A red heart on a black background

AI-generated content may be incorrect.

Conference Paper Submission System MERN Stack Project Report

# Team Members

Moustafa Mohamed Samir 58-11800

Omar Mohamed Zakaria 58-5054

Yaseen Gadalla 58-1662

Loujain Hussein 58-5449

# 1. Introduction

## 1.1 Purpose

This report outlines our experience developing a conference paper submission system using the MERN stack. The system is designed to streamline the process of paper submission, review, and decision-making by providing role-based access and workflow automation.

## 1.2 Overview

The platform allows users to sign in, submit papers for review, and receive email notifications based on the review outcome. It includes multiple user roles—author, reviewer, chair, and administrator—each with specific permissions and functionality. Only administrators can assign roles to users.

## 1.3 Technology Stack

We used the MERN stack, which consists of:

* - MongoDB: NoSQL database used to store user data, submissions, reviews, and roles.
* - Express.js: Backend web framework for managing API requests and authentication.
* - React.js: Frontend library for building user interfaces.
* - Node.js: JavaScript runtime for server-side logic.

## 1.4 Functionality and Implementation

User Roles and Access Control:

* Administrator: Can create and manage chair and reviewer accounts, oversee all conferences, and access all system areas.
* Chair: Assigns papers to reviewers based on expertise, reviews reports, and makes final accept/reject decisions. Send decision emails with reviewer comments.
* Reviewer: Reads assigned papers, submits review reports with grades and comments, and suggests improvements.
* Author: Registers online, submits papers as PDFs, and receives decision emails with reviewer feedback.

Conference and Paper Management:

* Supports multiple conferences simultaneously, each with its own submission deadline, papers, and reviewers.
* Authors upload PDF papers (4–8 pages) via an online form, stored in MongoDB
* Submission is blocked after the deadline, enforced by server-side checks.
* Chairs assign papers to 2–3 reviewers based on expertise, tracked in the database.

Database (MongoDB):

* Example collections: Users (with fields for role, email, password hash), Conferences (with fields for name, deadline), Papers (with fields for PDF file, authors, conference ID), Reviews (with fields for paper ID, reviewer ID, grades, comments).
* Used Mongoose for schema validation and querying.

## 2. System Design

## 2.1 Architecture

Below is the folder structure used in our project:

/client → React frontend code

/server → Node.js and Express backend

/server/models → Mongoose schemas for MongoDB

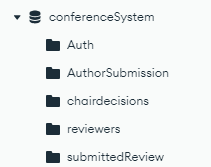
/server/routes → API endpoints

/server/middleware → Auth middleware for role access control

/server/config → Environment variables and DB configs

## 2.2 Database Design

The database schema is divided into multiple collections:

* - Auth: Stores user credentials and roles.
* - Author Submissions: Holds paper metadata such as title, abstract, authors, submission date, and file link.
* - Chair Decisions: Stores final decisions after reviewer input.
* - Reviewers: Maps reviewers to assigned papers.
* - Submitted Reviews: Contains review text, grades, and suggestions.

# 3. User Interface Design

* - Login/Register
* - Author Dashboard
* - Paper Submission Form
* - Reviewer Dashboard
* - Chair’s Review Assignment Panel
* - Admin Role Management