PONDICHERRY UNIVERSITY

(A Central University)

SCHOOL OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE

MASTER OF COMPUTER APPLICATION

NAME : ANJANA KUMARI
REGISTER NO. : 22352008

COURSE : MCA

SUBJECT : Faculty Data Collection

GUIDED BY : Dr. P Sujatha

FACULTY DATA COLLECTION

By

Anjana Kumari

(Registration Number: 22352008)

Project report submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATION

DEPARTMENT OF COMPUTER SCIENCE SCHOOL OF ENGINEERING & TECHNOLOGY

PONDICHERRY UNIVERSITY MAY 2024

BONAFIDE CERTIFICATE

This is to certify that this project work entitled " **FACULTY DATA COLLECTION**" is a bonafide record of work done by **ANJANA KUMARI** (Reg. Number 22352008) in the partial fulfilment for the degree of **Master of Computer Applications** of Pondicherry University.

This work has not been submitted elsewhere for the award of any other degree to the best of our knowledge.

INTERNAL GUIDE

DR. P SUJATHA
Professor
Department of Computer Science
School of Engineering & Technology
Pondicherry University
Pondicherry – 605 014

HEAD OF THE DEPARTMENT

Dr. S. K. V. JAYAKUMAR
Professor/HOD
Department of Computer Science
School of Engineering & Technology
Pondicherry University
Pondicherry – 605 014

Submitted for the Viva-Voce Examination held on:

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Every project, big or small, is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration, support and guidance of all those people who have been instrumental in making this project a success.

I express my heartfelt gratitude to my Project Guide **Dr. P SUJATHA**, Professor, Department of Computer Science, Pondicherry University, Pondicherry, for her whole - hearted assistance and direction not only for the duration of the project but for the entire duration of the course. I will always remain grateful to her and whose constant care about me has provided a new direction to work.

I express my gratitude to **Dr. S. K. V. JAYAKUMAR**, Professor and Head, Department of Computer Science, Pondicherry University, Pondicherry for his support and arranging the project in a good schedule.

Finally, I would like to express my regards for all the faculty members of Department of Computer Science and others involved in this project, directly or indirectly

(ANJANA KUMARI)

TABLE OF CONTENTS

5. IMPLEMENTATION

6. **SYSTEM TESTING**

6.1 SYSTEM IMPLEMENTATION

6.2 TESTING

6.2.1 UNIT TESTING

6.2.2 VALIDATION TESTING

6.2.3 FUNCTIONAL TESTING

7. CONCLUSION

OUTPUT SCREENSHOTS

8. BIBLIOGRAPHY

“Faculty Data Collection”

ABSTRACT : The Faculty Data Collection and submission system is comprehensive web Application desigened to straemline the process of gathering and submmiting information for accredition purpose for National Assessment and Accredition Council(NAAC). The application employs a secure login and signup authentication sysytem to ensure data integrity and confidentiality.

My project proposes a MERN stack-based Faculty Data Collection and Submission system that incorporates authentication to improve the efficiency and confidentiality of the data submission process.

The key enhancements are:

• Role Based Access Control
• Data Submission
• Data Validation and security
• Database Integration

Software Requirement :

- MongoDB
- Express.js
- React.js
- Node.js
- Firebase

Hardware Requirement:

- A web server with node.js installed.
- A database server with mongoDB installed.

Benefits of project:

- Streamlined Accreditation Process: The system streamlines the purpose of gathering and submitting related information for accreditation purpose, reducing the administrative burden .

- Efficiency and Effectiveness: By automating task such as data submission and validation ,the system improves the efficiency of accreditation procedure , saving time and effort.

- Integration Capabilities: Seamlessly integrates with other college systems, such as the student information system and alumni portal, facilitating data sharing and accessibility.

- Accreditation Compliance: By providing a structured and organized platform for data submission, the system helps institutions maintain compliance with accreditation standard and regulations, ensuring continue accreditation status.

- Enhanced Data Integrity: Through data validation mechanism and role based access control, the system ensures accuracy, completeness and confidentiality of submitted information , thereby enhancing data integrity.

1. INTRODUCTION

1.1 ABOUT THE PROJECT

In an era where educational institutions strive for excellence and accountability, accreditation stands as a pivotal benchmark ensuring quality and credibility. The National Assessment and Accreditation Council (NAAC) plays a crucial role in this domain, setting standards and evaluating institutions across India. However, the accreditation process often entails extensive paperwork, cumbersome data collection, and administrative complexities.

To address these challenges, the Faculty Data Collection and Submission System for NAAC emerges as a beacon of efficiency and innovation. Rooted in the robust MERN (MongoDB, Express.js, React.js, Node.js) stack, this web application revolutionizes the conventional methods of data gathering and submission. Its architecture, fortified by stringent authentication measures, not only ensures the integrity and confidentiality of information but also streamlines the accreditation process from end to end.

This project encapsulates a comprehensive suite of features tailored to meet the diverse needs of educational institutions navigating the accreditation landscape. Through role-based access control, it delineates clear hierarchies of authority, empowering stakeholders with tailored permissions for data submission and management. Moreover, the system's integration capabilities transcend organizational silos, fostering seamless collaboration with existing college systems such as student information repositories and alumni portals.

At its core, this system embodies efficiency and effectiveness. By automating tasks like data validation and submission, it alleviates the administrative burdens that often plague accreditation procedures. Institutions stand to benefit from significant time and resource savings, channeling their energies towards academic excellence and institutional advancement.

In essence, this project represents more than just a technological innovation—it embodies a paradigm shift in the accreditation landscape. By marrying technological prowess with institutional imperatives, it paves the way for a future where accreditation processes are not just obligations but opportunities for growth and distinction.

1.2 PROJECT PLAN

Project Plan:

1. Project Initiation
   - Define project objectives, scope, and stakeholders.
   - Establish project team roles and responsibilities.

2. Requirement Analysis
   - Gather detailed requirements from NAAC accreditation guidelines.
   - Conduct stakeholder interviews to identify specific needs and preferences.
   - Document functional and non-functional requirements.

3. Design Phase
   - Design system architecture using the MERN stack.
   - Develop wireframes and prototypes for user interface.
   - Define database schema and integration points.

4. Development
   - Implement authentication system for secure login and signup.
   - Develop role-based access control mechanism
   - Build data submission forms and validation logic.
   - Integrate with MongoDB for database functionality.
   - Develop APIs using Node.js and Express.js.
   - Implement frontend using React.js.

5. Testing
   - Conduct unit testing for individual components.
   - Perform integration testing to ensure seamless functionality.
   - Execute user acceptance testing with stakeholders for validation.
   - Address any bugs or issues identified during testing.

6. Deployment
   - Set up web server with Node.js and MongoDB.
   - Deploy application to production environment.
   - Configure security measures such as SSL certificates and firewall rules.

7. Training and Documentation
   - Provide training sessions for administrators and users.
   - Develop user manuals and documentation for reference
   - Ensure stakeholders are familiar with system functionalities and processes.

8. Maintenance and Support
   - Establish procedures for ongoing maintenance and updates.
   - Implement monitoring tools to track system performance.
   - Provide technical support for any issues encountered post-deployment.
   - Plan for future enhancements based on feedback and evolving requirements.

9. Project Closure
   - Conduct final review to ensure all project objectives are met.
   - Obtain sign-off from stakeholders.
   - Archive project documentation and codebase for future reference.
   - Conduct lessons learned session to capture insights and recommendations for future projects.

2. PROBLEM DEFINITION & FEASIBILITY ANALYSIS

PROBLEM ANALYSIS :

1. Understanding Project Requirements:

- Review Abstract: Analyzed the provided abstract to understand the project's objectives, scope, and key enhancements.

- Identify Stakeholders: Identify stakeholders involved in the project, such as faculty members, administrators, and accreditation bodies.

2. Gathering Technical Specifications:

- Assess Software Requirements: Evaluated the software requirements, including MongoDB, Express.js, React.js Node.js, and Firebase.
- Review Hardware Requirements: Considered the hardware requirements, such as web servers with Node.js and database servers with MongoDB installed.

3. Assessing Functional Requirements:

- Role-Based Access Control (RBAC): Defined user roles and their corresponding permissions to ensure appropriate access levels.
- Data Submission Process: Outlined the process for submitting accreditation-related data, considering ease of use and efficiency.
- Data Validation and Security Measures: Specified data validation rules and security protocols to protect sensitive information.
- Database Integration: Plan the integration of MongoDB and Firebase for efficient data storage and retrieval.

4. Identifying Non-Functional Requirements:

- Usability: Considered usability aspects such as intuitive interfaces and user-friendly interactions.
- Performance: Assess performance requirements to ensure the system operates efficiently under expected load.
- Security: Defined security requirements, including encryption, access controls, and compliance with privacy regulations.

5. Understanding User Perspectives:
- Student Needs: Analyzed administration perspectives to ensure the system meets their requirements for ease of data submission and accessibility.
- Administrator Requirements: Consider the needs of administrators for managing user roles, data validation, and system configuration.
- Accreditation Body Expectations: Understand the expectations of accreditation bodies for data submission, validation, and compliance.

6. Reviewing Compliance Standards:
- NAAC Guidelines: Review NAAC accreditation standards and requirements to ensure the system aligns with established guidelines.
- Data Privacy Regulation: Consider relevant data privacy regulations to ensure compliance with legal requirements.

7. Planning for Feedback Mechanisms:
- Feedback Collection: Plan for feedback mechanisms within the system to gather input from stakeholders throughout the development process.
- Iterative Improvement: Establish processes for incorporating feedback into the project to iteratively improve the system.

Feasibility Analysis:

1. Technical Feasibility
   - System Architecture: The proposed MERN (MongoDB, Express.js, React.js, Node.js) stack offers a robust foundation for building scalable and efficient web applications. With extensive documentation, community support, and a rich ecosystem of libraries and tools, the technical feasibility of implementing this architecture is high.

   - Integration Capabilities: Integration with MongoDB for database functionality and Firebase for authentication enhances the system's technical feasibility. Both MongoDB and Firebase offer extensive documentation, support, and scalability, making them suitable choices for this project.

   - Development Tools: Availability of development tools and frameworks such as React.js, Express.js, and Node.js further contributes to the technical feasibility by streamlining the development process and enhancing developer productivity.

2. Operational Feasibility :
   - User Acceptance: Conducting stakeholder interviews and involving end- users in the requirement gathering process ensures that the system meets their needs and preferences. User acceptance testing during the development phase further validates the system's operational feasibility by confirming its usability and effectiveness in real-world scenarios.

   - Training and Support: Providing comprehensive training sessions and documentation ensures that administrators and users can effectively utilize the system. Ongoing technical support post-deployment enhances operational feasibility by addressing any issues or concerns in a timely manner.

3. Financial Feasibility:
   - Cost-Benefit Analysis: While the development and deployment of the Faculty Data Collection and Submission System entail initial investments in terms of development resources, infrastructure, and licensing fees for third-party services, the long-term benefits outweigh these costs. The system's ability to streamline accreditation processes, improve efficiency, and enhance data integrity leads to cost savings in terms of administrative overhead and time.

   - Return on Investment (ROI): The projected ROI of implementing the system lies in its ability to reduce administrative burden, save time and resources, maintain accreditation compliance, and enhance institutional

credibility. By quantifying these benefits against the initial investment, stakeholders can assess the financial feasibility and potential returns of the project.

4. Legal and Regulatory Feasibility:

- Compliance Requirements: Ensuring compliance with data protection regulations such as GDPR (General Data Protection Regulation) and local privacy laws is crucial for the legal feasibility of the project. Implementing robust data security measures, obtaining necessary permissions for data collection and storage, and adhering to accreditation standards contribute to the system's legal feasibility.

- Intellectual Property Rights: Clearing intellectual property rights for third-party libraries, frameworks, and code used in the project ensures legal compliance and mitigates the risk of copyright infringement or licensing disputes.

5. Schedule Feasibility:

- Project Timeline: Developing a realistic project timeline with clearly defined milestones, deliverables, and dependencies ensures schedule feasibility. Allocating sufficient time for each phase of the project, including requirements gathering, design, development, testing, deployment, and training, helps manage expectations and mitigate schedule risks.

- Resource Availability: Assessing the availability of human resources, expertise, and infrastructure required for the project is essential for schedule feasibility. Adequate resource allocation and contingency planning for potential delays or setbacks contribute to the successful execution of the project within the defined timeline.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 HARDWARE REQUIREMENTS

1. Web Server
   - Processor: A multi-core processor (e.g., Intel Core i5 or higher) to handle concurrent user requests and system processes efficiently.
   - RAM: At least 8GB of RAM to support the web server's runtime memory requirements and ensure smooth operation under heavy loads.
   - Storage: Sufficient storage capacity (e.g., SSD with at least 256GB) to store application files, logs, and temporary data. SSD storage is preferred for faster read/write speeds.
   - Network Interface: Gigabit Ethernet interface for high-speed network connectivity to serve web pages and handle data transmission between the server and client devices.
   - Operating System: Any modern operating system compatible with Node.js and MongoDB, such as Linux (e.g., Ubuntu Server), Windows Server, or macOS.

2. Database Server

   - Processor: Similar to the web server, a multi-core processor to handle database operations efficiently.
   - RAM: At least 8GB of RAM dedicated to the database server to support database management system (e.g., MongoDB) operations and cache frequently accessed data.
   - Storage: Adequate storage capacity (e.g., SSD or HDD with at least 500GB) to store faculty data and database files. SSD storage is recommended for improved database performance.
   - Network Interface: Gigabit Ethernet interface for fast communication between the database server and web server.
   - Operating System: The database server should run an operating system compatible with the chosen database management system, such as Linux (e.g., Ubuntu Server) for MongoDB.

3. Networking Infrastructure:
   - Router: A reliable router capable of handling network

traffic and providing internet connectivity to the web and database servers.

- Switch: A managed switch with sufficient ports to connect the servers, client devices, and other network peripherals.
- Ethernet Cables: High-quality Ethernet cables to establish wired connections between network devices, ensuring stable and high- speed data transmission.

- Internet Connection: A stable and high-speed internet connection to enable remote access to the Faculty Data Collection System and ensure seamless operation.

4. Backup and Redundancy:
   - Backup Storage: Implement a backup solution to regularly back up faculty data and system configurations to prevent data loss in case of hardware failures or disasters.
   - Redundant Components: Consider redundant power supplies and RAID configurations for storage redundancy to minimize downtime and ensure data availability.

5. Monitoring and Management
   - Monitoring Tools: Deploy monitoring software to track the performance, availability, and security of hardware components in real-time.
   - Remote Management: Configure remote management capabilities for the servers to enable administrators to perform maintenance tasks and troubleshoot issues remotely.

3.2 SOFTWARE REQUIREMENTS

1. MongoDB
   - MongoDB is a NoSQL database management system used for storing and managing faculty data submitted through the system.
   - Version: Latest stable release of MongoDB.
   - MongoDB provides flexibility and scalability for handling diverse data types and large volumes of data efficiently.

2. Express.js
   - Express.js is a web application framework for Node.js, used for building the backend API and handling HTTP requests and responses.
   - Version: Latest stable release of Express.js.
   - Express.js simplifies the development of server-side logic and middleware functions, facilitating the creation of robust backend services.

3. React.js
   - React.js is a JavaScript library for building user interfaces, used for developing the frontend components of the Faculty Data Collection System.
   - Version: Latest stable release of React.js.
   - React.js enables the creation of interactive and dynamic user interfaces, enhancing user experience and engagement.

4. Node.js

- Node.js is a server-side JavaScript runtime environment, used for running the backend server and executing server-side logic.
- Version: Latest LTS (Long-Term Support) release of Node.js.
- Node.js provides event-driven architecture and non-blocking I/O operations, making it suitable for building scalable and high-performance web applications.

5. Firebase

- Firebase is a comprehensive platform provided by Google for building mobile and web applications, used for authentication and user management in the Faculty Data Collection System.
- Version: Firebase Authentication SDK for JavaScript.
- Firebase offers easy-to-use authentication services, including email/password authentication, social sign-in, and custom authentication methods, ensuring secure access to the system.

System Requirements

1. Operating System
   - The system should be compatible with major operating systems, including:
   - Windows (e.g., Windows 10, Windows Server)
   - Linux (e.g., Ubuntu, CentOS)
   - macOS

2. Web Browser Compatibility
   - The system should support modern web browsers to ensure compatibility and optimal user experience. Recommended browsers include:
   - Google Chrome
   - Mozilla Firefox
   - Safari
   - Microsoft Edge

3. Network Connectivity
   - Stable internet connection is required for accessing the Faculty Data Collection System, submitting data, and retrieving information.
   - The system should support both wired (Ethernet) and wireless (Wi-Fi) network connections.

4. Hardware Requirements for Client Devices
   - The client devices accessing the system should meet minimum hardware specifications for optimal performance, including:
   - Processor: Dual-core processor or higher
   - RAM: At least 4GB of RAM for smooth operation
   - Storage: Sufficient disk space for storing browser cache and temporary files

5. Screen Resolution
   - The system should support a variety of screen resolutions to accommodate different devices and display sizes, including desktop computers, laptops, tablets, and smartphones.
   - Recommended minimum screen resolution: 1280x800 pixels

6. Software Requirements for Client Devices
   - Web Browser: Latest version of compatible web browsers, including Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.
   - JavaScript: Ensure JavaScript is enabled in the web browser for dynamic functionality and interactivity.

7. Security Considerations
   - The system should implement security measures to protect user data and ensure confidentiality, integrity, and availability. This includes:
   - Secure Socket Layer (SSL) encryption for data transmission over the network
   - User authentication and authorization mechanisms to control access to sensitive information
   - Regular security updates and patches to address vulnerabilities and mitigate risks

8. Accessibility
   - The system should adhere to accessibility standards and guidelines to ensure equal access and usability for all users, including those with disabilities. This includes:
   - Compliance with Web Content Accessibility Guidelines (WCAG) for accessible web content
   - Support for assistive technologies such as screen readers and keyboard navigation

4. SYSTEM DESIGN

DESIGN PHASE :

1. Designing System Architecture:

- MERN Stack Architecture: Design the system architecture using the MERN stack, ensuring scalability, flexibility, and modularity.
- Integration Patterns: Plan integration patterns for seamless interaction between frontend and backend components.

2. User Interface Design:

• Wireframing and Mockups: Create wireframes and mockups based on user requirements and feedback, focusing on simplicity and usability.

3. Authentication and Authorization:

• RBAC Implementation: Define roles and permissions for users, implementing RBAC to control access levels effectively.

• Authentication Mechanisms: Design secure login and signup functionalities to authenticate users and protect data integrity.

4. Data Management:

• Database Schema Design: Design MongoDB collections and Firebase structures to organize data efficiently and support complex queries.

• Data Validation: Implement data validation rules to ensure the accuracy and completeness of submitted information.

5. Security Measures:

• Encryption: Implement encryption mechanisms to secure data transmission and storage.

• Access Controls: Define access controls to restrict unauthorized access to sensitive information.

6. Integration with External Systems:

• API Design: Design RESTful APIs for seamless integration with external systems, enabling data exchange and interoperability.

• Middleware Integration: Plan middleware components to handle data transformation and validation during integration.

7. Frontend Development:

• Develop frontend components using ReactJS, focusing on responsiveness and accessibility across various devices and browsers.

• Implement dynamic form generation functionality to render forms dynamically based on selected accreditation criteria.

8. Backend Development:

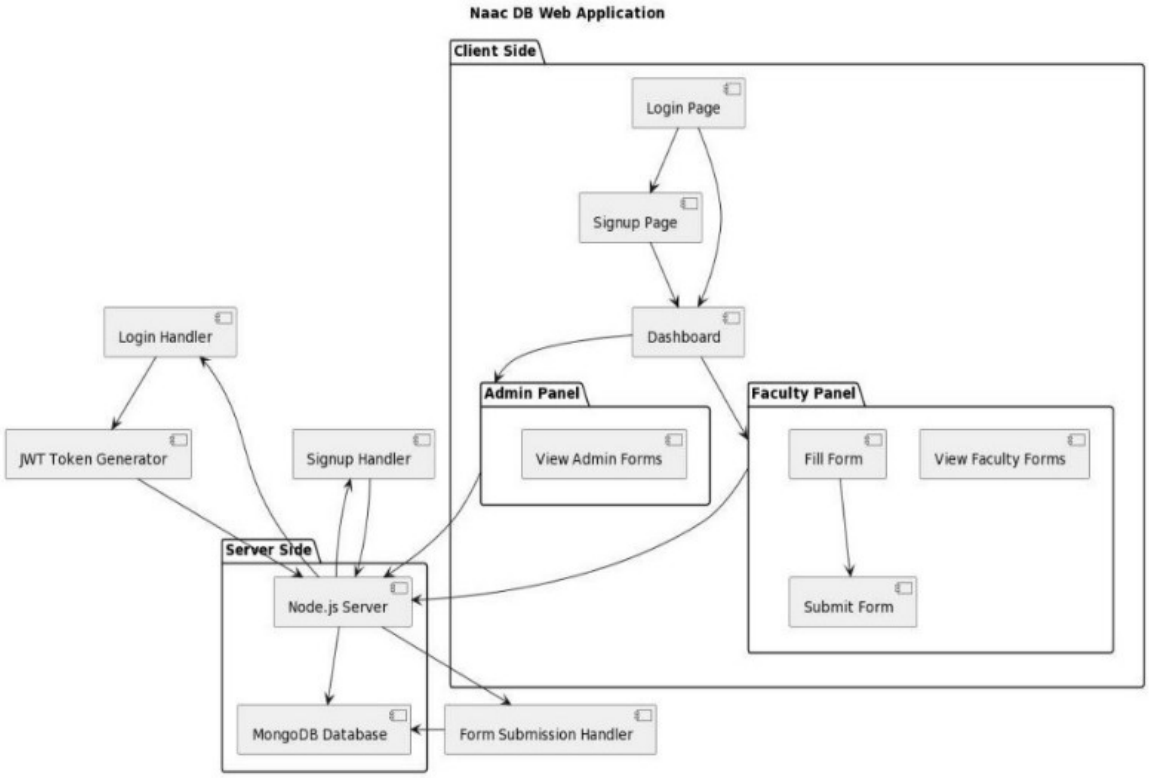
• Build RESTful APIs using NodeJS to handle user authentication, data submission, and retrieval operations.

• Integrate MongoDB with NodeJS for efficient data storage and retrieval, ensuring secure access to faculty information.7. Performance Optimization:

• Caching Strategies: Implementing caching mechanisms to optimize system performance and reduce latency.

• Scalability Considerations: Planed for horizontal and vertical scalability to accommodate increasing user loads.

4.2 USE CASE DIAGRAM



5. IMPLEMENTATION

Implementation Steps:

1. Setup Environment

- Install Node.js on the web server and MongoDB on the database server.
- Set up a project directory for the Faculty Data Collection and Submission System.

2. Backend Development:

- Initialize a Node.js project using npm and set up the Express.js framework for backend development.
- Create routes and controllers for handling authentication, user management, and data submission.
- Implement authentication using Firebase for secure login and signup functionality.
- Develop middleware functions for role-based access control to restrict access to authorized users.

3. Database Integration:

- Set up MongoDB database and define schemas for storing user information and submitted data.
- Establish connections between the backend application and the MongoDB database using Mongoose ORM (Object-Document Mapper).

4. Frontend Development:

- Initialize a React.js project within the existing directory structure.
- Design user interfaces for data submission forms, login, and signup using React components and JSX syntax.
- Implement form validation logic to ensure data completeness and integrity on the client side.
- Integrate Firebase authentication components for secure user authentication in the frontend.

5. Integration Capabilities:

- Develop API endpoints in the backend application to facilitate integration with other college systems.
- Implement data exchange mechanisms such as RESTful APIs or GraphQL for seamless communication between systems.
- Configure authentication and authorization mechanisms for data sharing with external systems securely.

6. Testing:

- Conduct unit testing for backend routes, controllers, and middleware functions using testing frameworks such as Jest or Mocha.
- Perform integration testing to ensure smooth communication between frontend and backend components.
- Validate authentication and access control mechanisms to verify data security and user privacy.

7. Deployment:

- Deploy the backend application to the web server using a suitable hosting service such as Heroku or AWS Elastic Beanstalk.
- Set up continuous integration and deployment pipelines to automate deployment processes.
- Configure SSL certificates and security measures to ensure data transmission security.

8. User Training and Documentation:

- Prepare user manuals and documentation to guide administrators and users on system usage.
- Conduct training sessions to familiarize stakeholders with the features and functionalities of the Faculty Data Collection and Submission System.

9. Maintenance and Support:

- Establish procedures for ongoing maintenance, monitoring, and updates of the system.
- Provide technical support and troubleshooting assistance to users as needed.
- Monitor system performance and address any issues or bugs that may arise post-deployment.

10. User Acceptance Testing (UAT):
    - Invite stakeholders to participate in UAT sessions to validate system functionalities and gather feedback for improvements.
    - Address any issues or concerns raised during UAT and make necessary adjustments to enhance user satisfaction and system usability.

6. SYSTEM TESTING

6.1 SYSTEM IMPLEMENTATION

- Implement each module of the project according to the design specifications.
- Ensure that the modules are developed using appropriate programming languages and libraries.
- Perform rigorous testing during the implementation phase to catch any bugs or errors early on.

6.2 TESTING

6.2.1 UNIT TESTING

- Conduct unit testing for each module individually to ensure that they perform as expected.

- Verify the correctness of functions and methods within each module.

- Test boundary cases and edge conditions to check for robustness.

- Use testing frameworks like pytest or unittest to automate the testing process.

6.2.2 VALIDATION TESTING

- Validate the system against the user requirements and specifications.

- Ensure that the system meets the intended objectives and functionalities.

- Verify that the system accurately detects and classifies emotions from speech recordings.

- Evaluate the system's performance in terms of accuracy, speed, and resource utilization.

6.2.3 FUNCTIONAL TESTING

- Perform functional testing to validate the end-to-end functionality of the system.

- Test each feature and functionality to ensure they work as intended.

- Conduct scenario-based testing to simulate real-world usage scenarios.

- Verify the integration of different modules and components within the system.

- Identify and address any discrepancies or deviations from the expected behavior.

7. CONCLUSION

Conclusion:

The Faculty Data Collection and Submission System for NAAC represents a significant advancement in the realm of accreditation processes for educational institutions. By leveraging modern technologies and innovative solutions, this project addresses key challenges faced in gathering and submitting faculty-related information for accreditation purposes.

Throughout the development of this system, careful attention has been paid to streamlining processes, enhancing efficiency, ensuring data integrity, and maintaining compliance with accreditation standards set forth by the National Assessment and Accreditation Council (NAAC).

The implementation of a MERN (MongoDB, Express.js, React.js, Node.js) stack, coupled with Firebase for authentication, has enabled the creation of a robust, secure, and user-friendly platform. The incorporation of role-based access control ensures that only authorized individuals have access to sensitive data, enhancing confidentiality and privacy.

One of the primary benefits of this project is the streamlining of the accreditation process, reducing administrative burden and saving valuable time and resources for educational institutions. Automation of tasks such as data submission and validation not only improves efficiency but also minimizes the risk of errors and inconsistencies in the data.

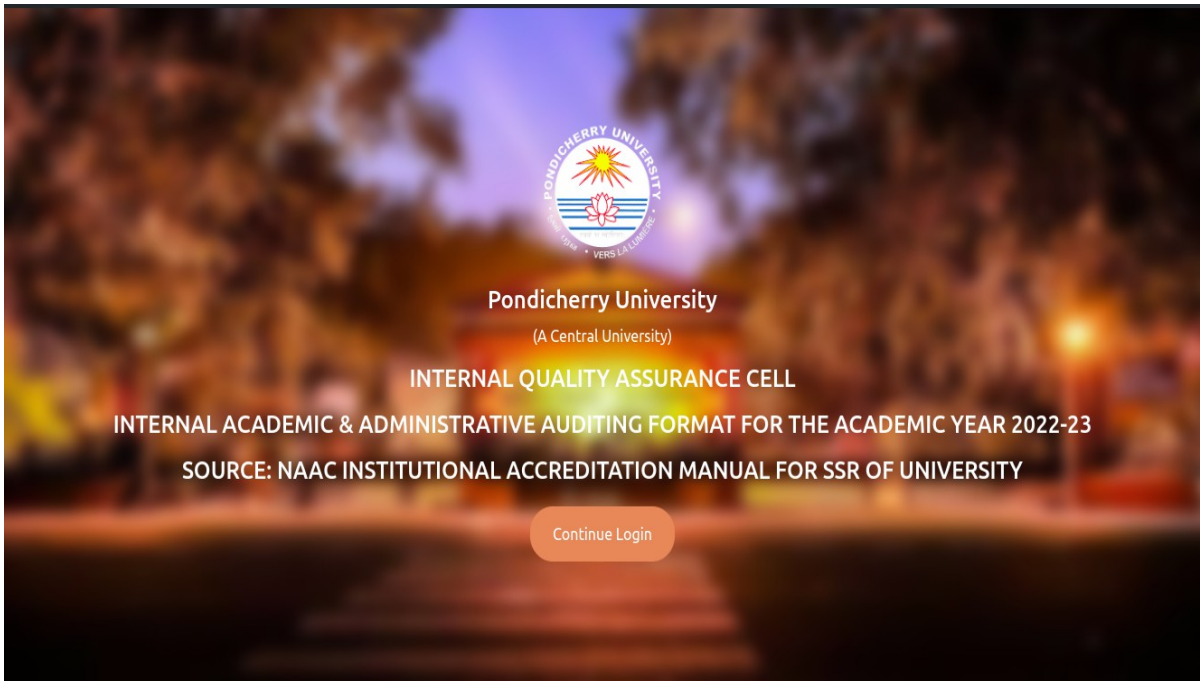
Furthermore, the integration capabilities of the system enable seamless collaboration with other college systems, such as student information systems and alumni portals, facilitating data sharing and accessibility across various platforms.

By providing a structured and organized platform for data submission, this system helps institutions maintain compliance with accreditation standards and regulations, ensuring continuous accreditation status. The implementation of data validation mechanisms and role-based access control further enhances data integrity, accuracy, and completeness, bolstering the credibility of the submitted information.

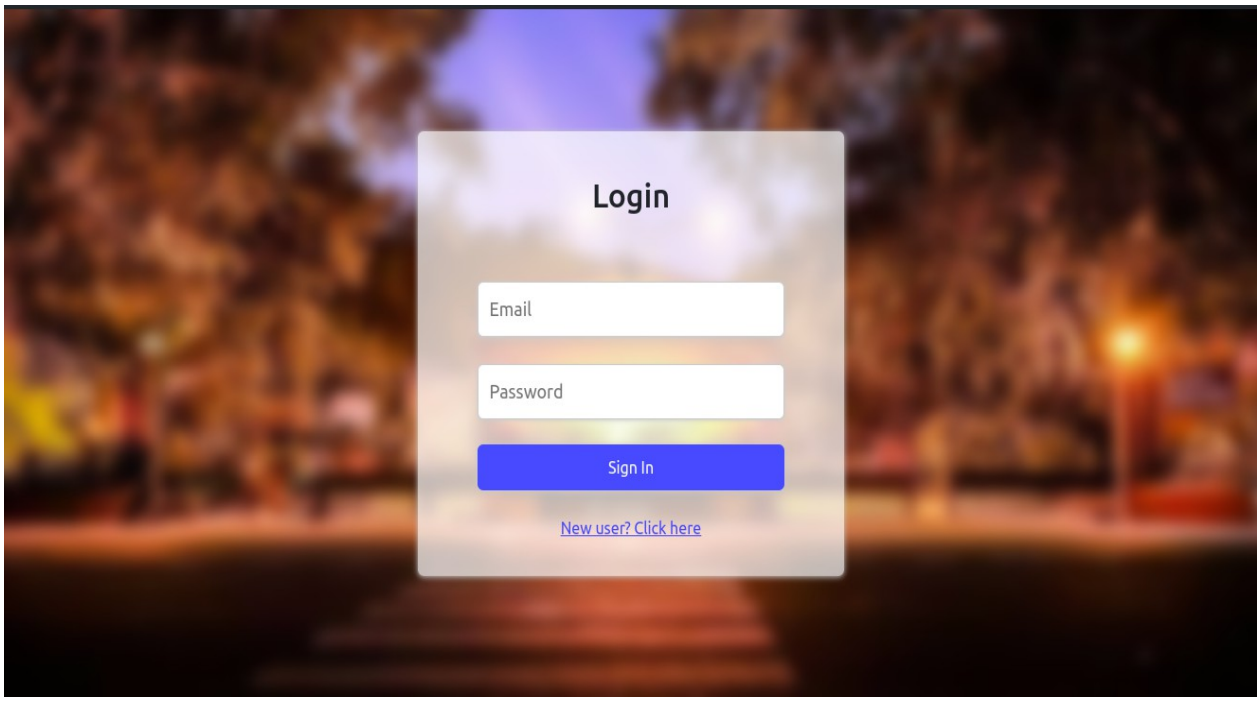
In conclusion, the Faculty Data Collection and Submission System for NAAC represents a significant step forward in simplifying and optimizing the accreditation process for educational institutions. By harnessing the power of technology, this system empowers institutions to meet accreditation requirements efficiently, accurately, and with confidence, ultimately contributing to the enhancement of academic excellence and institutional advancement.

OUTPUS SCREENSHOTS

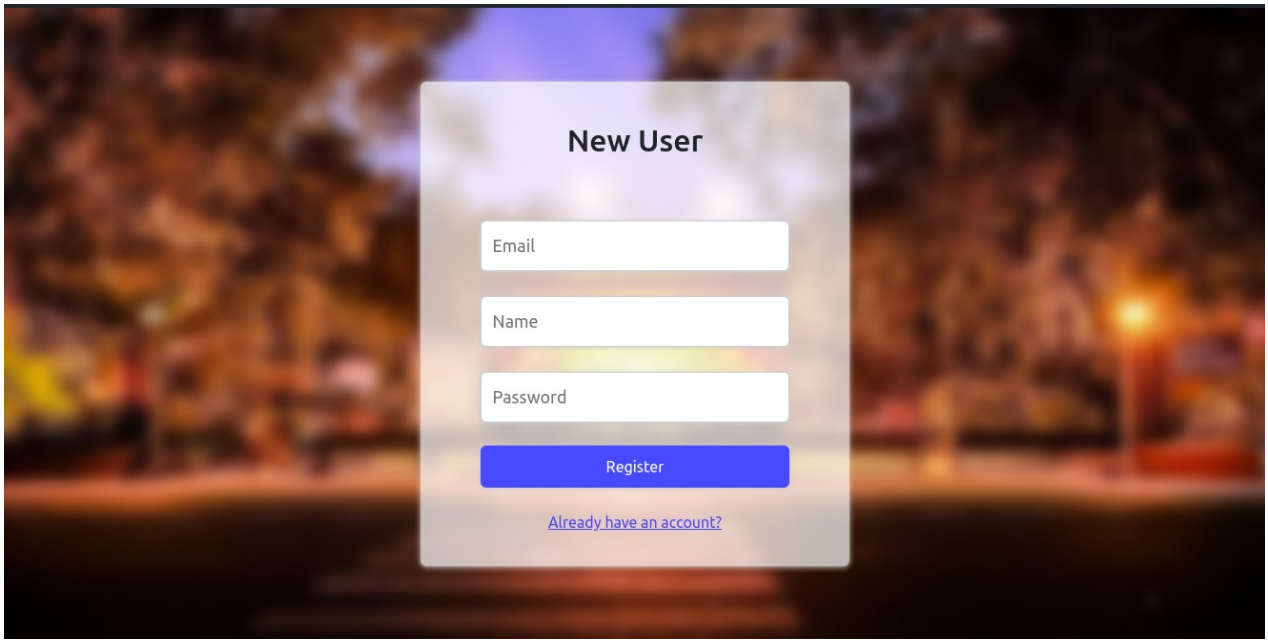
MAIN PAGE



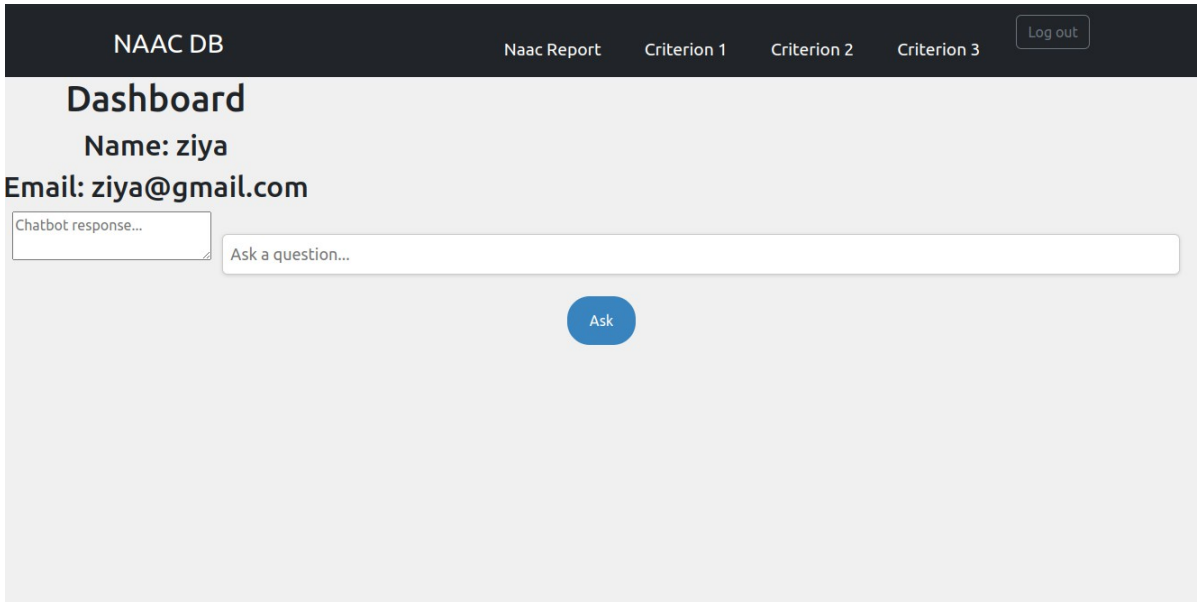
LOGIN PAGE



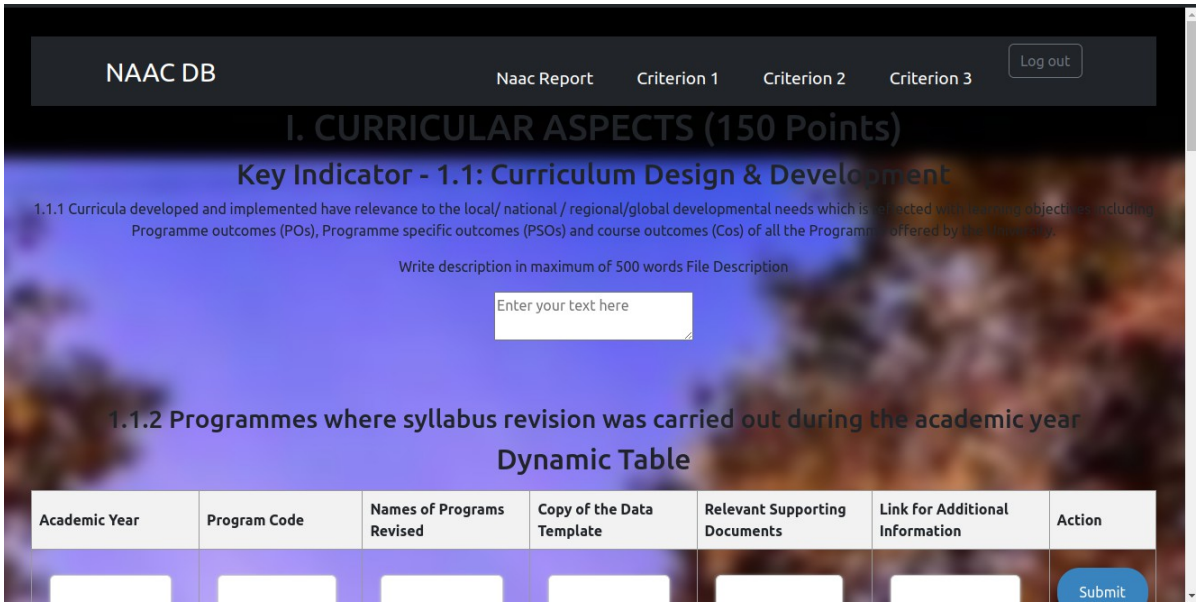
SIGN UP PAGE

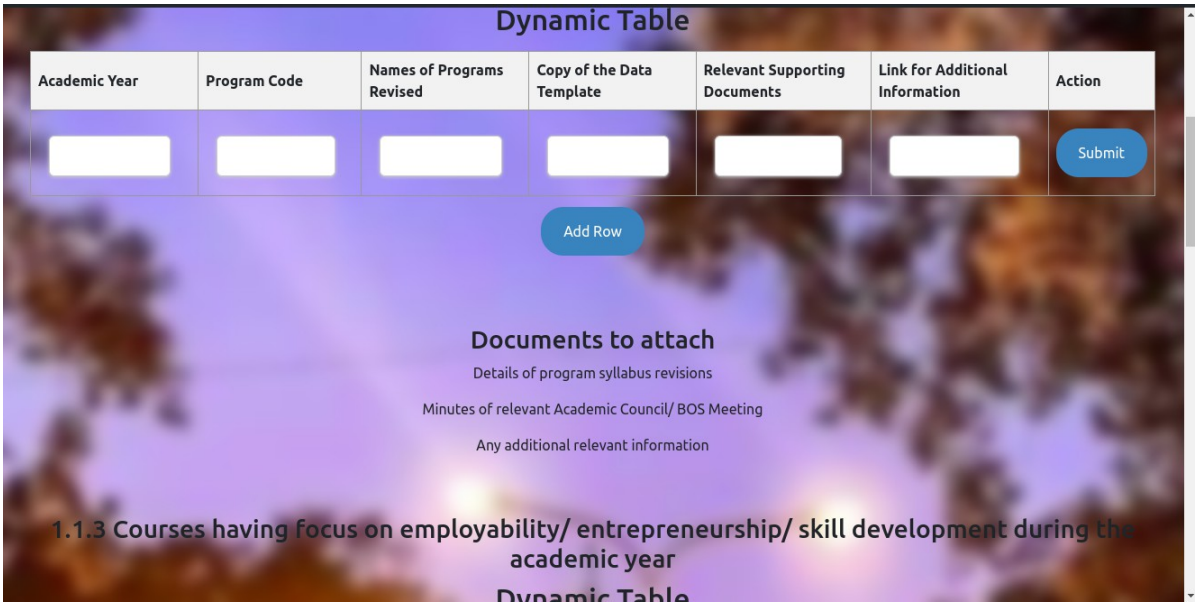


DASHBOARD

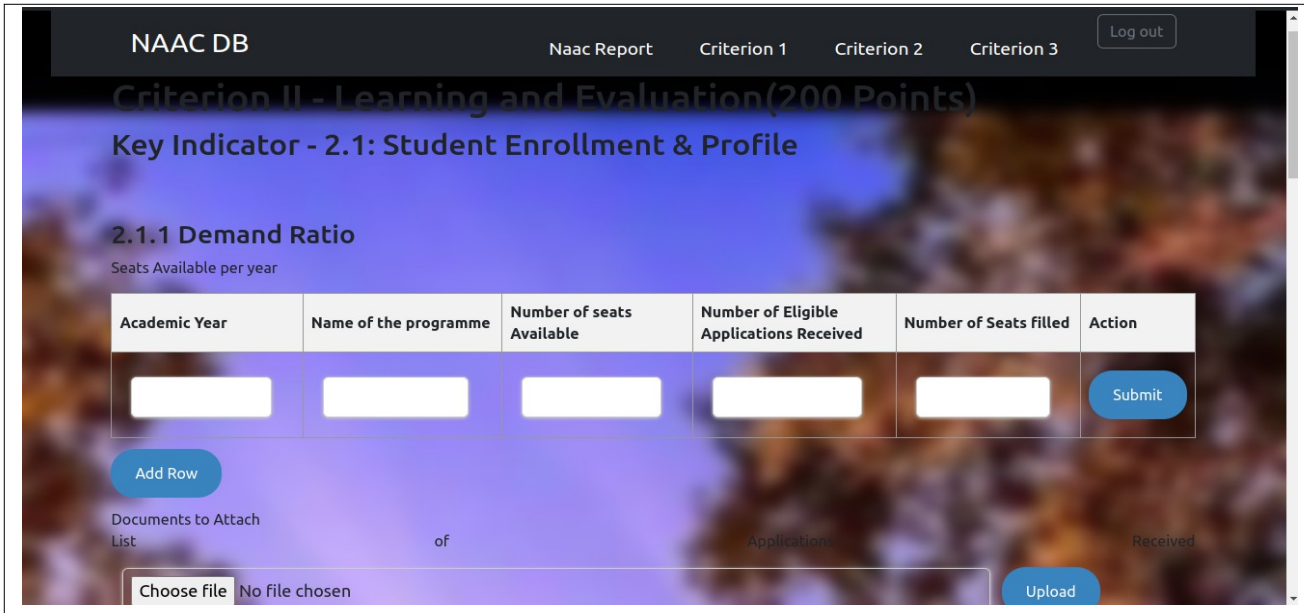


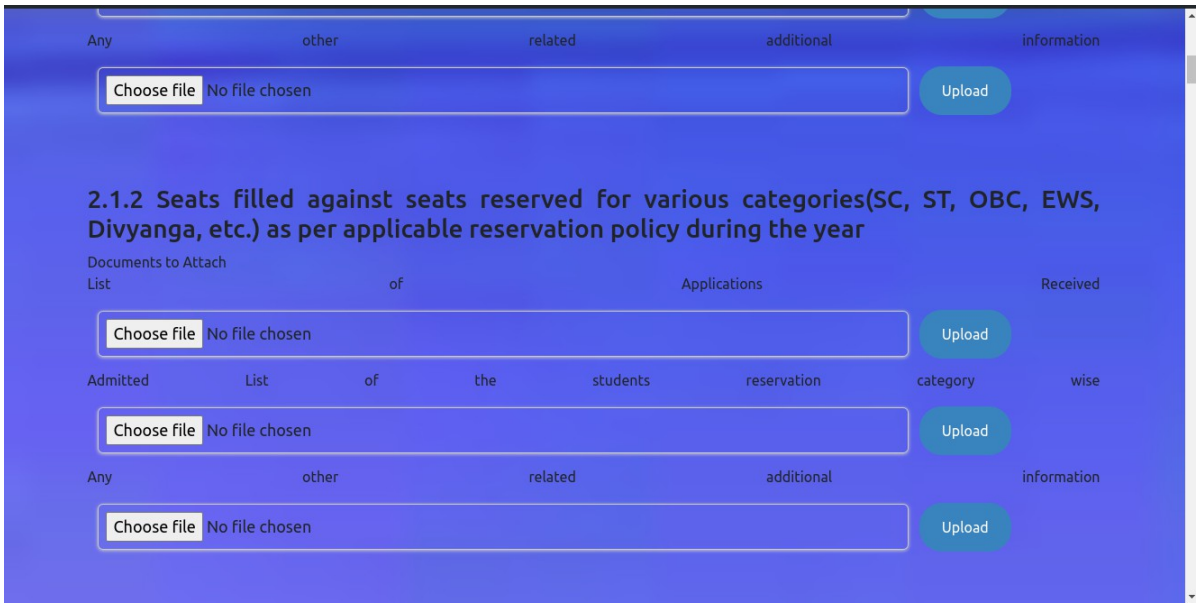
CRITERION 1





CRITERION 2


NAAC DB
Naac Report
Criterion 1
Criterion 2
Criterion 3
Log out
Criterion II - Learning and Evaluation(200 Points)
Key Indicator - 2.1: Student Enrollment & Profile
2.1.1 Demand Ratio
Seats Available per year
Academic Year
Name of the programme
Number of seats Available
Number of Eligible Applications Received
Number of Seats filled
Action
Submit
Add Row
Documents to Attach List of Applications Received
Choose file No file chosen
Upload


Any other related additional information
Choose file No file chosen
Upload
2.1.2 Seats filled against seats reserved for various categories(SC, ST, OBC, EWS, Divyanga, etc.) as per applicable reservation policy during the year
Documents to Attach List of Applications Received
Choose file No file chosen
Upload
Admitted List of the students reservation category wise
Choose file No file chosen
Upload
Any other related additional information
Choose file No file chosen
Upload

NAAC Report

Criterion III - Research, Innovations and Extension

Key Indicator - 3.1 Promotion of Research and Facilities

3.1.1 The institution Research facilities are frequently updated and there is well defined policy for promotion of research which is uploaded on the institutional website and implemented

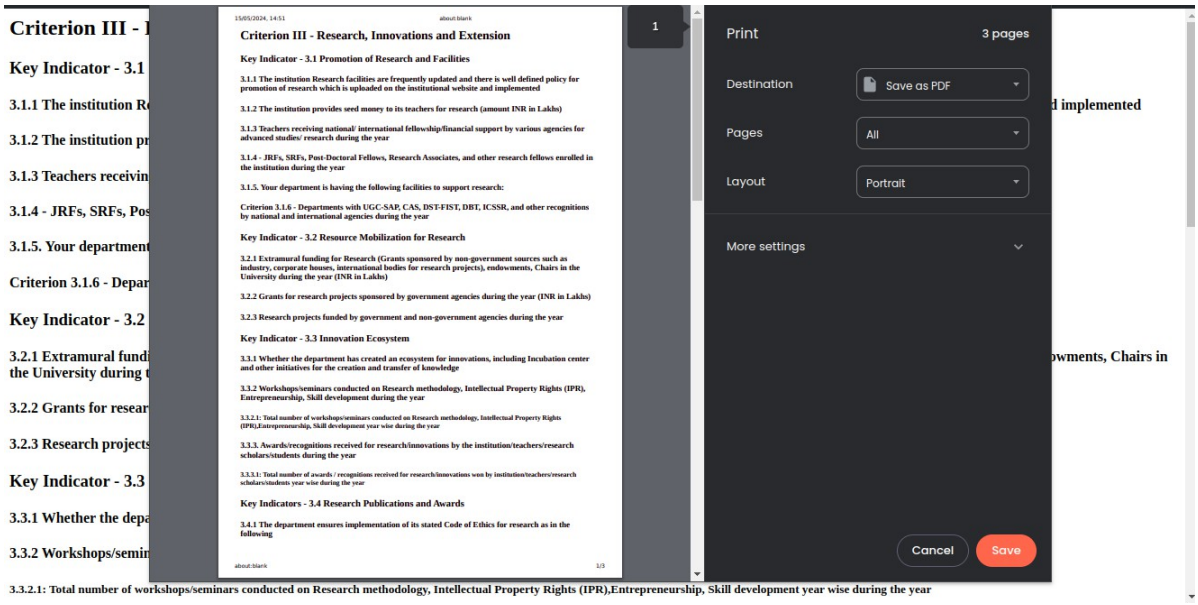
3.1.2 The institution provides seed money to its teachers for research (amount INR in Lakhs)

3.1.3 Teachers receiving national/ international fellowship/financial support by various agencies for advanced studies/ research during the year

3.1.4 - JRFs, SRFs, Post-Doctoral Fellows, Research Associates, and other research fellows enrolled in the institution during the year

3.1.5. Your department is having the following facilities to support research:

Criterion 3.1.6 - Departments with UGC-SAP, CAS, DST-FIST, DBT, ICSSR, and other recognitions by national and international agencies during the year



3.3.2.1: Total number of workshops/seminars conducted on Research methodology, Intellectual Property Rights (IPR),Entrepreneurship, Skill development year wise during the year

8. BIBLIOGRAPHY

REFERENCE

Bibliography:

1. MongoDB Documentation. (n.d.). Retrieved from https://docs.mongodb.com/

- This source provides comprehensive documentation and resources for MongoDB, which is utilized as the database system in the Faculty Data Collection and Submission System.

2. Express.js Documentation. (n.d.). Retrieved from https://expressjs.com/

- The official documentation for Express.js offers detailed information and tutorials for building web applications with Node.js, which is essential for backend development in the MERN stack.

3. React.js Documentation. (n.d.). Retrieved from https://reactjs.org/

- React.js documentation provides guidance and examples for developing user interfaces in JavaScript applications, a key component in the frontend development of the Faculty Data Collection and Submission System.

4. Node.js Documentation. (n.d.). Retrieved from https://nodejs.org/

- Node.js documentation offers resources for server-side JavaScript runtime environment, which is crucial for building scalable and efficient web applications in the MERN stack.

5. Firebase Documentation. (n.d.). Retrieved from https://firebase.google.com/docs

- The Firebase documentation provides guidance and resources for implementing authentication systems, crucial for ensuring data integrity and confidentiality in the Faculty Data Collection and Submission System.

6. National Assessment and Accreditation Council (NAAC). (n.d.).

- Information about NAAC's accreditation standards and guidelines was referenced to ensure compliance and alignment with the requirements of the accreditation process.

These sources were consulted to gather information, guidelines, and best practices for developing the Faculty Data Collection and Submission System for NAAC. They provided valuable insights and technical knowledge necessary for the successful implementation of the project.