



**ÇUKUROVA UNIVERSITY**  
**ENGINEERING AND ARCHITECTURE FACULTY**  
**DEPARTMENT OF COMPUTER ENGINEERING**

**GRADUATION THESIS**

**DOCUMENT MANAGEMENT SYSTEM FOR INTERMEDIATE LOGISTICS  
COMPANIES**

**BY**

**2022555473 – Aytek Sangün**

**ADVISOR**

**Prof. Dr. S. Ayşe ÖZEL**

**Jun 2025**

**ADANA**

## **ABSTRACT**

This project details the design and development of a web-based document management system, "LogiDocs," tailored for an intermediate logistics company. The primary objective is to address the challenges faced by the company in managing operational documents received via email, which often leads to difficulties in retrieval, processing, and overall operational inefficiency. The LogiDocs system allows employees to create and manage logistics operations (distinguishing between import and export types), register associated global companies (suppliers, buyers, clients) centrally, and assign them specific roles within each operation. Users can then upload relevant documents into a structured, hierarchical file system on the server, organized by operation and participating company/role. This approach ensures easy access, efficient organization, and mitigates database bloat by storing only file paths and metadata in the database. The application is built using a modern technology stack: Node.js and Express.js for the backend RESTful API, PostgreSQL as the relational database managed via Prisma ORM, and React.js for the dynamic and interactive frontend user interface. Key features include a custom user ID format, JWT-based user authentication with bcryptjs for password hashing, brute-force attack prevention on login, and a user-friendly interface for managing operations, global companies, participants within operations, and their associated documents. The system offers functionalities like search, sorting, and pagination for list views, and a collapsible UI for document lists within operation details to enhance usability. LogiDocs aims to significantly improve operational efficiency, document traceability, data integrity, and security for the target logistics company.

**Keywords:** Document Management System, Logistics Operations, Web Application, Node.js, React.js, PostgreSQL, Prisma ORM, JWT Authentication, File Management, Brute-Force Protection, Logistics Software.

## TABLE OF CONTENTS

|  |    |
|--|----|
| ABSTRACT .....   | 2  |
| 1. INTRODUCTION.....   | 6  |
| 1.1. Problem Definition .....                                | 6  |
| 1.2. Project Aim and Objectives.....                         | 6  |
| 1.3. Scope of the Project.....                               | 7  |
| 1.4. Contribution of the Project .....                       | 8  |
| 1.5. Structure of the Report .....                           | 9  |
| 2. RELATED WORK.....   | 10 |
| 2.1. General Purpose Document Management Systems .....       | 10 |
| 2.2. Document Management in Logistics and Supply Chain ..... | 10 |
| 2.3. Industry Applications and Case Studies .....            | 10 |
| 2.4. Comparison with the Proposed System (LogiDocs).....     | 11 |
| 2.5. Summary .....   | 12 |
| 3. MATERIALS AND METHODS .....                               | 13 |
| 3.1. System Architecture .....                               | 13 |
| 3.2. Technologies Used .....                                 | 13 |
| 3.2.1. Backend Technologies.....                             | 13 |
| 3.2.2. Frontend Technologies .....                           | 14 |
| 3.2.3. Development Tools .....                               | 14 |
| 3.3. Data Structure and Management .....                     | 15 |
| 4. PROPOSED METHOD (LOGIDOCs SYSTEM).....                    | 16 |
| 4.1. System Overview and User Roles .....                    | 16 |
| 4.1.1. System Workflow .....                                 | 16 |
| 4.1.2. User Groups.....                                      | 16 |
| 4.2. Database Design.....                                    | 17 |
| 4.2.1. Entity-Relationship (E-R) Diagram.....                | 17 |
| 4.2.2. Table Descriptions.....                               | 17 |
| 4.3. System Implementation Details.....                      | 19 |
| 4.3.1. Backend API Implementation .....                      | 19 |
| 4.3.2. Frontend Implementation .....                         | 21 |
| 4.4. Security Considerations.....                            | 24 |

|   |    |
|---|----|
| 4.4.1. Password Hashing: .....                    | 24 |
| 4.4.2. JWT for Session Management: .....          | 24 |
| 4.4.3. Input Validation:.....                     | 24 |
| 4.4.4. Brute-Force Attack Prevention:.....        | 25 |
| 4.4.5. SQL Injection Prevention (via ORM): .....  | 25 |
| 5. CONCLUSION .....                               | 26 |
| 5.1. Summary of Achievements .....                | 26 |
| 5.2. Strengths of the System.....                 | 26 |
| 5.3. Limitations and Weaknesses .....             | 27 |
| 5.4. Future Work and Potential Enhancements ..... | 27 |
| 5.5. Overall Contribution.....                    | 28 |
| 6. REFERENCES .....                               | 29 |
| 7. APPENDIX .....                                 | 31 |

- List of Figures
  - Figure 4.1: Entity-Relationship (E-R) Diagram for LogiDocs
  - Figure 4.2: Login Page Screenshot
  - Figure 4.3: Operation List Page Screenshot
  - Figure 4.4: New Operation Page Screenshot
  - Figure 4.5: Operation Detail Page Screenshot
  - Figure 4.6: Global Company Management Page Screenshot
- List of Abbreviations
  - DMS: Document Management System
  - UI: User Interface
  - UX: User Experience
  - API: Application Programming Interface
  - JWT: JSON Web Token
  - ORM: Object-Relational Mapper
  - CRUD: Create, Read, Update, Delete
  - SQLi: SQL Injection

## 1. INTRODUCTION

### 1.1. Problem Definition

Intermediate logistics companies often handle a high volume of documentation crucial for their daily operations. These documents, including invoices, bills of lading, customs declarations, packing lists, and various certificates, are typically exchanged via email. As the volume of operations grows, managing these email-based documents becomes increasingly cumbersome. Key challenges include:

- **Difficulty in Retrieval:** Locating specific documents related to a past operation can be time-consuming and inefficient, often involving sifting through numerous emails and attachments.
  - **Lack of Centralization:** Documents are scattered across different email accounts and local storage, leading to version control issues and a lack of a single source of truth.
  - **Processing Delays:** Finding and processing the correct documents quickly is essential for timely execution of logistics tasks. Delays in this area can impact delivery schedules and customer satisfaction.
  - **Collaboration Issues:** Multiple team members working on the same operation may struggle to access the latest versions of documents simultaneously.
  - **Security Concerns:** Sensitive documents stored in email inboxes or local drives may be vulnerable to unauthorized access or data loss.
  - **Inefficient Workflow:** Manual processes for organizing, tracking, and archiving documents consume valuable employee time and are prone to human error.
- The target logistics company for this project faces these exact issues, hindering its operational efficiency and scalability.

### 1.2. Project Aim and Objectives

The primary aim of this project, "LogiDocs," is to design and develop a secure, centralized, and user-friendly web-based document management system specifically tailored to the operational needs of an intermediate logistics company.

The main objectives to achieve this aim are:

1. To develop a system allowing users to create and categorize logistics operations (e.g., import/export) with unique identifiers.

2. To enable central registration and management of global partner companies (suppliers, buyers, clients).
3. To facilitate the assignment of these global companies to specific operations with defined roles.
4. To provide a structured, hierarchical system for uploading and organizing documents related to each operation and its participants.
5. To implement a secure user authentication system with custom user IDs, hashed passwords, and brute-force attack prevention.
6. To ensure documents are stored securely on the server, with only metadata and file paths stored in the database to optimize performance and database size.
7. To develop an intuitive frontend interface for easy navigation, data entry, document upload/retrieval, and list management (including search, sort, and pagination).
8. To improve document traceability, accessibility, and overall operational workflow efficiency for the company.

### 1.3. Scope of the Project

The LogiDocs system encompasses the following core functionalities:

- **User Management:** Secure login for authorized company employees. User accounts are pre-created by an administrator (no public registration).
- **Operation Management:** Creation of new operations with a unique operation number, user-defined name, and type (import/export). Listing, viewing details, updating (name/type), and deleting operations.
- **Global Company Management:** Centralized CRUD (Create, Read, Update, Delete) operations for global partner companies.
- **Operation Participant Management:** Assigning global companies to specific operations with defined roles (supplier, buyer, client) and removing them.
- **Document Management:** Uploading documents under specific operation participants. Documents are stored on the server in a structured folder hierarchy. Listing, viewing (via download/link), and deleting documents.

- **Data Presentation:** List views for operations and global companies feature search, multi-column sorting, and pagination. Operation detail pages present a hierarchical view of participants and their documents, with collapsible document lists for better readability.
- **User Feedback:** Toast notifications for success/error messages on various actions.

#### **Out of Scope for the current version:**

- Public user registration.
- Advanced role-based access control (RBAC) beyond basic authenticated user access (e.g., distinct admin/user roles with different permissions).
- Automated email integration for document fetching.
- Optical Character Recognition (OCR) for document content indexing or data extraction.
- Advanced workflow automation or approval processes.
- Version control for documents.
- Advanced reporting and analytics.

#### **1.4. Contribution of the Project**

The LogiDocs system is expected to provide significant contributions to the target logistics company and potentially to other small to medium-sized logistics enterprises facing similar document management challenges:

- **Increased Operational Efficiency:** Centralizing documents and providing quick search/retrieval reduces the time spent searching for information and speeds up transaction processes.
- **Improved Document Traceability and Accessibility:** All documents related to an operation are stored in a single, structured location, accessible to authorized personnel anytime, anywhere.
- **Reduced Errors:** Minimizes errors associated with manual document handling, lost emails, or using outdated document versions.



- **Enhanced Data Security:** Storing documents on a dedicated server with access controls is more secure than relying on scattered email inboxes. JWT authentication and password hashing further secure user access.
- **Better Collaboration:** Provides a shared platform for team members to access and manage operational documents.
- **Cost-Effectiveness:** As a custom-developed solution leveraging open-source technologies, it can be more cost-effective than licensing commercial DMS software, especially for smaller companies.
- **Scalability Foundation:** The modular design and modern technology stack provide a solid foundation for future enhancements and scaling as the company grows. Academically, this project demonstrates the practical application of web development technologies (Node.js, React, PostgreSQL, Prisma) to solve a real-world business problem, incorporating database design, API development, frontend UI/UX considerations, and security best practices.

### 1.5. Structure of the Report

This report is organized into six main sections. Section 1, the Introduction, defines the problem, outlines the project's aim, objectives, scope, and contributions. Section 2, Related Work, discusses existing document management systems and similar solutions in the logistics domain, comparing them with the proposed system. Section 3, Materials and Methods, details the system architecture, technologies, tools, and data structures employed in this project. Section 4, Proposed Method, provides a comprehensive description of the LogiDocs system, including its database design, system workflow, user interface, and key implementation details with code snippets. Section 5, Conclusion, summarizes the project's outcomes, discusses its strengths and limitations, and suggests avenues for future work. Finally, Section 6 lists the references cited throughout the report.

## **2. RELATED WORK**

Document management has become a critical factor for operational efficiency in logistics companies. Especially for small to medium-sized enterprises (SMEs), conventional file-sharing tools or generic document management systems (DMS) are often insufficient to handle structured document flows and participant-based access control. This section evaluates relevant studies, existing DMS platforms, industry applications, and compares them with the proposed system, LogiDocs.

### **2.1. General Purpose Document Management Systems**

Widely used DMS platforms include:

- Microsoft SharePoint: Provides powerful document management, workflow automation, and version control but can be overly complex and expensive for SMEs.
- Google Workspace (Drive) and Dropbox Business: Offer basic cloud storage and file sharing with limited process-specific capabilities.
- M-Files and DocuWare: Leverage metadata-driven document classification and audit trails but are typically oriented toward larger enterprises.

While these systems provide essential features like access control, versioning, and document history, they often require customization or lack the specific structure needed for logistics operations (e.g., Operation → Participant → Documents).

### **2.2. Document Management in Logistics and Supply Chain**

Logistics operations generate high volumes of critical documents. Digitalization in this domain has been shown to significantly improve traceability, speed, and compliance.

According to Zakrivashevich [1], DMS adoption in logistics enhances operational traceability and secure data sharing. Vrečer et al. [2] demonstrated that DMS lifecycle management is closely tied to organizational maturity, affecting implementation success.

A technology-focused review by Ahmadi Achachlouei et al. [3] highlighted how modular automation architectures support user-specific document workflows, which are essential in logistics. These findings support LogiDocs' structure, which is tailored to logistics-specific workflows.

### **2.3. Industry Applications and Case Studies**

Several real-world implementations validate the advantages of tailored DMS in logistics:

- DHL used Document Logistix to streamline document centralization and reduce paper-based inefficiencies [4].
- GSC Logistics integrated DocuWare to improve legal compliance and document accessibility [5].
- Kyocera documented how digitizing transport documents reduced physical storage costs and enhanced data protection [6].

These implementations reflect the same goals as LogiDocs: operational efficiency, centralized document access, and secure role-based access.

## 2.4. Comparison with the Proposed System (LogiDocs)

Common Features:

- Centralized document repository
- Searchable and filterable document and operation records
- Secure login with authentication
- Transition from email-based/manual document handling to structured digital systems

**Unique Advantages of LogiDocs:**

| Feature         | LogiDocs                               | Commercial DMS                  |
|-----------------|--|---------------------------------|
| Custom Workflow | Operation → Participant → Document     | General folder-based models     |
| Simplicity      | Minimal UI tailored for logistics      | Often complex interfaces        |
| Cost            | Built with open-source tools           | Licensing and integration costs |
| Control         | Full source code and hosting control   | Vendor-dependent                |
| Custom Security | Custom user ID, brute-force protection | Standard practices              |

LogiDocs is purpose-built for an intermediate logistics company. Its open-source stack (Node.js, PostgreSQL, React.js) allows full customization and cost-effective scalability.

## **2.5. Summary**

While many commercial and academic systems offer document management capabilities, most lack the task-specific architecture and usability optimizations needed by small logistics companies. LogiDocs fills this gap by offering a tailored, secure, and efficient solution.

### 3. MATERIALS AND METHODS

#### 3.1. System Architecture

LogiDocs employs a standard three-tier architecture:

1. **Presentation Tier (Frontend):** A single-page application (SPA) built with React.js, responsible for the user interface and user interaction. It communicates with the backend via RESTful API calls.
2. **Application Tier (Backend):** A RESTful API server built with Node.js and Express.js. It handles business logic, data processing, user authentication, file uploads, and communication with the database.
3. **Data Tier (Database & File System):**
  - **PostgreSQL:** A relational database used to store metadata about users, operations, global companies, operation participants, and documents (including file paths).
  - **Server File System:** Used to store the actual uploaded document files in a structured directory hierarchy.

#### 3.2. Technologies Used

##### 3.2.1. Backend Technologies

**3.2.1.1. Node.js:** A JavaScript runtime environment chosen for its asynchronous, non-blocking I/O model, making it efficient for I/O-intensive applications like file uploads and API request handling. Its large npm (Node Package Manager) ecosystem provides a wealth of libraries.

**3.2.1.2. Express.js:** A minimal and flexible Node.js web application framework used to build the RESTful API, providing robust features for routing, middleware, and request/response handling.

**3.2.1.3. PostgreSQL:** A powerful, open-source object-relational database system known for its reliability, feature robustness, and data integrity. Chosen for its strong support for relational data structures required by the application.

**3.2.1.4. Prisma ORM:** A modern database toolkit for Node.js and TypeScript. It provides a type-safe query builder, schema migration system, and an intuitive API for database interactions, simplifying data access and reducing boilerplate SQL.

**3.2.1.5. JSON Web Tokens (JWT):** Used for implementing stateless user authentication.

Upon successful login, a JWT is generated and sent to the client, which then includes it in the Authorization header for subsequent requests to protected API routes.

**3.2.1.6. Bcrypt.js:** A library for hashing passwords. It incorporates a salt to protect against rainbow table attacks, ensuring that user passwords are stored securely.

**3.2.1.7. Multer:** A Node.js middleware for handling multipart/form-data, used primarily for uploading files. It facilitates saving uploaded files to the server's file system.

### **3.2.2. Frontend Technologies**

**3.2.2.1. React.js:** A popular JavaScript library for building user interfaces, developed by Facebook. Its component-based architecture allows for the creation of reusable UI elements and efficient state management, leading to dynamic and responsive web applications.

**3.2.2.2. React Router DOM:** Provides declarative routing for React applications, enabling navigation between different views/pages within the single-page application.

**3.2.2.3. Axios:** A promise-based HTTP client for the browser and Node.js. Used to make API requests from the frontend to the backend server.

**3.2.2.4. React Hook Form:** A performant, flexible, and extensible forms library for React. It simplifies form state management, validation, and submission handling by leveraging uncontrolled inputs.

**3.2.2.5. React Toastify:** A library for adding non-intrusive toast notifications to the application, providing user feedback for actions like successful operations or errors.

**3.2.2.6. Vite:** A modern frontend build tool that provides an extremely fast development server and optimized builds for production. Chosen over Create React App for its speed and developer experience.

### **3.2.3. Development Tools**

**3.2.3.1. Visual Studio Code:** A lightweight but powerful source code editor with excellent support for JavaScript, Node.js, and React development, including debugging and Git integration.

**3.2.3.2. Git & GitHub:** Git for version control to track changes in the codebase, and GitHub for remote repository hosting and collaboration (if applicable).

**3.2.3.3. Postman:** An API platform for designing, building, testing, and iterating on APIs. Used extensively for testing backend API endpoints during development.

**3.2.3.4. pgAdmin:** A feature-rich open-source administration and development platform for PostgreSQL, used for database inspection and management.

### 3.3. Data Structure and Management

The primary data managed by the system includes:

- **User Data:** Usernames (custom format L-XXXXXX), hashed passwords, names.
- **Operation Data:** Unique operation numbers, user-defined names, types (import/export), creation/update timestamps.
- **Global Company Data:** Unique company names, optional address, tax number, contact information.
- **Operation Participant Data:** Links operations to global companies with a specific role (supplier, buyer, client) for that operation.
- **Document Metadata:** Original file names, system-generated stored file names, file paths on the server, MIME types, file sizes, and upload timestamps.  
Actual document files are stored on the server's file system in a structured hierarchy: uploads/<operation\_number\_slug>/<company\_name\_slug>-<role\_slug>/<stored\_file\_name>. This separation of metadata (in DB) and files (on file system) optimizes database performance and storage.

## 4. PROPOSED METHOD (LOGIDOCs SYSTEM)

### 4.1. System Overview and User Roles

#### 4.1.1. System Workflow

The LogiDocs system operates based on the following general workflow:

1. **Authentication:** Authorized employees log in using their pre-assigned custom User ID (e.g., "L-000001") and password. The system validates credentials and, upon success, issues a JWT.
2. **Global Company Management (Optional Initial Step):** Users can add new global partner companies or manage existing ones. This is typically done once per company.
3. **Operation Creation:** Users create a new logistics operation, providing a unique operation number, a descriptive name, and selecting the type (import or export).
4. **Participant Assignment:** Users assign global companies to the created operation, specifying their role (supplier, buyer, or client) for that particular operation. A global company can participate in multiple operations, potentially with different roles.
5. **Document Upload:** Users upload relevant documents under the appropriate participant within an operation. Documents are stored on the server, and their metadata is saved in the database, linked to the participant.
6. **Data Retrieval and Management:** Users can list, search, sort, and paginate through operations and global companies. They can view operation details, which display a hierarchical list of participants and their associated documents. Document lists are transformed (collapsible) for better readability. Users can also update operation details (name/type), manage participants, and upload/delete documents.

#### 4.1.2. User Groups

Currently, the system implements a single user group:

- **Authenticated Employee:** Any logged-in user has full access to create, read, update, and delete operations, companies, participants, and documents.

#### Planned Future User Groups:

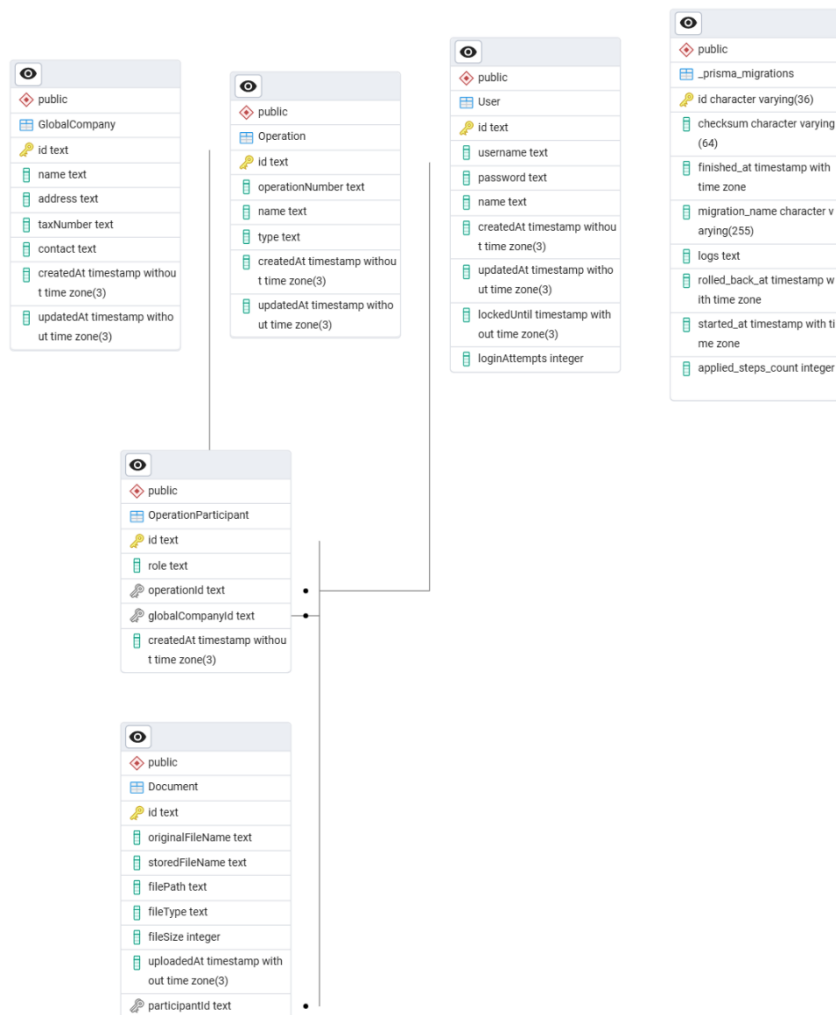
- **Administrator:** Would have additional privileges, such as managing user accounts (creating new users, resetting passwords), system configuration, and potentially accessing audit logs.



- **Standard User:** Might have restricted permissions, e.g., only able to view certain operations or upload documents but not delete operations.

## 4.2. Database Design

### 4.2.1. Entity-Relationship (E-R) Diagram



**Figure 4.1:** Entity-Relationship (E-R) Diagram for LogiDocs

### 4.2.2. Table Descriptions

- **User:** Stores information about system users.
  - id (PK, UUID): Unique identifier for the user.
  - username (String, Unique): Custom user ID, e.g., "L-000001".
  - password (String): Bcrypt-hashed password.
  - name (String, Optional): User's name.

- loginAttempts (Int): Count of failed login attempts.
- lockedUntil (DateTime, Optional): Timestamp until which the account is locked due to excessive failed logins.
- **Operation:** Represents a single logistics operation.
  - id (PK, UUID): Unique identifier for the operation.
  - operationNumber (String, Unique): Company-defined unique number for the operation.
  - name (String): Descriptive name of the operation.
  - type (String): Type of operation ("ithalat" or "ihracat").
- **GlobalCompany:** Stores information about centrally managed partner companies.
  - id (PK, UUID): Unique identifier for the global company.
  - name (String, Unique): Name of the company.
  - address, taxNumber, contact (String, Optional): Additional company details.
- **OperationParticipant:** Acts as a junction table linking Operation and GlobalCompany, and defines the role of the company in that specific operation. It also holds the documents for that role in that operation.
  - id (PK, UUID): Unique identifier for the participation instance.
  - role (String): Role of the company in the operation ("tedarikci", "alici", "musteri").
  - operationId (FK, String): Foreign key referencing Operation.id.
  - globalCompanyId (FK, String): Foreign key referencing GlobalCompany.id.
- **Document:** Stores metadata about uploaded documents.
  - id (PK, UUID): Unique identifier for the document.
  - originalFileName (String): The original name of the uploaded file.
  - storedFileName (String): The unique name (e.g., UUID-based) under which the file is stored on the server.

- filePath (String, Unique): The relative path to the file on the server's file system.
- fileType (String): MIME type of the file.
- fileSize (Int): Size of the file in bytes.
- participantId (FK, String): Foreign key referencing OperationParticipant.id, linking the document to a specific company's role in an operation.

## 4.3. System Implementation Details

### 4.3.1. Backend API Implementation

The backend is a RESTful API built with Node.js and Express.js, interacting with the PostgreSQL database via Prisma ORM.

#### 4.3.1.1. Authentication Module (Login, Brute-Force Protection)

The `/api/auth/login` endpoint handles user login. It validates credentials against hashed passwords stored in the `User` table. Upon successful authentication, a JWT is generated. To prevent brute-force attacks, the system tracks failed login attempts. After 5 failed attempts, the account is locked for 3 minutes.

```
if (user.lockedUntil && new Date() < new Date(user.lockedUntil)) {
  const timeLeft = Math.ceil((new Date(user.lockedUntil).getTime() - new Date().getTime()) / (1000 * 60));
  return res.status(429).json({ error: `Çok fazla başarısız deneme. Hesabınız ${timeLeft} dakika daha kilitli.` });
}

const isMatch = await bcrypt.compare(password, user.password);

if (isMatch) {
  let attempts = user.loginAttempts + 1;
  let updateData = { loginAttempts: attempts };

  if (attempts >= MAX_LOGIN_ATTEMPTS) {
    updateData.lockedUntil = new Date(Date.now() + LOCKOUT_DURATION_MINUTES * 60 * 1000);
    updateData.loginAttempts = 0; // Kilitlendikten sonra denemeleri sıfırla
    await prisma.user.update({ where: { id: user.id }, data: updateData });
    return res.status(429).json({ error: `Çok fazla başarısız deneme. Hesabınız ${LOCKOUT_DURATION_MINUTES} dakika kilitlendi.` });
  } else {
    await prisma.user.update({ where: { id: user.id }, data: updateData });
  }
  return res.status(401).json({ error: 'Geçersiz Kullanıcı ID veya şifre.' });
}

if (user.loginAttempts > 0 || user.lockedUntil) {
  await prisma.user.update({
    where: { id: user.id },
    data: { loginAttempts: 0, lockedUntil: null },
  });
}

const payload = {
  user: { id: user.id, username: user.username, name: user.name }
};
jwt.sign(payload, JWT_SECRET, { expiresIn: '1h' }, (err, token) => {
  if (err) throw err;
  res.json({
    token,
    user: { id: user.id, username: user.username, name: user.name }
  });
});
```

#### 4.3.1.2. Operation Management Module (`/api/operations`)

Provides CRUD endpoints for operations. `GET /` supports search (by name, number), type filtering, sorting, and pagination. `GET /:id` retrieves detailed operation data including participants and their documents.

#### 4.3.1.3. Global Company Management Module (`/api/global-companies`)

Provides CRUD endpoints for global companies. `GET /` supports search, sorting, and pagination. A separate `GET /all` endpoint provides all companies without pagination for dropdown population.

#### 4.3.1.4. Operation Participant Management Module

(`/api/operations/:operationId/participants`)

Allows adding and removing global companies (with roles) as participants to/from an operation.

#### 4.3.1.5. Document Upload and Management Module

(`/api/participants/:participantId/documents`, `/api/files/documents/:documentId`)

Handles file uploads using Multer. Files are saved to a structured directory on the server. Endpoints allow uploading documents for a specific participant and deleting documents by ID (which also removes the physical file).

```
const storage = multer.diskStorage({
  destination: async function (req, file, cb) {

    const { participantId } = req.params;
    if (!participantId) {
      return cb(new Error("Participant ID not found in request params for destination"));
    }

    try {
      const participant = await prisma.operationParticipant.findUnique({
        where: { id: participantId },
        include: {
          operation: true,
          globalCompany: true
        }
      });

      if (!participant || !participant.operation || !participant.globalCompany) {
        return cb(new Error("Participant, associated operation, or global company not found for destination."));
      }

      const operationNumberSlug = slugify(participant.operation.operationNumber);
      const companyNameSlug = slugify(participant.globalCompany.name);
      const roleSlug = slugify(participant.role);

      const uploadPath = path.join(__dirname, '..', 'uploads', operationNumberSlug, `${companyNameSlug}-${roleSlug}`);

      if (!fs.existsSync(uploadPath)) {
        fs.mkdir(uploadPath, { recursive: true }, (err) => {
          if (err) {
            console.error("Error creating directory:", err);
            return cb(err);
          }
          cb(null, uploadPath);
        });
      } else {
        cb(null, uploadPath);
      }
    }
  }
});
```

```

    filename: function (req, file, cb) {
      const originalNameSlug = slugify(path.parse(file.originalname).name);
      const extension = path.extname(file.originalname);
      const uniqueFileName = `${uuidv4()}_${originalNameSlug}${extension}`;
      cb(null, uniqueFileName);
    }
  });

  const fileFilter = (req, file, cb) => {
    const allowedMimeTypes = [
      'application/pdf', 'application/msword',
      'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
      'application/vnd.ms-excel', 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',
      'image/jpeg', 'image/png', 'image/gif'
    ];
    if (allowedMimeTypes.includes(file.mimetype)) {
      cb(null, true);
    } else {
      cb(new Error('Geçersiz dosya tipi.'), false);
    }
  };

const upload = multer({
  storage: storage,
  limits: { fileSize: 1024 * 1024 * 10 }, // 10 MB
  fileFilter: fileFilter
});

const newDocument = await prisma.document.create({
  data: {
    originalFileName: req.file.originalname,
    storedFileName: req.file.filename,
    filePath: relativeFilePath,
    fileType: req.file.mimetype,
    fileSize: req.file.size,
    participantId: participantId, // Artık participantId
  },
});

```

## 4.3.2. Frontend Implementation

The frontend is a Single Page Application (SPA) built using React.js, Vite, and React Router DOM for navigation.

### 4.3.2.1. User Interface (UI) Design and Navigation

The UI is designed to be clean and intuitive. A main navigation bar provides access to key sections: Operations, New Operation, and Global Companies (for authenticated users).

Login/Logout options are also present. List views utilize tables with sorting and pagination.

Detail views use cards and collapsible sections for clarity.

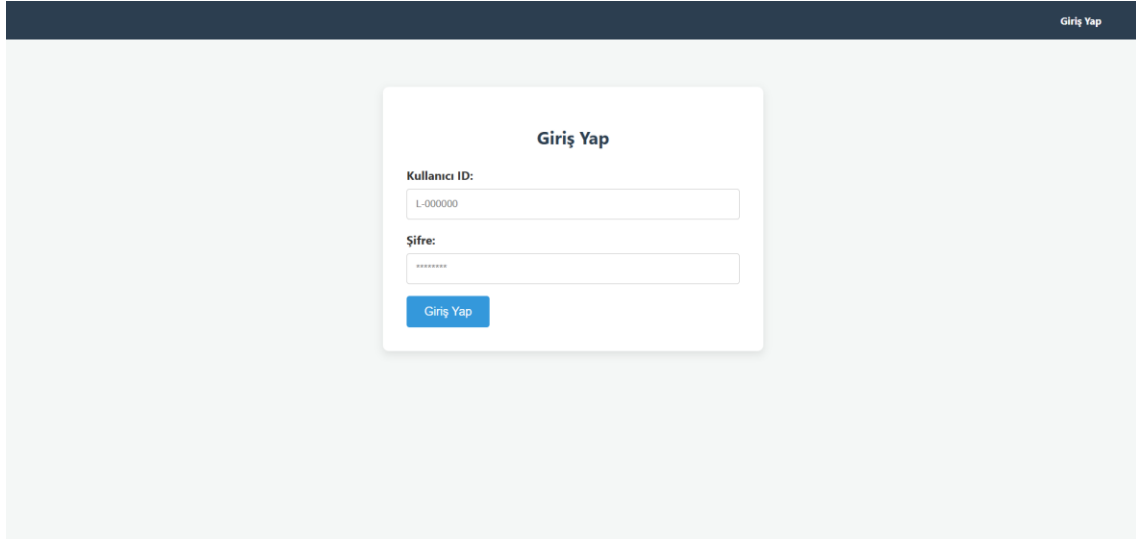
### 4.3.2.2. State Management

**Global State (Authentication):** React Context API (`AuthContext`) is used to manage global authentication state (current user, token, loading status) and provide login/logout functions throughout the application.

**Local Component State:** `useState` and `useEffect` hooks are extensively used for managing local component state, such as form inputs, loading indicators for specific actions, error messages, and data fetched from the API.

#### 4.3.2.3. Key Pages and Functionalities

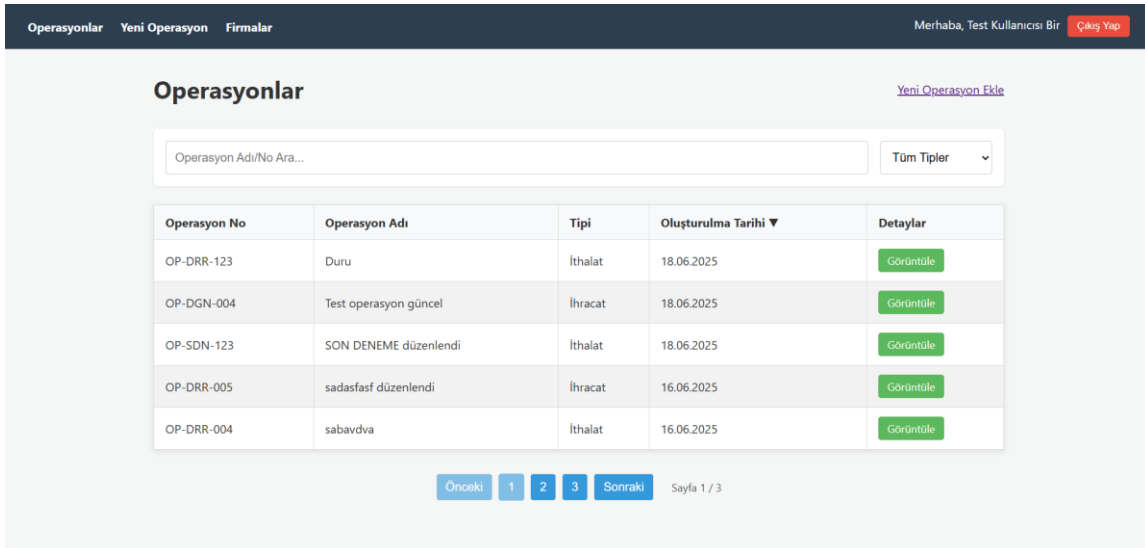
- **Login Page:** Allows users to log in with their custom User ID and password. Implements client-side validation for input format.



The screenshot shows a login page titled "Giriş Yap". It features a central form with two input fields: "Kullanıcı ID:" and "Şifre:". The "Kullanıcı ID:" field contains the text "L-000000". Below the fields is a blue button labeled "Giriş Yap". The page has a dark blue header with the text "Giriş Yap" on the right.

**Figure 4.2:** Login Page Screenshot

- **Operation List Page:** Displays a paginated and sortable list of all operations. Users can search for operations by name or number and filter by type (import/export).



The screenshot shows the "Operasyonlar" page. It has a dark blue header with navigation links "Operasyonlar", "Yeni Operasyon", and "Firmalar". On the right, it says "Merhaba, Test Kullanıcısı Bir" and "Çıkış Yap". Below the header, there is a search bar "Operasyon Adı/No Ara..." and a dropdown menu "Tüm Tipler". The main content is a table with the following data:

| Operasyon No | Operasyon Adı         | Tipi    | Oluşturulma Tarihi ▼ | Detaylar                  |
|--------------|-----------------------|---------|----------------------|---------------------------|
| OP-DRR-123   | Duru                  | İthalat | 18.06.2025           | <a href="#">Görüntüle</a> |
| OP-DGN-004   | Test operasyon güncel | İhracat | 18.06.2025           | <a href="#">Görüntüle</a> |
| OP-SDN-123   | SON DENEME düzenlendi | İthalat | 18.06.2025           | <a href="#">Görüntüle</a> |
| OP-DRR-005   | sadasfasf düzenlendi  | İhracat | 16.06.2025           | <a href="#">Görüntüle</a> |
| OP-DRR-004   | sabavdva              | İthalat | 16.06.2025           | <a href="#">Görüntüle</a> |

At the bottom of the table, there are pagination controls: "Önceki", "1", "2", "3", "Sonraki", and "Sayfa 1 / 3".

**Figure 4.3:** Operation List Page Screenshot

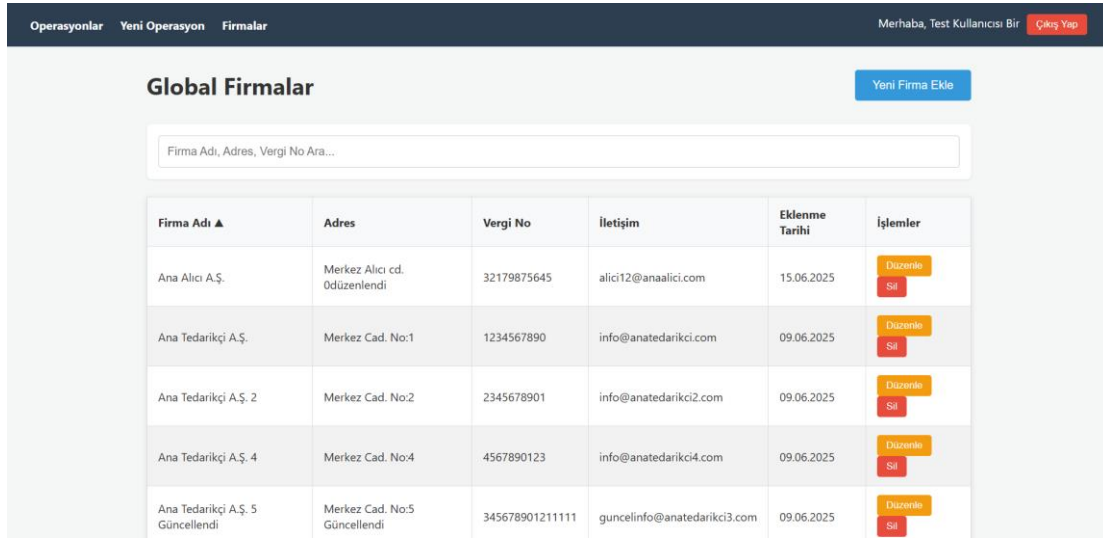
- **New Operation Page:** A form, managed by `react-hook-form`, allows users to create new operations with client-side validation.

**Figure 4.4:** New Operation Page Screenshot

- **Operation Detail Page:** Shows comprehensive details of a selected operation, including its participants (company name and role) and a list of documents for each participant. Document lists are collapsible to improve readability. Users can add/remove participants, upload/delete documents, and edit/delete the operation itself from this page, often via modal dialogs.

**Figure 4.5:** Operation Detail Page Screenshot

- **Global Company Management Page:** Provides full CRUD functionality for global companies in a paginated and sortable table. Forms for adding/editing companies are displayed in modals and managed by `react-hook-form`.



**Figure 4.6:** Global Company Management Page Screenshot

```

<div key={participant.id} className="participant-card">
  <div className="participant-header clickable" onClick={() => toggleParticipantExpansion(participant.id)}>
    <h3>
      {participant.globalCompany.name} <span className="role-badge">{participant.role}</span>
      <span className="expand-indicator">
        {expandedParticipants[participant.id] ? '▼' : '▶'}
      </span>
    </h3>
    <div>
      <button onClick={(e) => { e.stopPropagation(); openUploadModal(participant);}} className="button-small success" disabled={actionLoading}>Evrak Yükle</button>
      <button onClick={(e) => { e.stopPropagation(); handleRemoveParticipant(participant.id, participant.globalCompany.name);}} className="button-small danger" disabled={actionLoading}>Kaldır</button>
    </div>
  </div>
  {expandedParticipants[participant.id] && (
    <div className="documents-section">
      <div>
        {participant.documents && participant.documents.length > 0 ? (
          <ul className="document-list">
            {participant.documents.map((doc) => (
              <li key={doc.id}>
                <a href={doc.fullPath} target="_blank" rel="noopener noreferrer">
                  {doc.originalFileName}
                </a>
                <span className="file-info"> {(doc.fileType), ((doc.fileSize / 1024).toFixed(1)) KB}</span>
                <button onClick={() => handleDeleteDocument(doc.id, doc.originalFileName)} className="button-tiny danger" title="Evrakı Sil" disabled={actionLoading}>Sil</button>
              </li>
            ))}
          </ul>
        ) : (
          <p>Bu katılımcı için henüz evrak yüklenmemiş.</p>
        )
      )}
    </div>
  )}
</div>

```

## 4.4. Security Considerations

Several security measures have been implemented:

**4.4.1. Password Hashing:** bcryptjs is used to hash user passwords before storing them in the database, protecting them even if the database is compromised.

**4.4.2. JWT for Session Management:** Stateless authentication using JWTs prevents session hijacking issues common with session cookies and scales better. Tokens have an expiration time (1 hour).

**4.4.3. Input Validation:** Both client-side (using react-hook-form) and server-side validation are performed on user inputs to ensure data integrity and prevent common injection attacks (though the primary defense against XSS would be React's default escaping). The custom User ID format (L-XXXXXX) is also enforced.



**4.4.4. Brute-Force Attack Prevention:** The login endpoint limits the number of failed login attempts (5 attempts) before temporarily locking an account (for 3 minutes), mitigating brute-force attacks.

**4.4.5. SQL Injection Prevention (via ORM):** The use of Prisma ORM significantly reduces the risk of SQL injection attacks, as it typically uses parameterized queries or properly escapes inputs.

## 5. CONCLUSION

### 5.1. Summary of Achievements

The LogiDocs project successfully developed a functional web-based document management system tailored for an intermediate logistics company. All core objectives outlined in the introduction have been met:

- A secure login system with custom User IDs and brute-force protection is in place.
- Users can create, list, view, update, and delete logistics operations.
- A central repository for global partner companies with full CRUD functionality has been implemented.
- The system allows assigning these global companies to operations with specific roles.
- A hierarchical document upload and management system, linked to operation participants, is functional, storing files on the server and metadata in the database.
- The user interface provides list views with search, sorting, and pagination, and detail views with clear, organized information, including collapsible sections for document lists.
- User feedback is provided via toast notifications.
- Form handling and validation are managed efficiently using react-hook-form.

### 5.2. Strengths of the System

- **Tailored to Specific Needs:** The system directly addresses the workflow and hierarchical data structure (Operation -> Participant -> Document) of the target logistics company.
- **User-Friendly Interface:** Focuses on simplicity and ease of use for core tasks.
- **Improved Efficiency:** Centralization and quick access to documents are expected to significantly reduce time spent on manual searches and processing.
- **Enhanced Data Integrity & Organization:** Structured storage minimizes data redundancy and ensures documents are consistently organized.
- **Security:** Implements essential security practices like password hashing, JWT authentication, and brute-force prevention.

- **Scalable Technology Stack:** The use of Node.js, React, and PostgreSQL provides a modern, scalable foundation.
- **Cost-Effective:** Built with open-source technologies, offering a potentially lower TCO compared to commercial alternatives.

### 5.3. Limitations and Weaknesses

- **No Advanced RBAC:** Currently, all authenticated users have the same level of access. Granular role-based permissions (Admin vs. Standard User) are not yet implemented.
- **Limited Search Functionality:** Search is available for operation names/numbers and company details, but not for document content or advanced metadata filtering.
- **No Document Versioning:** The system does not currently support version control for uploaded documents.
- **Manual Processes Still Required:** Document upload is manual; no automated fetching from emails or other sources.
- **Reporting and Analytics:** Lacks advanced reporting or analytics features on operational data or document usage.
- **Testing Coverage:** While functionally tested during development, comprehensive automated unit, integration, and E2E tests are not yet in place.
- **UI/UX Refinements:** While functional, the UI could benefit from further polishing, a professional theme, or integration with a more comprehensive UI library for consistency. Loading indicators could be more sophisticated (e.g., spinners).

### 5.4. Future Work and Potential Enhancements

The LogiDocs system has a strong foundation for future enhancements:

- **Implement Role-Based Access Control (RBAC):** Define Admin and User roles with distinct permissions.
- **Advanced Search and Filtering:** Introduce document content search (e.g., using Elasticsearch or a similar indexing service after OCR), and more advanced filtering options for lists.

- **Document Versioning:** Allow users to upload new versions of documents while retaining access to previous versions.
- **Audit Trails:** Log user activities (creations, updates, deletions, downloads) for accountability and security.
- **Dashboard and Reporting:** Create a dashboard summarizing key metrics and provide basic reporting capabilities.
- **Email Integration (Advanced):** Explore options to automatically parse attachments from specific email addresses or threads related to an operation.
- **OCR Integration:** Implement OCR to extract text from image-based documents (like scanned PDFs) for indexing and search.
- **Workflow Automation:** Introduce simple approval workflows for documents or operations.
- **Notifications:** Implement in-app or email notifications for important events (e.g., new document uploaded, operation status change).
- **Enhanced UI/UX:** Integrate a UI component library (e.g., Ant Design, Material-UI) for a more polished and consistent look and feel. Add more sophisticated loading indicators.
- **Comprehensive Testing:** Develop a suite of automated tests.

## 5.5. Overall Contribution

LogiDocs successfully demonstrates a practical and effective solution to a common problem faced by small to medium-sized logistics companies: inefficient management of operational documents. By providing a centralized, structured, and secure web-based platform, the system has the potential to significantly enhance operational efficiency, reduce errors, improve document traceability, and support better decision-making. The project also serves as a valuable learning experience in full-stack web development, showcasing the integration of modern backend and frontend technologies to deliver a real-world application. It provides a solid, customizable alternative to more generic or expensive commercial DMS solutions for companies with similar specific needs.

## 6. REFERENCES

- **Technology Documentation:**

- Node.js. (n.d.). *About Node.js®*. Retrieved from <https://nodejs.org/en/about/>
- Express. (n.d.). *Express - Node.js web application framework*. Retrieved from <https://expressjs.com/>
- PostgreSQL. (n.d.). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Retrieved from <https://www.postgresql.org/>
- Prisma. (n.d.). *Next-generation Node.js and TypeScript ORM*. Retrieved from <https://www.prisma.io/>
- React. (n.d.). *A JavaScript library for building user interfaces*. Retrieved from <https://reactjs.org/>
- Vite. (n.d.). *Next Generation Frontend Tooling*. Retrieved from <https://vitejs.dev/>
- JSON Web Tokens. (n.d.). *Introduction to JSON Web Tokens*. Retrieved from <https://jwt.io/introduction>
- Bcrypt.js GitHub Repository. (n.d.). Retrieved from <https://github.com/kelektiv/node.bcrypt.js>
- React Hook Form. (n.d.). *Performant, flexible and extensible forms with easy-to-use validation*. Retrieved from <https://react-hook-form.com/>
- React Toastify. (n.d.). *React-Toastify*. Retrieved from <https://fkhadra.github.io/react-toastify/introduction>
- Axios GitHub Repository. (n.d.). Retrieved from <https://github.com/axios/axios>
- React Router. (n.d.). *Declarative routing for React applications*. Retrieved from <https://reactrouter.com/>

- **Conceptual Articles / Books**

- [1] A. Zakrivashevich, "How Document Management Systems Allow Logistics Companies Adapt to Digital Transformation," XB Software, 2021.

- [2] I. Vrečer, J. Marolt, M. Borstnar, and B. Gajšek, "Managing Document Management Systems' Life Cycle in Relation to Organizational Maturity," *Sustainability*, vol. 15, no. 7, p. 6171, 2023.
- [3] M. Ahmadi Achachlouei, O. Patil, T. Joshi, and V. Nair, "Document Automation Architectures and Technologies: A Survey," *arXiv preprint*, arXiv:2105.09965, 2021.

- **Similar Systems or Case Studies**

- [4] Document Logistix, "Case Study: DHL Streamlines Global Document Processes," [Online]. Available: <https://www.document-logistix.com>
- [5] DocuWare, "GSC Logistics Case Study," [Online]. Available: <https://start.docuware.com>
- [6] Kyocera Document Solutions, "Document Management in the Logistics and Transport Industry," [Online]. Available: <https://www.kyoceradocumentsolutions.eu>

## **7. APPENDIX**

All source code used in this study is available at the following GitHub repository:

- <https://github.com/AytekSangun/LojistikEvrakYonetimi>