Title: IEEE Research Paper Verification and Screening System Design Documentation

Introduction: This document provides a detailed design overview for the IEEE Research Paper Verification and Screening System. The system is designed to streamline the submission, review, and verification of research papers through a multi-stage process involving users, reviewers, and administrators.

System Overview: The system comprises three modules: User Module, Reviewer Module, and Admin Module. Users submit papers, which undergo peer screening, camera-ready screening, and presentation screening. Reviewers assess papers, and administrators monitor the entire process through an admin dashboard.

1. **User Module:**
    - Users register with a unique code generated upon successful registration.
    - Upload initial research papers in PDF format.
    - Papers move through peer screening, where two peers provide recommendations and marks.
    - If recommended, the paper proceeds; otherwise, it is rejected.
    - Users upload a camera-ready version based on peer feedback.
    - Papers undergo presentation screening with reviewer recommendations.
    - Notifications are sent at each stage, and users can check the status.
2. **Reviewer Module:**
    - Reviewers log in to access assigned papers.
    - Reviewers recommend or reject papers in the peer screening stage.
    - Upon successful peer screening, reviewers are notified to review the camera-ready version.
    - Reviewers provide recommendations and marks for camera-ready screening.
    - Admin Module and User Module are updated at each stage.
3. **Admin Module:**
    - Admins logs in to access the admin dashboard.
    - Enter the application number to view the entire flow of a paper.
    - Details include reviewer names, comments, and marks for each stage.
    - No updates are allowed through API; only read-only access for viewing paper status and reviewer names.

Review Process Workflow:

1. **Paper Submission:**
    - Users submit initial research papers in PDF format.
    - A unique alphanumeric code is generated for each user upon registration.
2. **Peer Screening:**
    - Two peers are assigned to review each submitted paper.
    - Peers provide recommendations, marks, and comments.
    - If both peers recommend, the paper proceeds; else, it's rejected.

3. **Camera-Ready Screening:**
    - Users upload a revised camera-ready version based on peer feedback.
    - Different reviewers assess the camera-ready paper, providing recommendations and marks.
    - If both reviewers recommend, it proceeds; else, it's rejected.
4. **Pre-Presentation Screening:**
    - Users apply for presentation upon successful camera-ready screening.
    - New reviewers assess the presentation application, providing recommendations and marks.
    - If both recommend, the paper is accepted; else, it's rejected.

5. **Head Counselor Screening (Admin Module):**
    - Admins monitor the process through the admin dashboard.
    - They view paper status, reviewer details, and overall marks.
    - No direct updates are performed by admins; they ensure the integrity of the screening process.

.

Design Considerations:

Assumptions and Dependencies:

- Assumption: Users have a unique 16-digit alphanumeric code.
- Assumption: Papers undergo three stages: peer screening, camera-ready screening, and presentation screening.
- Dependency: The system relies on a functional MySQL database.

General Constraints:

- Time constraints for development and deployment.
- Accessibility through standard web browsers.
- Intuitive user interface design for users and reviewers.

Goals and Guidelines:

- Security and privacy of user data.
- User-friendly interface for seamless navigation.

- Scalable and maintainable system architecture.
- Optimize for performance to handle a potentially large number of submissions.

Tech Stack Implementation:

Frontend: React.js

1. Use React.js for building the user interface.
2. Leverage React Router for navigation within the application.
3. Implement responsive design for seamless usability across devices.
4. Utilize state management libraries like Redux for efficient data handling.
5. Incorporate Axios for making asynchronous requests to the backend.

Backend: Spring Boot (Java)

1. Develop the backend services using Spring Boot for a scalable and modular architecture.
2. Use Spring Security for robust user authentication and authorization.
3. Implement RESTful APIs to facilitate communication between the frontend and backend.
4. Utilize Spring Data JPA for simplified database operations.
5. Apply validation mechanisms to ensure data integrity and security.

Database: MySQL

1. Employ MySQL as the relational database management system.
2. Design normalized tables to store user information, papers, reviews, and admin details.
3. Use indexes for optimizing query performance.
4. Implement proper foreign key relationships to maintain data integrity.
5. Consider database backups and recovery mechanisms for data protection.