consists of various files containing detailed information about researchers and reviewers, sourced from the AMiner database [28]. The AMiner database is

a comprehensive resource, encompassing 5,354,309 papers and 48,227,950 citation relationships. To enhance the efficiency of the recommendation system, all information in this directory is organized alphabetically. This organization not only facilitates quicker data access and processing but also improves the overall speed of generating recommendations. Each file within this directory corresponds to researchers and reviewers whose names start with a specific letter of the alphabet. The careful alphabetical arrangement ensures that the system can swiftly match reviewers with relevant papers. By systematically processing and structuring the data, the system can efficiently handle large volumes of information, thereby providing accurate and timely recommendations. This meticulous organization is fundamental to the seamless operation of the recommendation system, enabling it to deliver precise results.

4.6 Web Client

The web client constitutes the primary interface through which users interact with conference data. Using React.js for frontend development, along with CSS for styling and Chakra UI components for a visually appealing design, the client offers a seamless and intuitive user experience.

4.6.1 Folder Structure and Files

The React.js application adheres to a standard folder structure to ensure organization and maintainability. Upon initiation, the `client` folder was established as the project root. The React application was then initialized within this folder using the following command:

```
npx create-react-app client
```

This command uses `create-react-app`, a utility provided by the React team, to set up a new React project named `client` within the designated directory. It automatically configures the necessary files and dependencies, streamlining the setup process and allowing developers to focus on building the application.

The exact structure of the client folder displayed in the Figure 4.6.

Within the `client` directory, two main folders facilitate development:

- `src`: This directory serves as the core of the React application, housing all source code, including:
  - `components`: Reusable React components are stored here, contributing to the web client's structure and functionality. Components encapsulate specific UI elements or functional units, promoting modularity and code reuse.
  - `hooks`: Custom React hooks reside in this directory, facilitating specific functionalities across components. Hooks enable the extraction of component logic into reusable functions, enhancing code clarity and maintainability.
  - `pages`: Individual page components, representing different views or routes within the web client, are located here. Each page component corresponds to a distinct interface screen or navigation route, allowing for clear separation of concerns and improved navigation management.

  - `context`: Context providers, essential for enabling data sharing between components, are housed within this directory. Contexts provide a centralized location for storing and accessing shared data or state across components, promoting efficient communication and reducing prop drilling.

- `public`: This directory contains static assets such as HTML files and images, which are served to the client without undergoing any processing. The `public` folder serves as the entry point for the application, hosting the main HTML file (`index.html`) and other static resources required for initializing the client application.

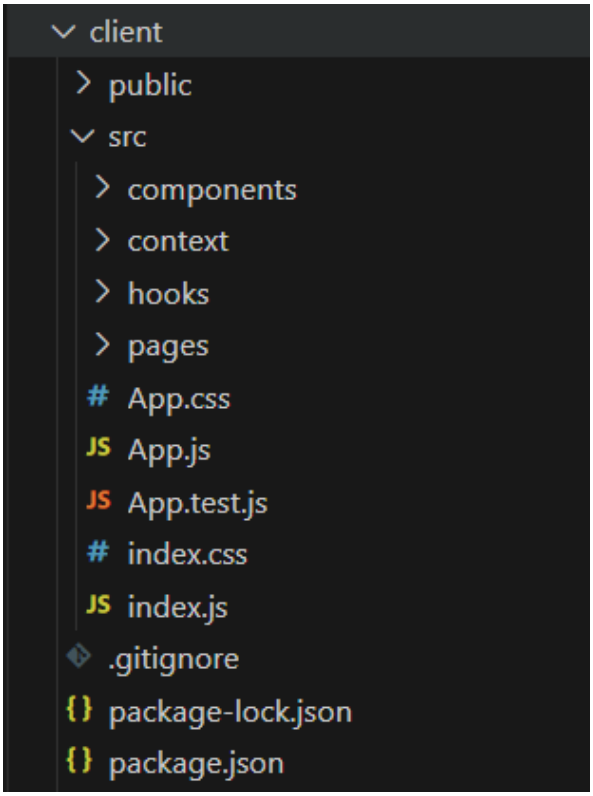

Figure 4.6: The client folder

The `index.js` file, located within the `src` directory, serves as the entry point for the React application. This file orchestrates the bootstrapping process, utilizing ReactDOM to render the root component of the application into the designated HTML element specified in `public/index.html`. By adhering to this structured approach, the web client ensures scalability, maintainability, and code organization, facilitating efficient development and enhancement of conference recommendation functionalities.

In the `App.js` file, the application utilizes React Router to manage client-side routing.

The `BrowserRouter` component encapsulates the entire application and enables declarative routing through the `Routes` component. Each `Route` corresponds to a specific URL path and renders a corresponding component when the path matches. The `Navigate` component provides conditional navigation based on user authentication status.

1. **"/":** Renders the `Select` component if the user is not authenticated, otherwise navigates to the home page.
2. **"/login":** Displays the `Login` component if the user is not logged in, otherwise redirects to the home page.
3. **"/register":** Renders the `Register` component for user registration if not logged in, otherwise redirects to the home page.
4. **"/home":** Displays the `Home` component if the user is authenticated, otherwise navigates to the root.
5. **"/conferences":** Renders the `ChooseFunctionality` component for conference-related actions if the user is logged in, otherwise redirects to the root.
6. **"/conferences/:acronym/:conferenceId":** Displays conference details using the `ConferenceDisplay` component if the user is logged in, otherwise redirects to the root.
7. **"/viewconferences":** Renders the `ViewConferences` component to view available conferences if the user is authenticated, otherwise navigates to the root.
8. **"/createconference":** Displays the `CreateConference` component to create a new conference if the user is logged in, otherwise redirects to the root.
9. **"/paper":** Renders the `Paper` component if the user is authenticated, otherwise navigates to the root.
10. **"/paper/:paperId":** Displays paper details using the `PaperReview` component if the user is logged in, otherwise redirects to the root.
11. **"/about":** Renders the `About` component for application information if the user is authenticated, otherwise navigates to the root.

12. **"/profile":** Displays the `Profile` component for user profile management if the user is logged in, otherwise redirects to the root.

13. **"/uploadreviewers":** Renders the `UploadReviewers` component for uploading reviewer details if the user is authenticated, otherwise navigates to the root.

14. **"/addreviewersexcel/:id":** Displays the `ReviewersExcel` component for adding reviewers via Excel if the user is logged in, otherwise redirects to the root.

15. **"/addreviewershand/:id":** Renders the `ReviewersHand` component for manually adding reviewers if the user is authenticated, otherwise navigates to the root.

16. **"/addpaper/:id":** Displays the `Paper` component for adding a new paper if the user is logged in, otherwise redirects to the root.

17. **"/addpaperexcel/:id":** Renders the `PaperExcel` component for adding papers via Excel if the user is authenticated, otherwise navigates to the root.

18. **"/addpapercsv/:id":** Displays the `PaperCsv` component for adding papers via CSV if the user is logged in, otherwise redirects to the root.

19. **"/addreviewerscsv/:id":** Renders the `ReviewersCsv` component for adding reviewers via CSV if the user is authenticated, otherwise navigates to the root.

20. **"/addpapernew/:id":** Displays the `PaperNew` component for adding a new paper if the user is logged in, otherwise redirects to the root.

21. **"/recommendation/:id":** Renders the `Recommendation` component for conference recommendations if the user is authenticated, otherwise navigates to the root.

22. **"/joinconference":** Displays the `JoinConference` component for joining conferences if the user is logged in, otherwise redirects to the root.

23. **"/conferencerecommendation/:id":** Renders the `ConferenceWithRecom` component for conference recommendations if the user is authenticated, otherwise navigates to the root.

The `BrowserRouter` component wraps the application, providing client-side routing capabilities. `Routes` define different paths and their corresponding components, while

`Route` components render the appropriate component based on the current URL. The `Navigate` component facilitates conditional redirection based on the user's authentication status. By employing React Router, the application maintains a structured navigation flow, enhancing user experience and overall usability.

4.6.2 hooks folder

It is important to refer to the `hooks` folder. In general, hooks [29] provide access to states for functional components while creating a React application. It allows you to use state and other React features without writing a class. Placing them in a dedicated directory allows for easy access and reuse across components throughout your application. The hooks that are inside the hook `hooks` are:

1. **useAuthContext**: provides access to the authentication context. It allows functional components to access the authentication state and dispatch actions related to authentication.

2. **useConference**: manages conference-related functionalities. It handles actions such as inserting conferences and retrieving conferences by author.

3. **useLogin**: manages the login functionality. It handles user authentication by sending a request to the server with provided credentials. Upon successful login, it updates the user's authentication status and stores user data in local storage.

4. **useLogout**: handles the logout process. It removes user authentication status and clears user data from local storage upon logout.

5. **usePaper**: manages paper-related functionalities. It handles actions such as inserting papers, retrieving papers by conference, and fetching paper details.

6. **useRecommendation**: manages conference recommendation functionality. It retrieves conference recommendations based on the provided conference ID and user token.

7. **useRegister**: handles user registration. It sends user registration data to the server and updates user authentication status upon successful registration.

8. **useReviewer**: manages reviewer-related functionalities. It handles actions such as inserting reviewers for conferences and retrieving reviewers by conference ID.

4.6.3 Components Folder

The `components` folder within the simple folder structure is straightforward, housing all the application's components [30]. At present, it comprises just two files: `Header.js` and `Footer.js`. However, as the project expands beyond 10-15 components, managing this folder can become challenging. Generally, components [31] serve as the foundation of any React project. This folder typically contains a collection of UI components such as buttons, modals, inputs, loaders, etc., which are utilized across different files within the project. In this particular project, `Header.js` encompasses the code for the web application's header, utilized on every page, while `Footer.js` contains the footer's code.

Generally, as the project grows, it's common to organize components into subfolders based on their functionality or purpose, such as `UI`, `Layout`, `Forms`, etc. This helps maintain a clear and structured codebase, making it easier to locate and manage components.

In summary, while the components folder may start small, it plays a crucial role in structuring and maintaining a React project. By following best practices for organization, naming, and documentation, developers can effectively manage the growing complexity of the components folder and ensure the scalability and maintainability of the project.

4.6.4 pages folder

In this thesis, the `pages` folder plays a crucial role in organizing the routes of the React application, ensuring that each file corresponds to a specific route, as illustrated in Figure 4.7. The pages folder is like a map for the project, showing where each part of the website or app can be found. It keeps everything organized, making it easier to manage and navigate through different sections of the application.

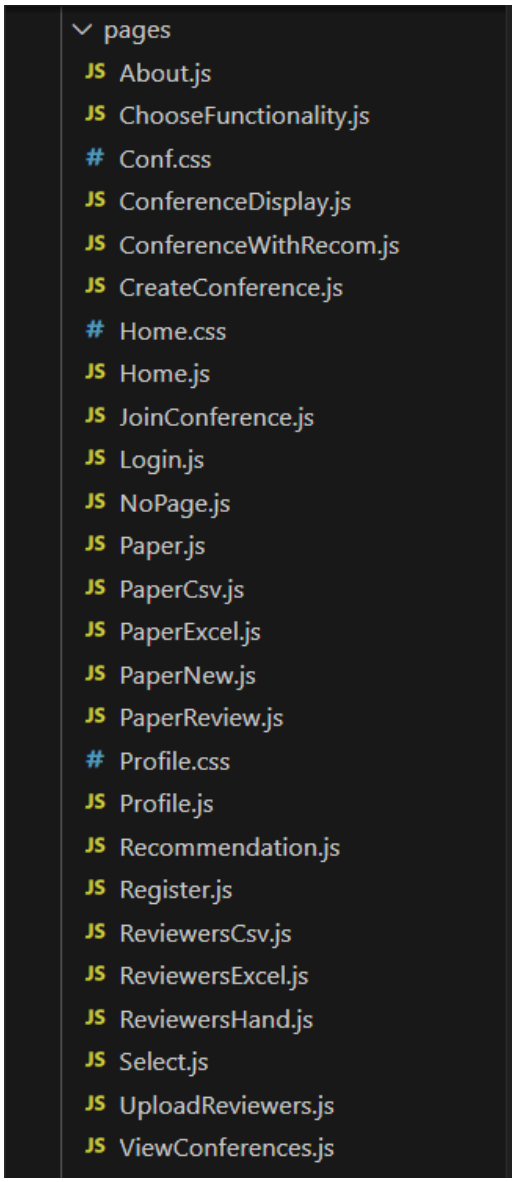

Figure 4.7: The pages folder

4.6.5 context folder

The `context` folder contains the `AuthContext.js` file, which defines a React context called **AuthContext** and a provider component named **AuthContextProvider**. This context provides state management for user authentication within the application. The **AuthContextProvider** utilizes a reducer to manage authentication state changes such as login and logout. Additionally, it uses the **useEffect** hook to persist user authentication status across page refreshes by storing user data in the **localStorage**. Finally, the **AuthContextProvider** component renders its children components, providing them access to the authentication state and dispatch function through the context's value prop.

Chapter 5

Graphical User Interface

To better illustrate the application's capabilities, it is important to present the client's graphical environment when the user uses the application through the browser. In more detail, the screens included in the client are as follows:

1. Initial page where the user can select the Login or the Register option as the (Figure 5.1) shows.

2. Register page (Figure 5.3) where the user can enter his credentials: first name, last name, email, username, and password for his account. After clicking the `Register` button, the user's account is created.

3. Login page (Figure 5.2) where the user can enter his username and password to log into the application. After successfully logging in, the app redirects him to the main page.

4. Home Page (Figure 5.4) for displaying important information for the user. On the main page, the user can read general information about the web application and its objectives. The user can also navigate to the page where he can select desired functionalities.

5. Selection of functionality page (Figure 5.5). Here, the user has three options: create a new conference, view a conference that he has created, and join another conference.

6. Page for creating a new conference (Figure 5.7). Here, the user can create a new conference by providing the application with the following details: author, title, acronym, webpage, city, country, first day, last day, primary area, secondary area, and organizer of the conference.

7. Page where the user can see the conferences that have been created (Figure 5.6). By clicking in their name can redirect to a page that shows the basic information for every conference.

8. Page that displays basic conference information (Figure 5.8): name, list of papers, and list of reviewers participating in the conference. This page features various buttons for different operations, including: uploading a single paper with abstract and title, uploading papers via CSV, uploading papers via Excel, uploading a single paper, uploading reviewers by name and email, uploading reviewers via Excel, uploading reviewers via CSV, and accessing recommendations.

9. Page where users can upload a single paper by providing the title and abstract (Figure 5.9).

10. Page where users can upload a CSV file containing information for new papers (title and abstract).

11. Page where users can upload an Excel file containing information for new papers (title and abstract) (Figure 5.11).

12. Page where users can upload a PDF file (scientific paper), and the application extracts the title and abstract from it (Figure 5.10).

13. Page where users can upload a CSV file containing information for new reviewers (name and email) (Figure 5.14).

14. Page where users can upload an Excel file containing information for new reviewers (name and email) (Figure 5.12).

15. Page where users can access recommendations for each reviewer. Each reviewer will be assigned three papers to review, as generated by the integrated recommendation system within the web application. Users have the option to export the displayed results by clicking either the `Export the assignments in CSV` button for a CSV file or the `Export the assignments in Excel` button for an Excel file. Also, they can export in Excel the papers of the conference with the button `Download the paper with IDs in Excel`.

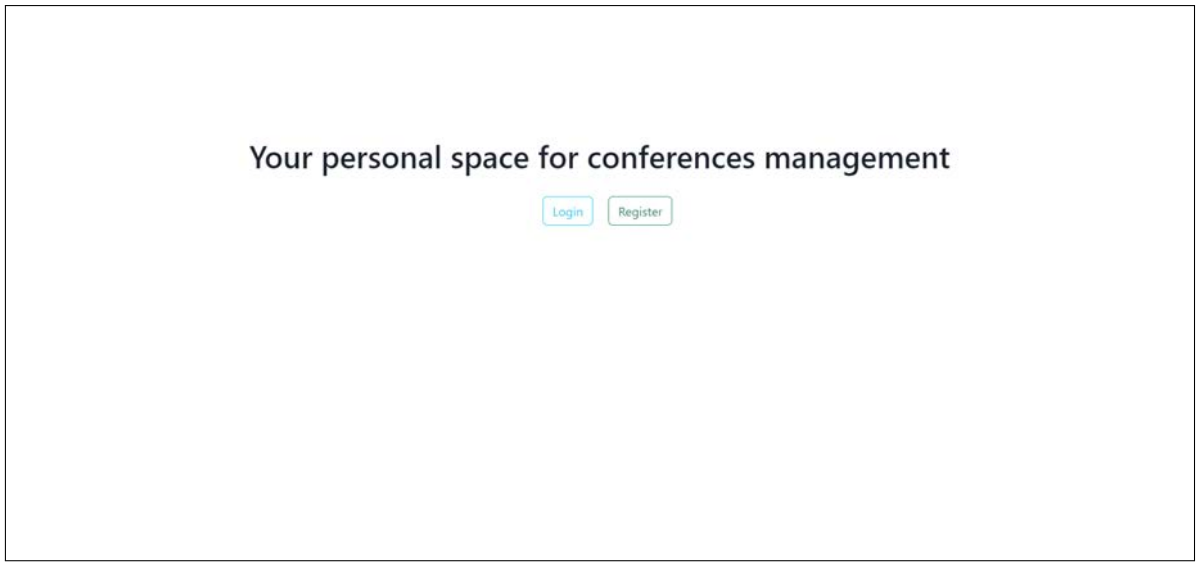

Figure 5.1: The first page of the app

Users have the option to either Login or Register. The Registration page prompts users to input their first name, last name, email, username, and password to set up a new account. Figure 5.1 represents the `/` route, while Figure 5.2 corresponds to the `/login` route, and Figure 5.3 to the `/register` route. Both the Register and Login pages feature buttons allowing users to easily navigate between them if needed.

Figure 5.2: The login page

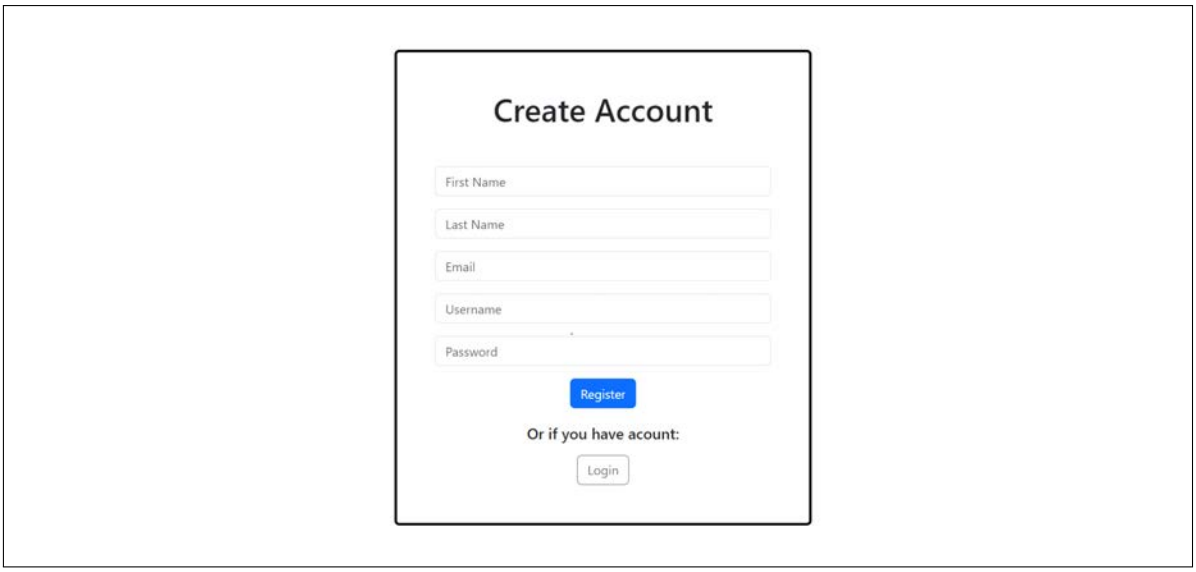

Figure 5.3: The register page

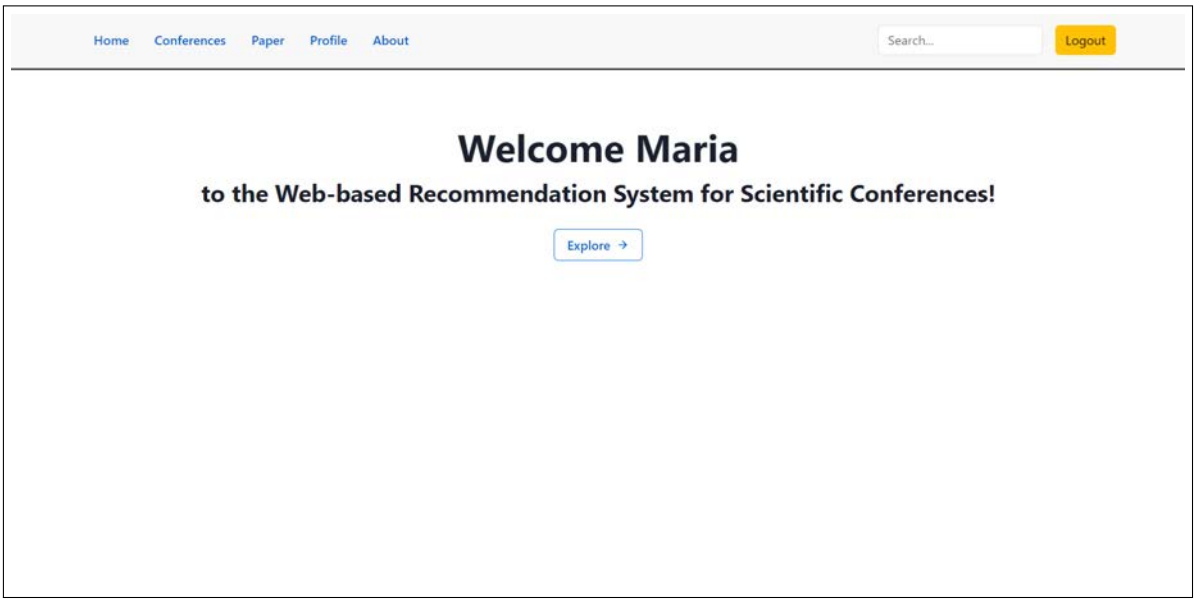

Figure 5.4: The initial page of the app (home page), where basic information about it exists

The homepage's basic features (5.4) are easy to spot for readers. In the header, there are several buttons: `Home`, `Conferences`, `Paper`, `Profile`, `About`, and `Logout`. Each button directs the user to different routes and pages. For instance, the Home button leads to the homepage, Conferences takes the user to the `/conferences` page where they can explore conference-related activities, Profile directs them to `/profile`, and Logout logs them out of the app. Additionally, the Explore button at the center of the page also redirects the user to the `/conferences` route.

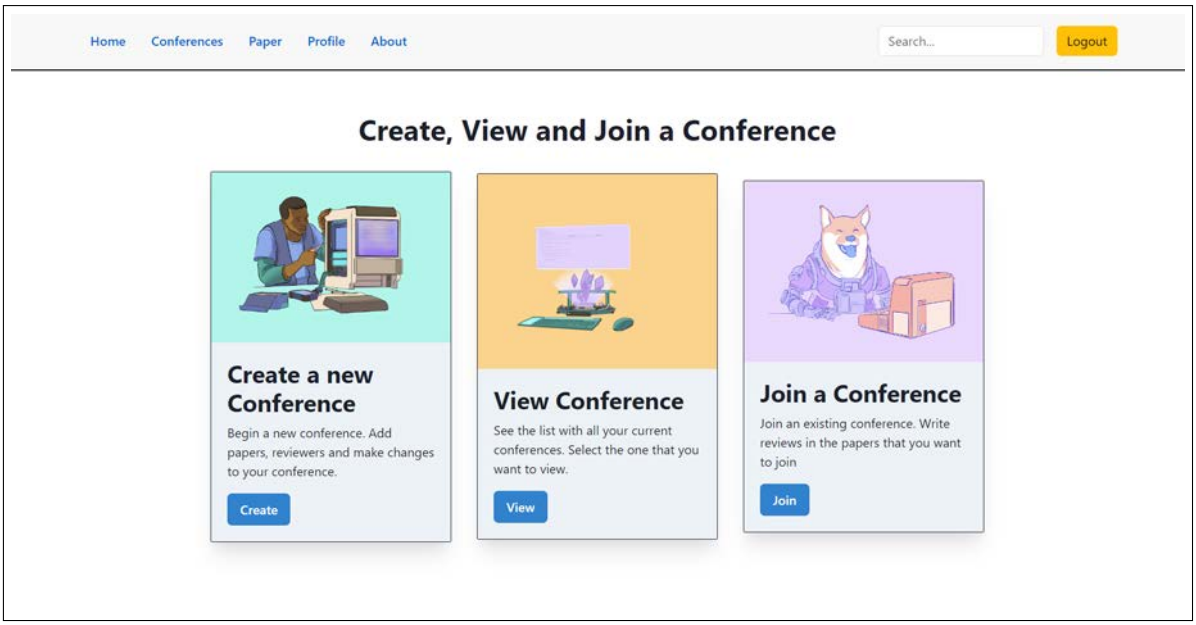

Figure 5.5: The page where the user can select what functionality wants

In Figure 5.5, readers can observe the options available to users within the application. Users can initiate a new conference by clicking the `Create` button and redirect to page that displayed in Figure 5.7, review conferences they have already created via the `View` button, which will redirect the user in the route `/viewconferences` (Figure 5.6), and participate in other conferences by selecting the `Join` button.

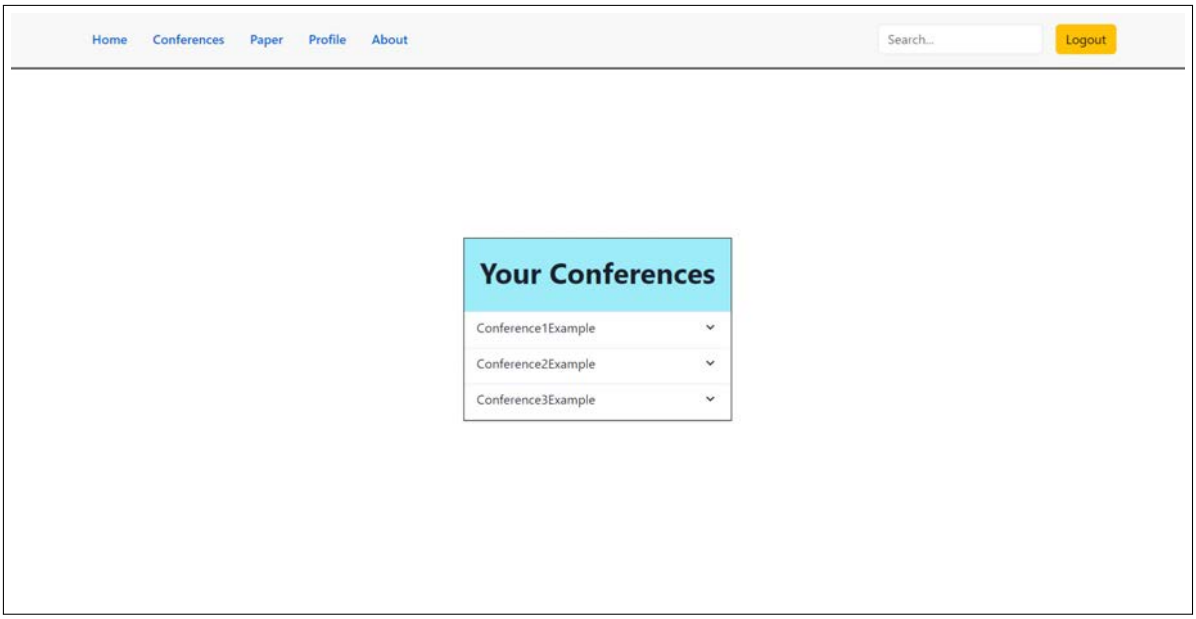

Figure 5.6: The page where the user can view the personal conferences. The names of conferences here is some test names

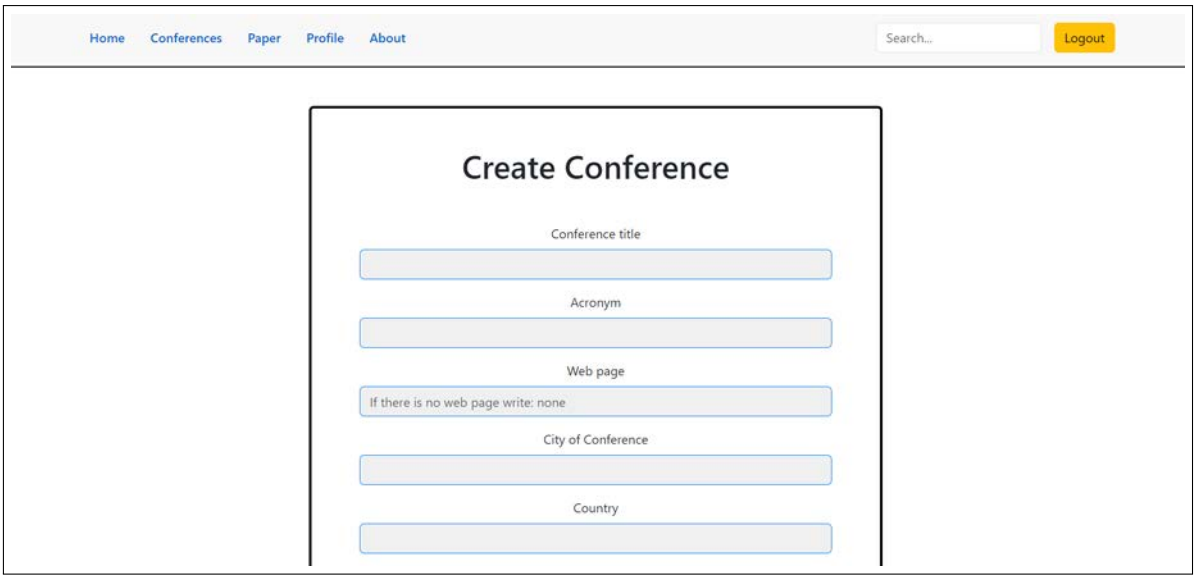

Figure 5.7: The page where the user create a new conference

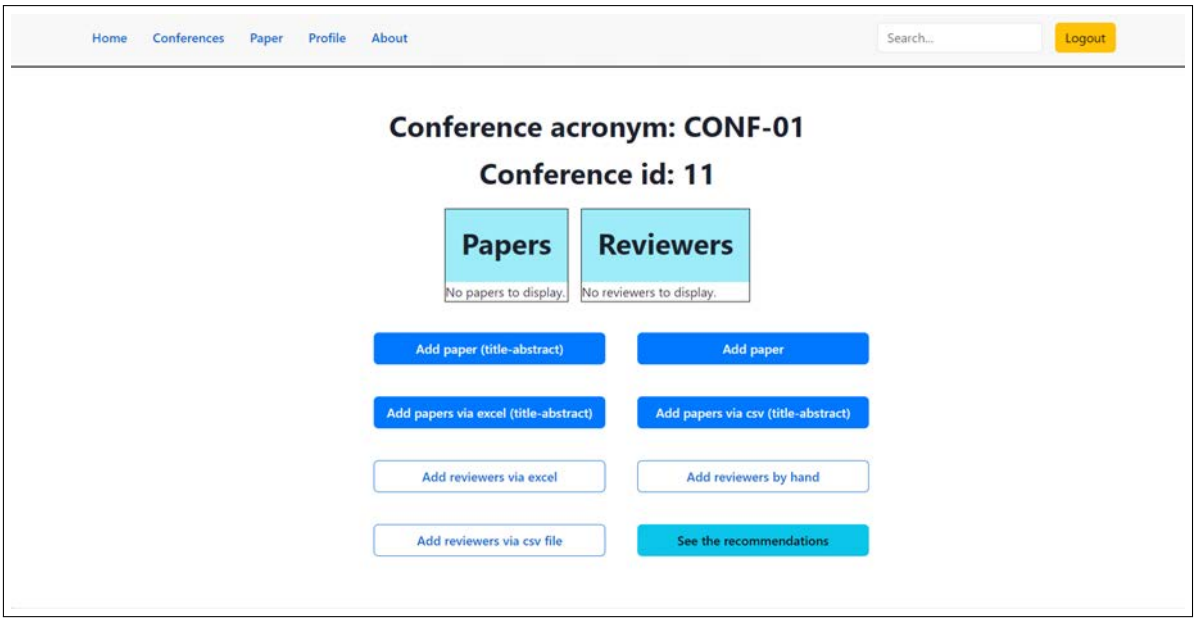

Figure 5.8: The page where the user can see the papers and the reviewers of his conference

Figure 5.8 corresponds to the route `/conferences/:acronym/:conferenceId`, where users can access papers and reviewers linked to their conference and manage various conference-related tasks, including adding papers or reviewers. Clicking the `Add paper (title-abstract)` button redirects the user to the `/addpaper/:id` route (Figure 5.9). Alternatively, clicking the 'Add paper' button redirects them to the `/addpapernew/:id` route (Figure 5.10). Lastly, clicking the `Add papers via excel (title-abstract)` button (Figure 5.11) initiates the process of adding papers via an Excel file.

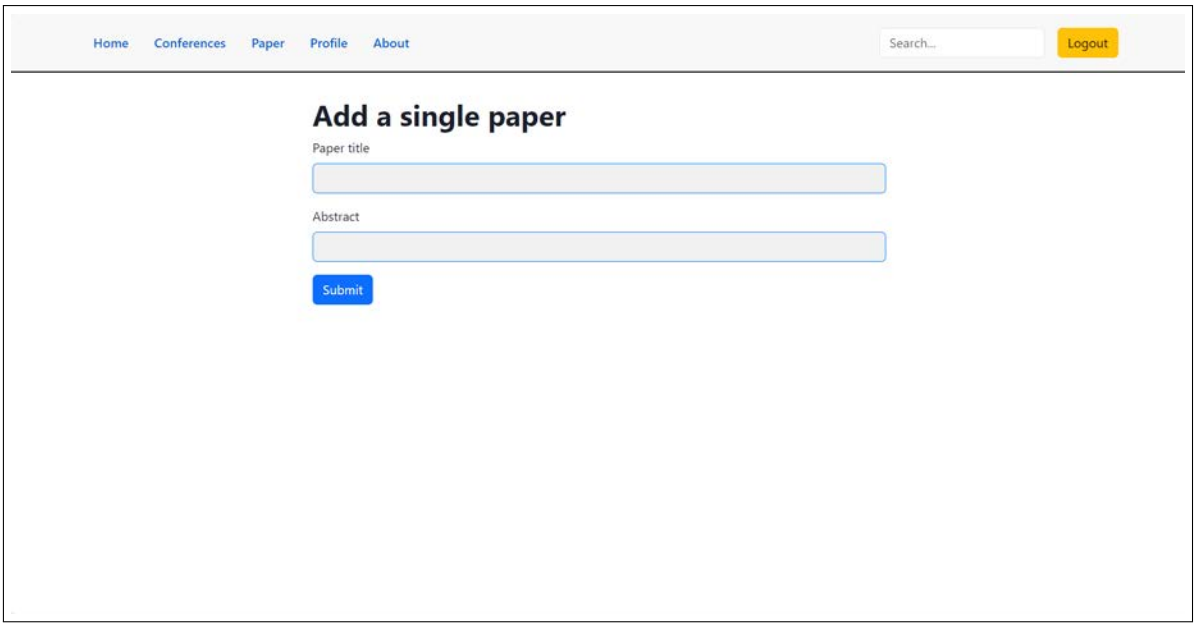

Figure 5.9: The page where the user add a single paper, giving the title and the abstact

In Figure 5.9, is presented the page where the users can seamlessly add a single paper to their conference by entering both the title and abstract. Figure 5.10 showcases a different approach, where users simply upload the paper, and the system handles the rest autonomously. Lastly, in Figure 5.11, users are provided with a convenient option to upload multiple papers simultaneously using an Excel file containing titles and abstracts.

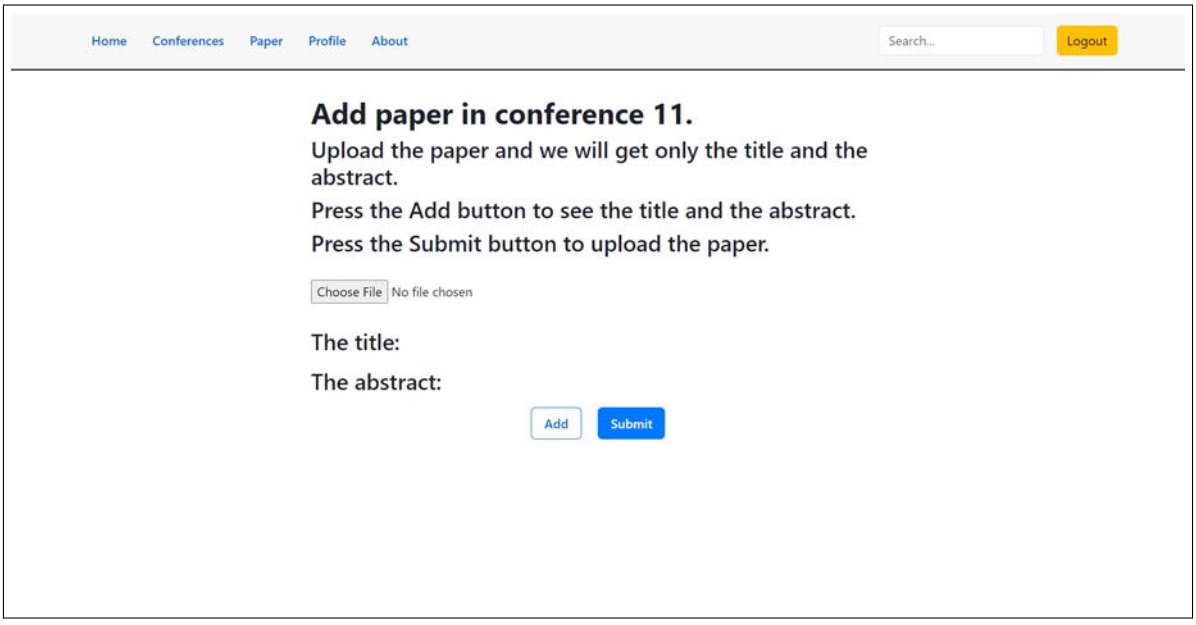

Figure 5.10: The page where the user can upload his paper

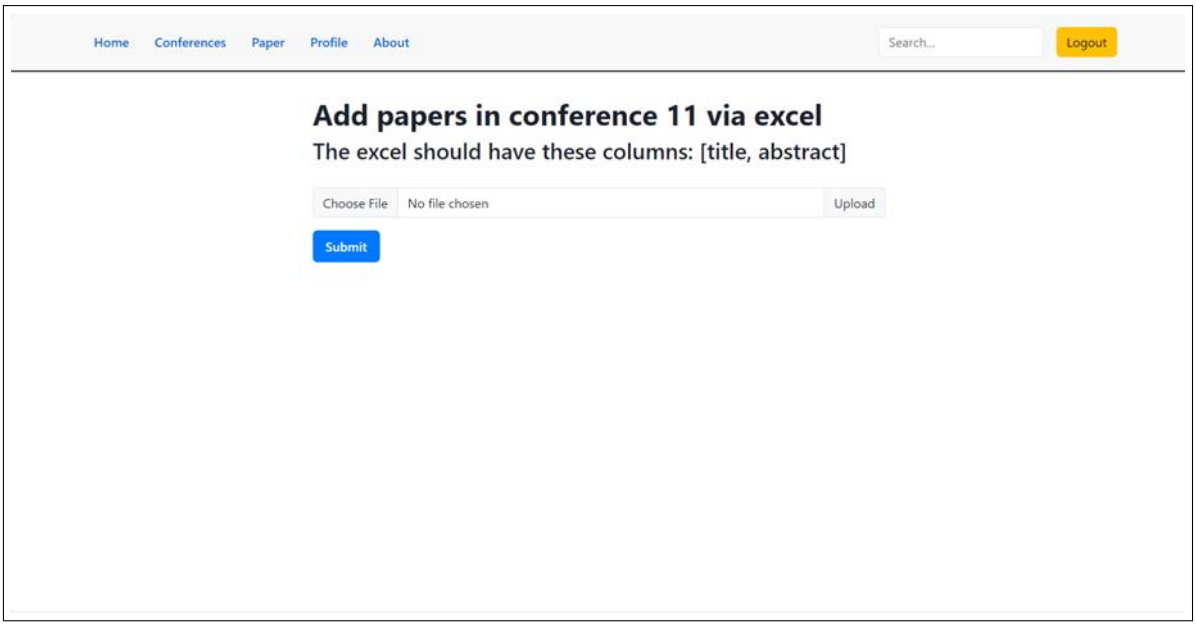

Figure 5.11: The page where the user upload multiple papers with their title and abstract, via Excel

In Figure 5.12, users are given the flexibility to upload multiple reviewers for their scientific conference in one go, using an Excel file that includes their names and titles. Figure 5.13 demonstrates an alternative method, where users can swiftly add reviewers to the database by providing both their names and email addresses. Also, there is the option to add the reviewers with the help of a CSV file, as the 5.13 depicts.

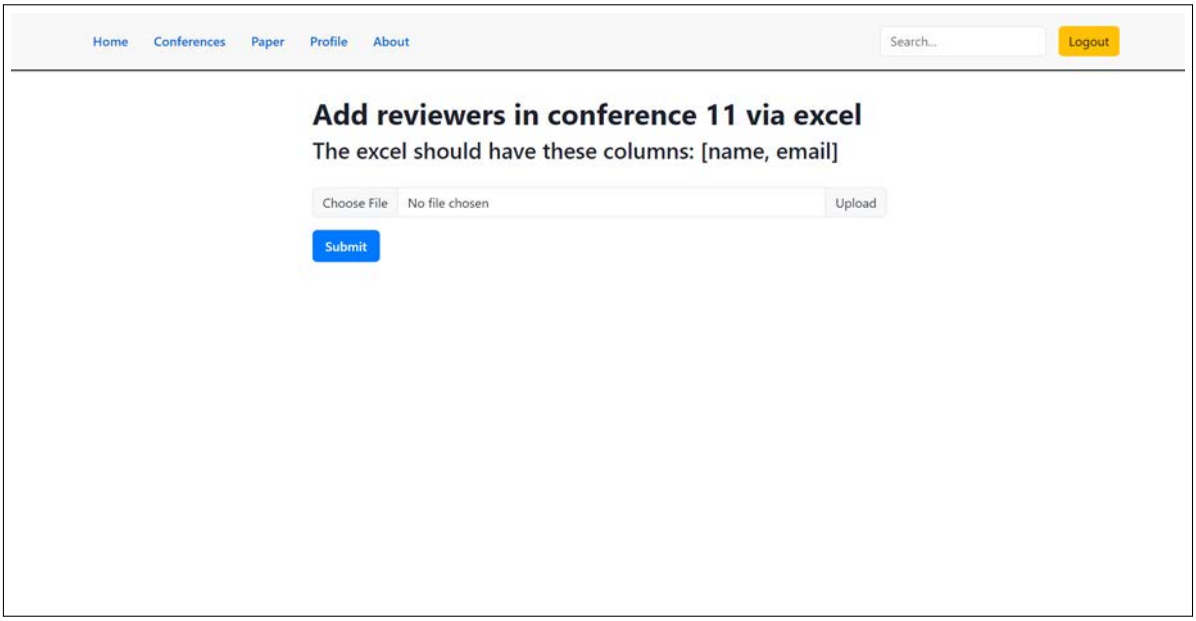

Figure 5.12: The page where users can upload multiple reviewers at once through an Excel file containing their names and emails.

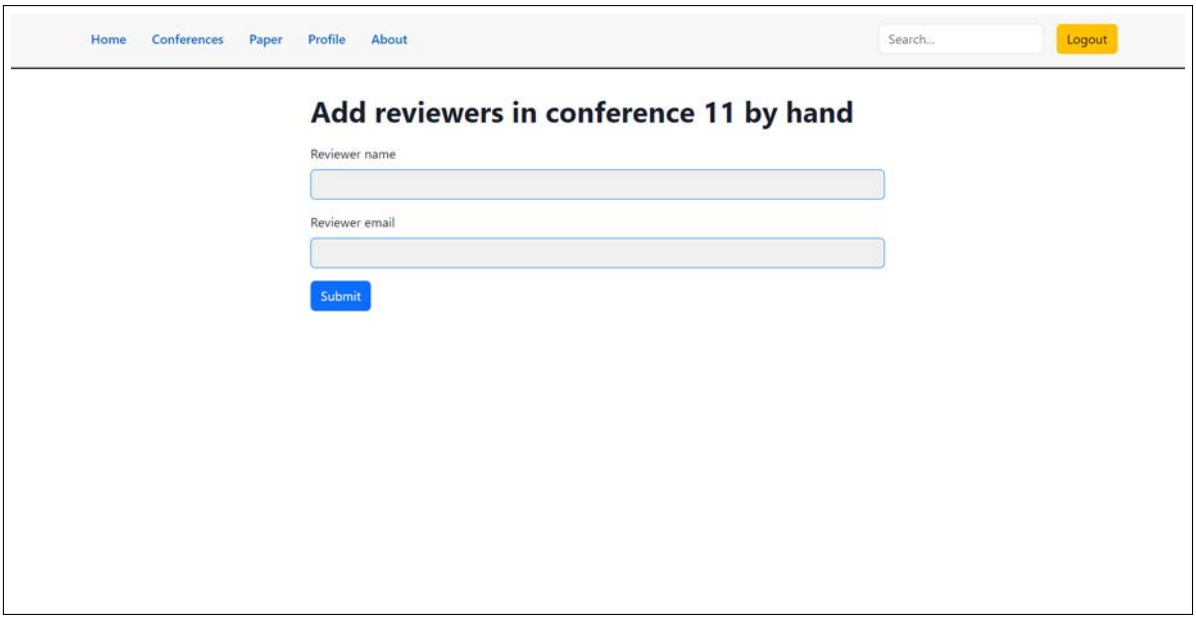

Figure 5.13: The page where the user can upload reviewers with their name and email by hand one each time

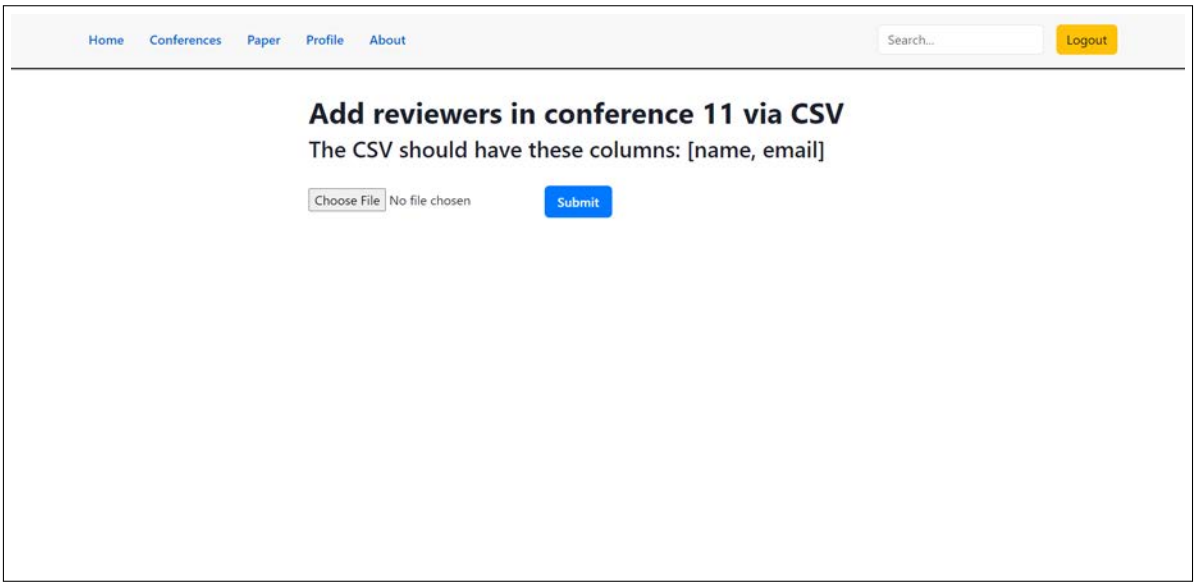

Figure 5.14: The page where the user can upload multiple reviewers with their name and email via a CSV file

Chapter 6

Usage example

This chapter demonstrates the functionality of the application by showing how a user can log in, create a conference, add papers and reviewers, and view the recommendations. To test our system, we will create a user account. The user will be named `Spiros Kavvathas` (first name: `Spiros`, last name: `Kavvathas`) , and the username will be `skavvathas`, as shown in Figure 6.1.

The test conference will include 70 reviewers and 210 papers sourced from the AMiner database [28]. Specifically, the data comes from the DBLP-Citation-network V13 [28], which contains 5,354,309 papers and 48,227,950 citation relationships. The data used here have been modified and saved in CSV files.

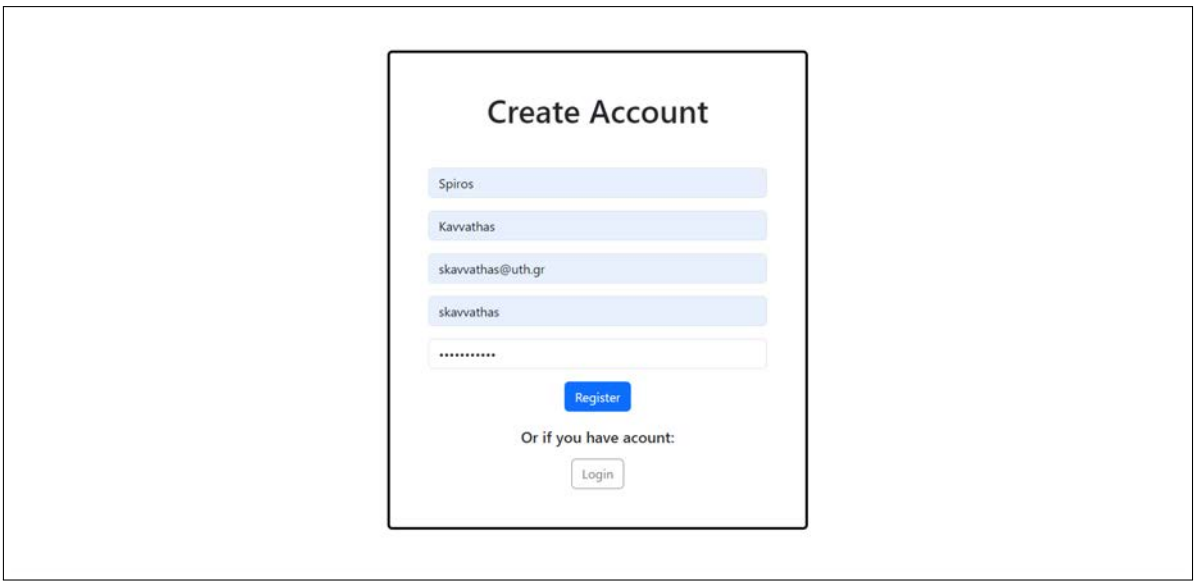

Figure 6.1: Conference form example

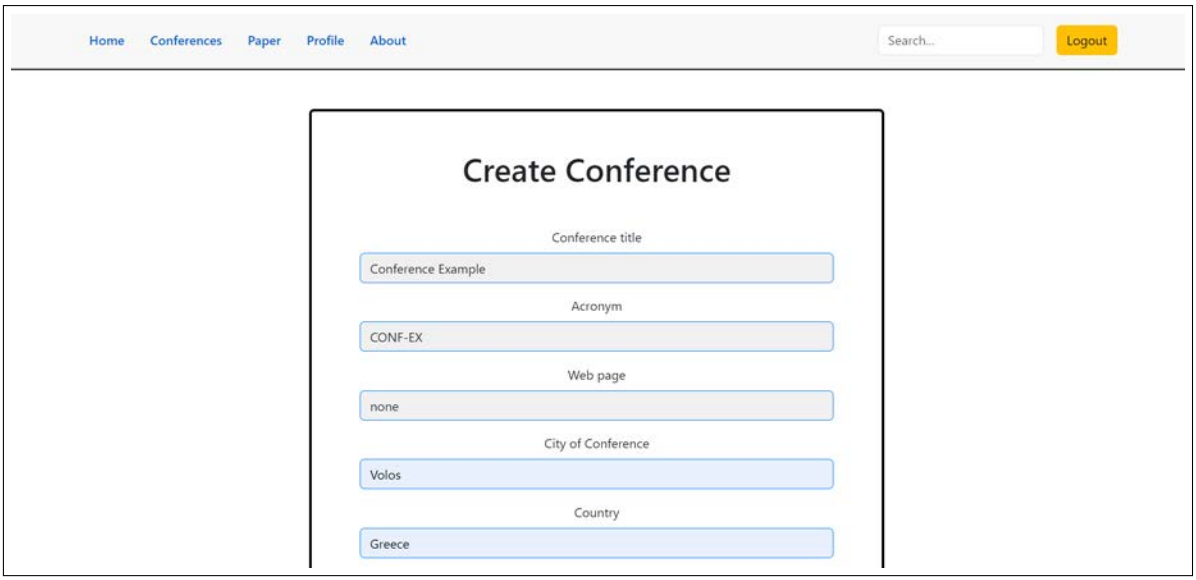

Figure 6.2: Example of creation of a conference

The conference that is created from the user (`Spiros Kavvathas`) has the following details: The conference is titled `Conference Example` with the acronym `CONF-EX` as the Figure 6.2 shows. It will take place in Volos, Greece, from July 2, 2024, to July 10, 2024. The primary area of focus will be Engineering, with a secondary area designated as Mathematics and Statistics. The event is organized by the University of Thessaly. The conference does not have a specified webpage at this time and has 'none' listed for this detail. After the creation, the user can view the basic information about the conference and explore the available functionalities within the platform, as illustrated in Figure 6.3.

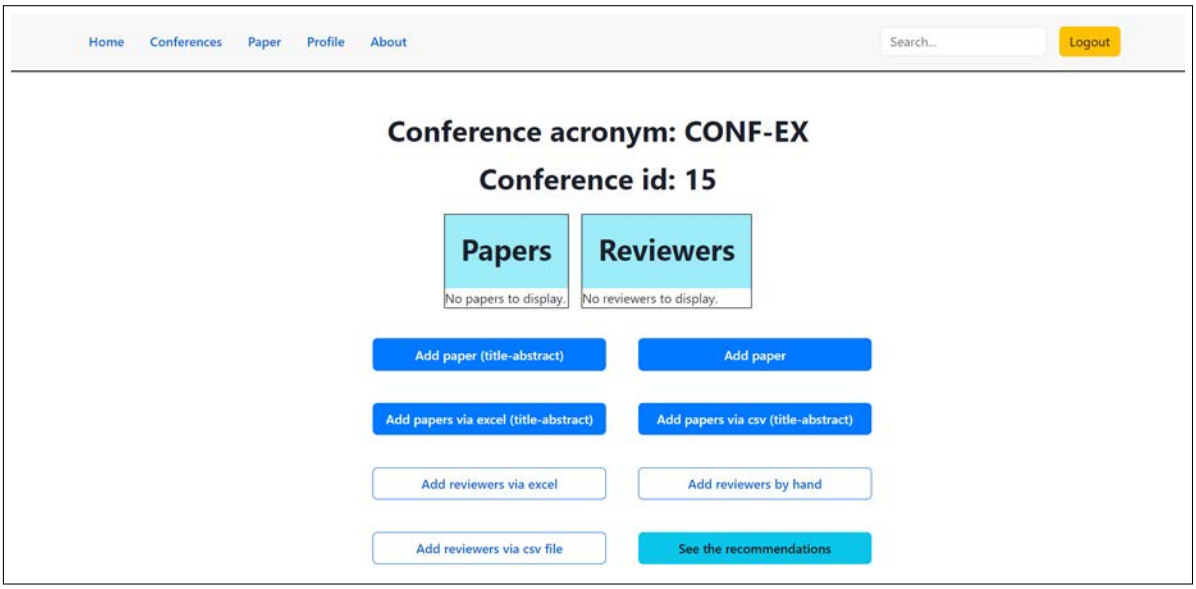

Figure 6.3: The conference before the upload of papers and reviewers

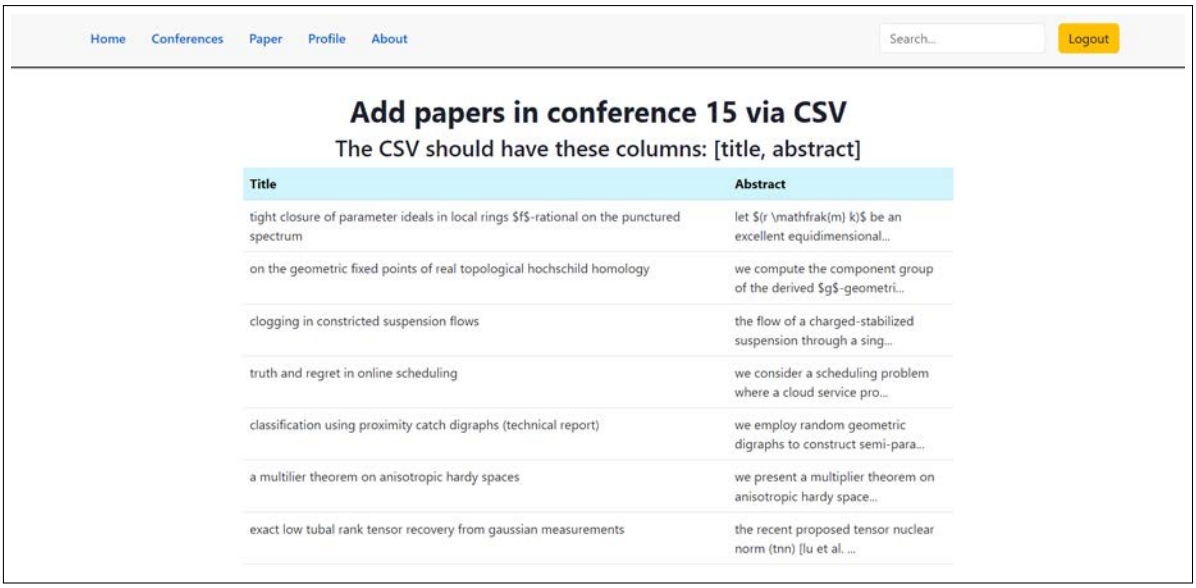

Figure 6.4: The upload of paper_test.csv file.

The data to be uploaded is in CSV format, because it was easy to integrate them, consisting of two files named `reviewers_test.csv` and `papers_test.csv`. Figure 6.4 illustrates how the application displays paper titles and abstracts immediately after they are uploaded but before they are submitted and saved in the database. Similarly, Figure 6.5 demonstrates the interface displaying reviewer information (name and email).

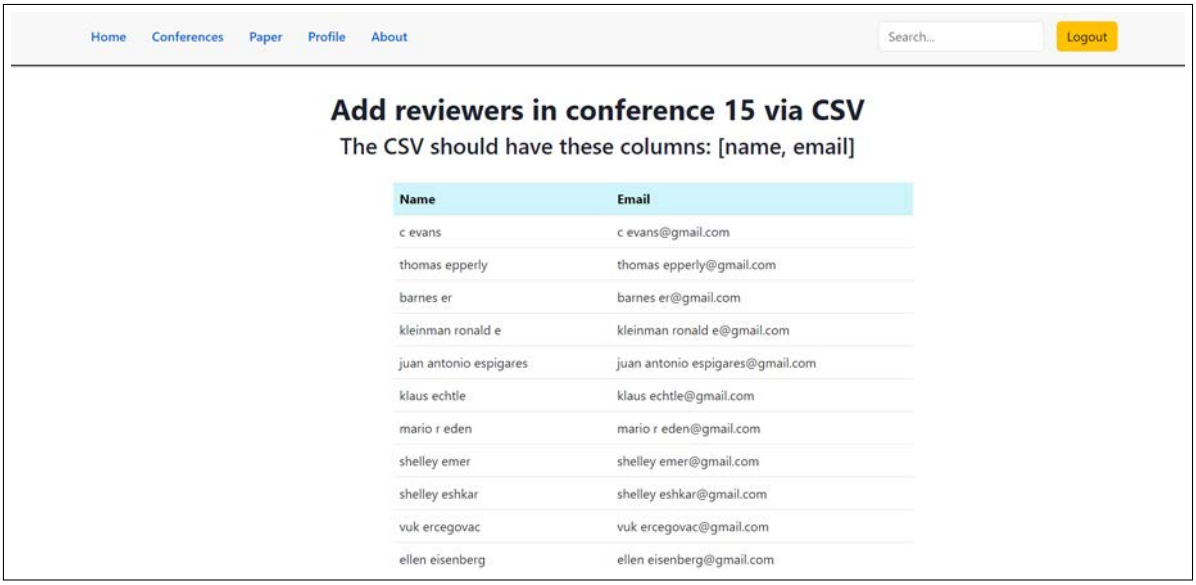

Figure 6.5: The upload of reviewers_test.csv file.

After uploading papers and assigning reviewers for the conference, users can view recommendations by clicking the `See Recommendations` button, as shown in the bottom right of Figure 6.3. These recommendations are generated based on the uploaded data. Upon clicking the button, the recommendation system begins processing the data, a procedure explained in detail in Section 4.5. The assignment results for this example are illustrated in Figure 6.6. After reviewing the recommendations, users have the option to export them either as a CSV file or in XLSX format for further analysis or sharing (Figure 6.7). It is worth mentioning, that the recommendation system generate the results after 31 minutes.

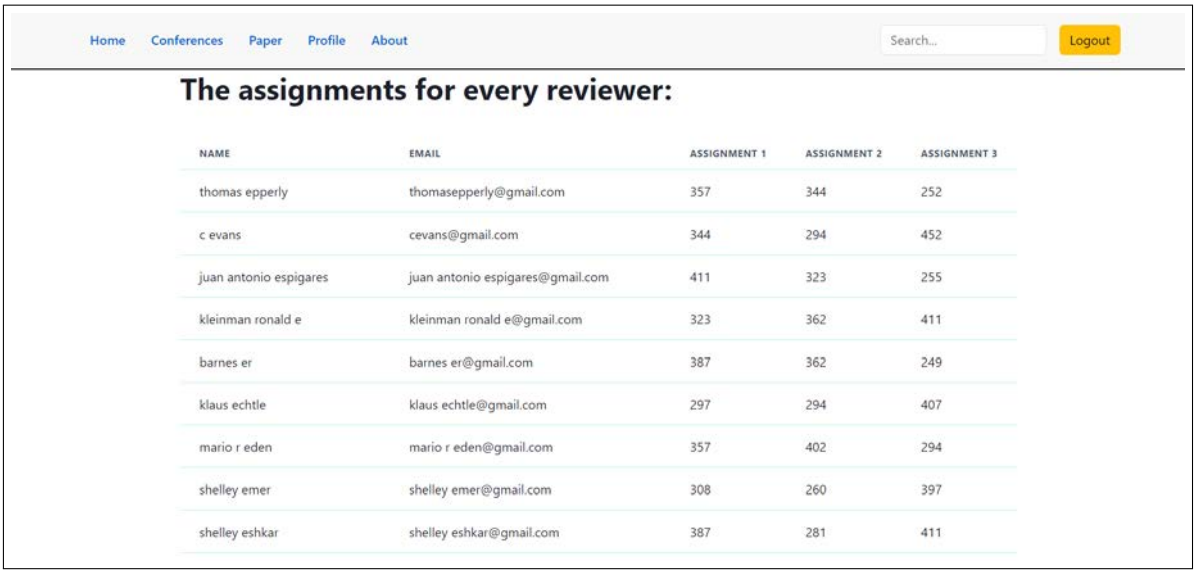

Figure 6.6: The assignment results from the recommendation system.

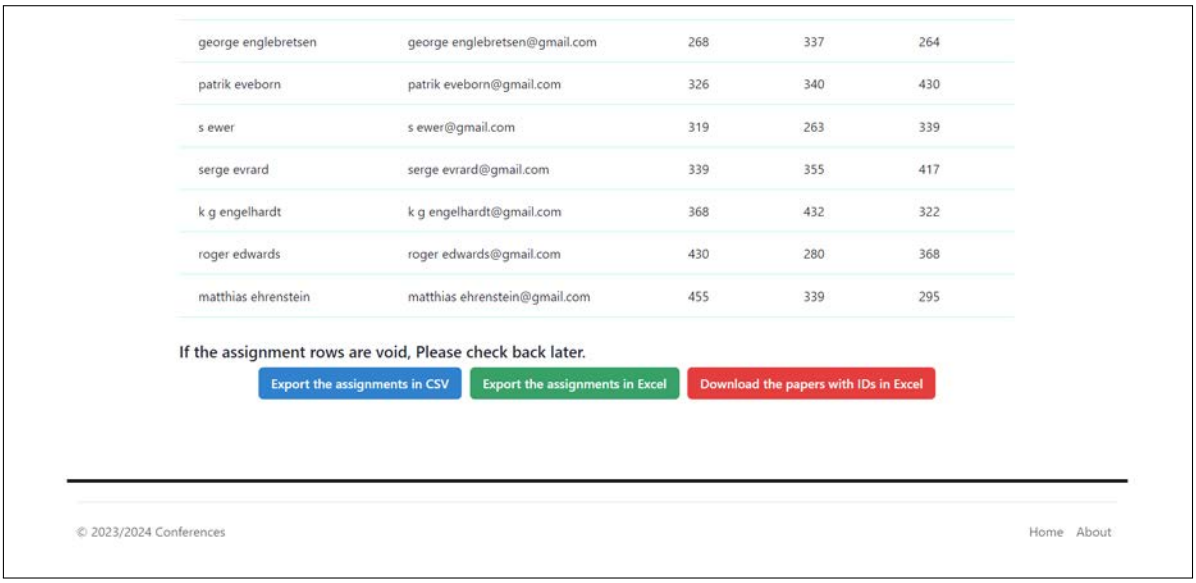

Figure 6.7: The user can download both the assignments and the paper IDs.

Figure 6.8: The recommendation system when starts to create the profiles.

When the user clicks the button `See Recommendations` button the recommendation system starts the process of creating profiles for every user as the Figure 6.8 shows. The system continuously monitors the start_recommendation column, repeatedly checking if its value is set to 1. Also, the Figure 6.8 shows the connections that has been established between the database and the system. This process of the recommendation system starts when the user type `python main.py`.

Chapter 7

Conclusions

7.1 Summary and Conclusions

Summarizing, the work accomplished in this thesis is expected to be highly beneficial to professors and individuals seeking simplified management of scientific conferences and paper assignments. The web application represents a comprehensive full-stack development effort, with both backend and frontend components implemented from scratch. Users are anticipated to have a seamless experience within the application. Feedback from users suggests that the overall experience is positive, with the application's flow enhancing the efficiency and effectiveness of the paper assignment process.

Implementing the web-based system was a challenging task, but the results are highly satisfactory. Developing the application using React.js, Express.js, Node.js, SQL, and MySQL presented significant difficulties. However, these challenges were mitigated through careful and detailed study of the documentation for each framework and library. This thorough understanding allowed us to overcome the complexities and create a highly functional and user-friendly application. The process required extensive research and problem-solving, particularly when integrating various components and ensuring seamless interaction between the frontend and backend. Despite these hurdles, the end result is a robust and efficient system that meets the needs of its users.

7.2 Future work

While the present study has provided valuable insights, there are avenues for further exploration and improvement. The web application has a big room for different improvements and approaches. Firstly, easily can be added the functionality of author rating and commenting on the paper that the user has add in the conference. In this scenario, after a specific paper has been assigned, the author can add their review of the paper, which will then be added to the database (new table can be added).

Additionally, one significant area for enhancement is the application's user interface (UI). Future development efforts could prioritize refining the CSS to create a more visually appealing and user-friendly experience. Incorporating a greater number of Chakra UI components can streamline the design process and ensure consistency throughout the application. Furthermore, exploring various styling libraries, such as Bootstrap, Material UI, Blueprint UI, and Ant Design, offers opportunities to leverage their unique features and strengths. By doing so, we can improve the aesthetics, functionality, and overall user experience of the application. Adopting these libraries could also provide access to a broader range of pre-designed components and tools, making the UI development process more efficient and effective.

Also, different possibilities could be added to the recommendation system. Users could be given the option to change the recommendations if they find them unsuitable (for instance, if a reviewer is recommended a paper that is not appropriate for them). Additionally, functionality could be added to allow users to make assignments manually, rather than relying solely on the recommendation system. These features would provide greater flexibility and control for users, enhancing the overall effectiveness of the application.

Additionally, if the application receives positive feedback from a large number of users, it could be published online. This would involve integrating a web hosting service to make the application accessible to more people. To accomplish this, several steps and tools can be used.

To accomplish this, several steps and tools can be used. First, a web hosting service can be chosen, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure for robust and scalable hosting solutions. For simpler needs, services like Heroku, Netlify, or Vercel, which are user-friendly and suitable for hosting web applications, can be considered. Next, a domain name that reflects the purpose of the web application can be set up using domain registration services like GoDaddy, Namecheap, or Google Domains. To

deploy the application, a deployment platform can be used to upload the web application to the hosting service; platforms like GitHub provide integration with many hosting services, allowing for seamless deployment. The application's database should also be hosted online using managed database solutions such as Amazon RDS, Google Cloud SQL, or Azure SQL Database, or services like MongoDB Atlas for cloud-hosted databases can be utilized. SSL (Secure Sockets Layer) should be implemented to secure data transmission between users and the server; most hosting services offer easy ways to obtain and install SSL certificates. Tools provided by the hosting service, such as AWS CloudWatch, Google Stackdriver, and Azure Monitor, can be used to monitor the performance of the application and scale resources as needed. Finally, tools like Google Analytics can be integrated to gather user feedback and monitor how users interact with the application, helping to understand areas for improvement and how users benefit from the application.

Bibliography

[1] Vaios Stergiopoulos, Michael Vassilakopoulos, Eleni Tousidou, and Antonio Corral. An academic recommender system on large citation data based on clustering, graph-modeling and deep-learning, Published: 18 April 2024. Data Structuring Eng. Lab., Dept. of Electrical Computer Engineering, University of Thessaly, Volos, Greece.

[2] Don Conry, Yehuda Koren, and Naren Ramakrishnan. Recommender systems for the conference paper assignment problem, 22 June 2009. Department of Computer Science, Virginia Tech; Yahoo! Research, Israel.

[3] Judy Goldsmith and Robert H. Sloan. The ai conference paper assignment problem, 20 June 2007. Department of Computer Science, University of Illinois-Chicago.

[4] COMS. COMS - Conference Management Software. `https://conference-service.com/index.html`. (Date accessed: 22-03-2024).

[5] EasyChair. `https://easychair.org/`. (Date accessed: 23-03-2024).

[6] Converia - Conference Management Software. `https://www.converia.de/en/`. (Date accessed: 23-03-2024).

[7] TrustRadius. Converia - Reviews. `https://www.converia.de/en//`. (Date accessed: 26-03-2024).

[8] Citation Network Dataset. `https://www.aminer.org/citation`. (Date accessed: 29-03-2024).

[9] Meshal Alfarhood and Jianlin Cheng. CATA++: A Collaborative Dual Attentive Autoencoder Method for Recommending Scientific Articles , 2020.

[10] Intelivita. Web Applications Types — Examples of Web Apps + Benefits and Challenges. https://www.intelivita.com/blog/types-of-web-applications/. (Date accessed: 30-03-2024).

[11] MDN Web Docs. Progressive web apps. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps. (Date accessed: 02-04-2024).

[12] Zappier. The 6 best eCommerce website building platforms for online stores in 2024. https://zapier.com/blog/best-ecommerce-shopping-cart-software/. (Date accessed: 05-04-2024).

[13] Optimizely. Content management system. https://www.optimizely.com/optimization-glossary/content-management-system/. (Date accessed: 06-04-2024).

[14] BigCommerce essentials. What is JavaScript, and why is it important? https://www.bigcommerce.com/glossary/javascript/. (Date accessed: 06-04-2024).

[15] Javapoint. What is CSS. https://www.javatpoint.com/what-is-css. (Date accessed: 15-04-2024).

[16] React. React documentation. https://react.dev/learn. (Date accessed: 27-04-2024).

[17] freeCodeCamp. JSX in React – Explained with Examples. https://www.freecodecamp.org/news/jsx-in-react-introduction/. (Date accessed: 28-04-2024).

[18] Medium. Virtual DOM and Real DOM: Understanding the Differences. https://medium.com/@surksha8/virtual-dom-and-real-dom-understanding-the-differences-da8f3fab4261. (Date accessed: 30-04-2024).

[19] Chakra UI. https://v2.chakra-ui.com/. (Date accessed: 07-05-2024).

[20] BRAINHUB. What Is Node JS Used For? - 5 Top Implementations. https://brainhub.eu/library/what-is-nodejs-used-for. (Date accessed: 09-05-2024).

[21] Kinsta. What Is Node.js and Why You Should Use It. https://kinsta.com/knowledgebase/what-is-node-js/. (Date accessed: 09-05-2024).

[22] Express.js. https://expressjs.com/. (Date accessed: 10-05-2024).

[23] MDN Web Docs. Express/Node introduction. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. (Date accessed: 10-05-2024).

[24] talend. What is MySQL? Everything You Need to Know. https://www.talend.com/resources/what-is-mysql/. (Date accessed: 12-05-2024).

[25] Authgear. Web Application Authentication: How It Works and How to Implement It. https://www.authgear.com/post/web-application-authentication-guide#definition. (Date accessed: 13-05-2024).

[26] LogicMonitor. What Are JSON Web Tokens? https://www.logicmonitor.com/blog/what-are-json-web-tokens. (Date accessed: 13-05-2024).

[27] Hugging Face. BERT. https://huggingface.co/docs/transformers/model_doc/bert. (Date accessed: 15-06-2024).

[28] AMiner. Citation Network Dataset. https://www.aminer.cn/citation. (Date accessed: 11-06-2024).

[29] GeeksForGeeks. Folder Structure for a React JS Project. https://www.geeksforgeeks.org/folder-structure-for-a-react-js-project/. (Date accessed: 13-05-2024).

[30] Web Dev Blog. How To Structure React Projects From Beginner To Advanced. https://blog.webdevsimplified.com/2022-07/react-folder-structure/. (Date accessed: 13-05-2024).

[31] Xenonstack. Understanding Reactjs Project Structure and Best Practices. https://www.xenonstack.com/insights/reactjs-project-structure. (Date accessed: 15-05-2024).

APPENDICES

Appendix A

Code

The code that was created for the presented thesis can be downloaded after communication with the author and can be found in the link :

`https://github.com/skavvathas/Conferences-Web-App`

The code of the recommendation system can be retrieved after communication with Vaios Stergiopoulos.

The git repository includes two folders, one for the client and another for the server. Additionally, it contains a README file providing analytical guidance to users, and a schemas.sql file containing the database schemas. This file can be utilized by users to set up a local MySQL server.