



BHARAT ELECTRONIC LIMITED
A Govt. of India (Ministry of Defence)
P.O. BHARAT – 201010
GHAZIABAD (U.P.) INDIA
Website : <http://www.bel-india.in>

Internship Project Report

On

“Organization Chart System with Excel Integration and Hierarchy Visualization”

Submitted in partial fulfillment for award of

BACHELOR OF TECHNOLOGY

Degree

In

COMPUTER SCIENCE & ENGINEERING

2025-26

Under the Guidance of:

Mr. Ashish Mittal

D&E / Antenna Department

BEL Ghaziabad

Submitted By:

Neeraj Chaurasiya

(Roll no – 2200330100146)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY
DELHI-MEERUT ROAD, GHAZIABAD

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

Phase 1: Local Storage Based Frontend Application

1. Project Title:

Interactive Organization Chart with Excel Upload and Local Storage Visualization

2. Abstract:

This phase of the Organization Chart project was focused on building a fully client-side solution to visualize the hierarchical structure of an organization. Using only frontend technologies, the application allowed users to upload Excel files containing employee data (e.g., Staff No, Name, Designation, Parent) and render this data in a visually engaging organizational tree using the vis-network library. **The data was stored locally using localStorage**, enabling temporary persistence without any server or backend.

3. Objectives:

(1). Excel Upload:

Users can upload an Excel file with employee details like name, staff number, designation, and manager (parent). This helps in bulk data entry.

(2). Data Parsing with XLSX.js:

The file is read using xlsx.js and converted to JSON format. Unwanted spaces and irregular keys are cleaned for consistency.

(3). Local Storage:

Cleaned data is saved in the browser's localStorage, so it stays even after page reload (unless manually cleared).

(4). Tree Rendering with vis-network:

The organization structure is shown as a tree where each box is an employee and lines show reporting relationships.

(5). Extra Features:

Includes zoom, drag, search by name/ID, export as Excel, dark mode toggle, and a popup with employee details on click.

4. Tools & Technologies Used:

- HTML, CSS, JavaScript
- XLSX.js
- vis-network
- localStorage

5. System Workflow:

1. The user uploads an Excel file containing employee details.
2. The **XLSX.js** library parses the file and converts it into a clean JSON format.
3. The parsed data is cleaned (extra spaces removed, keys standardized) and stored in **localStorage**.
4. The **vis-network** library reads this data and renders a dynamic organization chart, using Staff No as unique identifiers and Parent to build reporting relationships.

6. Features Implemented:

(1). Excel File Upload:

Users can upload Excel sheets containing employee hierarchy data.

(2). Tree View Visualization:

A dynamic, interactive tree is rendered to represent the reporting structure of employees.

(3). Search Functionality:

Allows users to search for employees by name or Staff No., and highlights the matching node.

(4). Zoom, Drag and Fullscreen View:

Users can zoom in/out, drag the chart to navigate, and switch to fullscreen for better visibility.

(5). Export Options:

Filtered data can be downloaded as an Excel file for backup or further use.

(6). Designation-based Coloring:

Each designation is assigned a unique background color for quick visual distinction.

(7). Popup with Detailed Employee Info:

Clicking on a node shows a popup with detailed information like name, ID, designation, reporting manager, and project roles

7. Limitations:

- Data is not persistent across different systems.
- localStorage gets cleared on browser cache removal.

8. Conclusion:

This phase laid the groundwork for visualizing organizational hierarchy from Excel data using frontend technologies. It helped validate the structure and interface but lacked persistence and cross-system support.

Phase 2: Dynamic Organization Chart with Backend Integration

1. Project Title:

Dynamic Employee Organization Chart with Backend and MySQL Integration

2. Abstract:

This phase extended the previous frontend-only project into a robust full-stack system with Node.js as backend and MySQL as the database. The solution offers persistent, scalable storage and allows real-time syncing of data across systems. All previously frontend-only features are retained and enhanced with backend support, enabling multi-user access and offline operability.

3. Objectives:

- Develop backend with Node.js and Express.js.
- Use MySQL to store employee hierarchy persistently.
- Replace localStorage with API-based architecture.
- Keep frontend features while adding backend functionality.
- Enable offline working using caching.

4. Tools & Technologies Used:

- **Frontend:** HTML, CSS, JavaScript, XLSX.js, vis-network
- **Backend:** Node.js, Express.js
- **Database:** MySQL
- **Tools:** dotenv, Postman, npm

5. Backend Functionality:

- REST APIs developed for Excel Upload (/upload) and Data Fetch (/data).
- Upload route parses Excel, cleans data, and inserts into MySQL.
- Data route fetches the full organizational chart from MySQL.
- All credentials are managed through a .env file.

6. MySQL Table Structure:

- ID (Auto-Increment)
- Employee Name
- Staff Number (Unique)
- Designation
- Parent (Reporting Manager)
- Project-1, Role-1, Project-2, Role-2, Project-3, Role-3

7. Frontend Functionality:

- Uses API to retrieve org data
- Renders updated hierarchy instantly
- Popup includes projects/roles and designation
- Export and search remain functional

8. Key Features:

- Backend Persistence
- Excel Upload + Live Sync
- Multi-device Access
- Works Offline
- Scalable Architecture

9. Challenges Faced:

- Structuring backend routes for Excel parsing
- Mapping relationships between employees
- Ensuring data sync in offline conditions
- Managing Excel format variations
- Maintaining frontend layout with growing data

10. Future Enhancements:

- Admin/User Login System
- Real-time Sync using WebSocket
- Upload Profile Photos
- Add/Edit/Delete Employees without Excel
- Mobile Friendly Interface

11. User Interface Snapshots

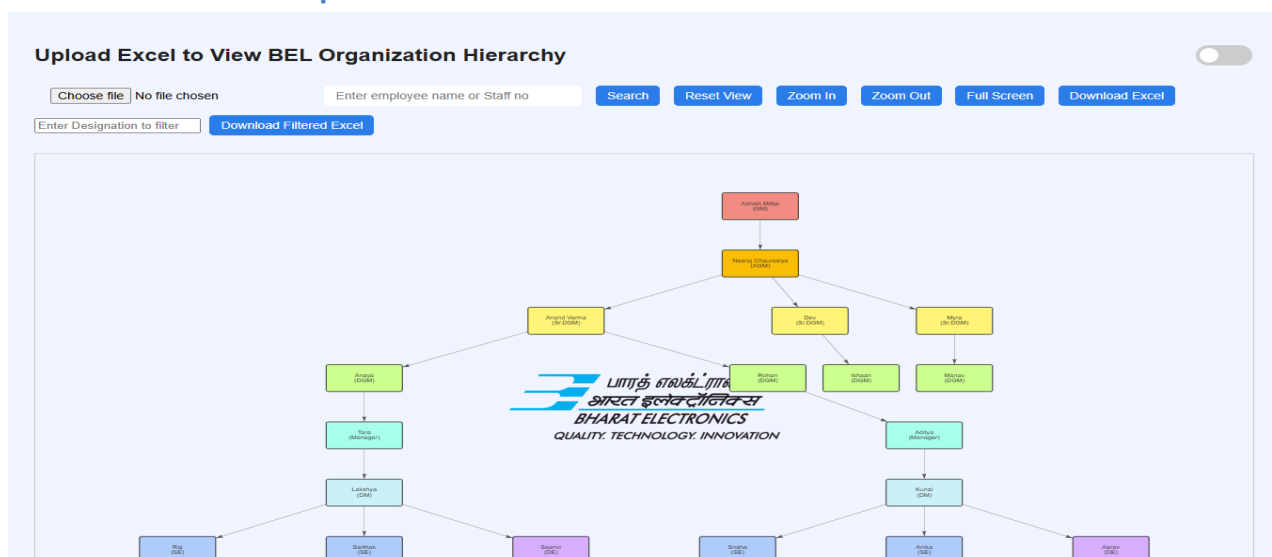


Figure 1: Organization Tree View

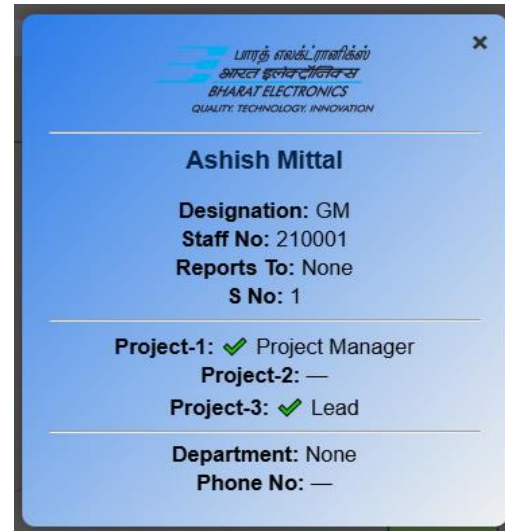


Neeraj Chaurasiya

Designation: AGM
Staff No: 210002
Reports To: 210001
S No: 2

Project-1: ✓ Lead
Project-2: ✓ Project Manager
Project-3: —

Department: None
Phone No: —



Ashish Mittal

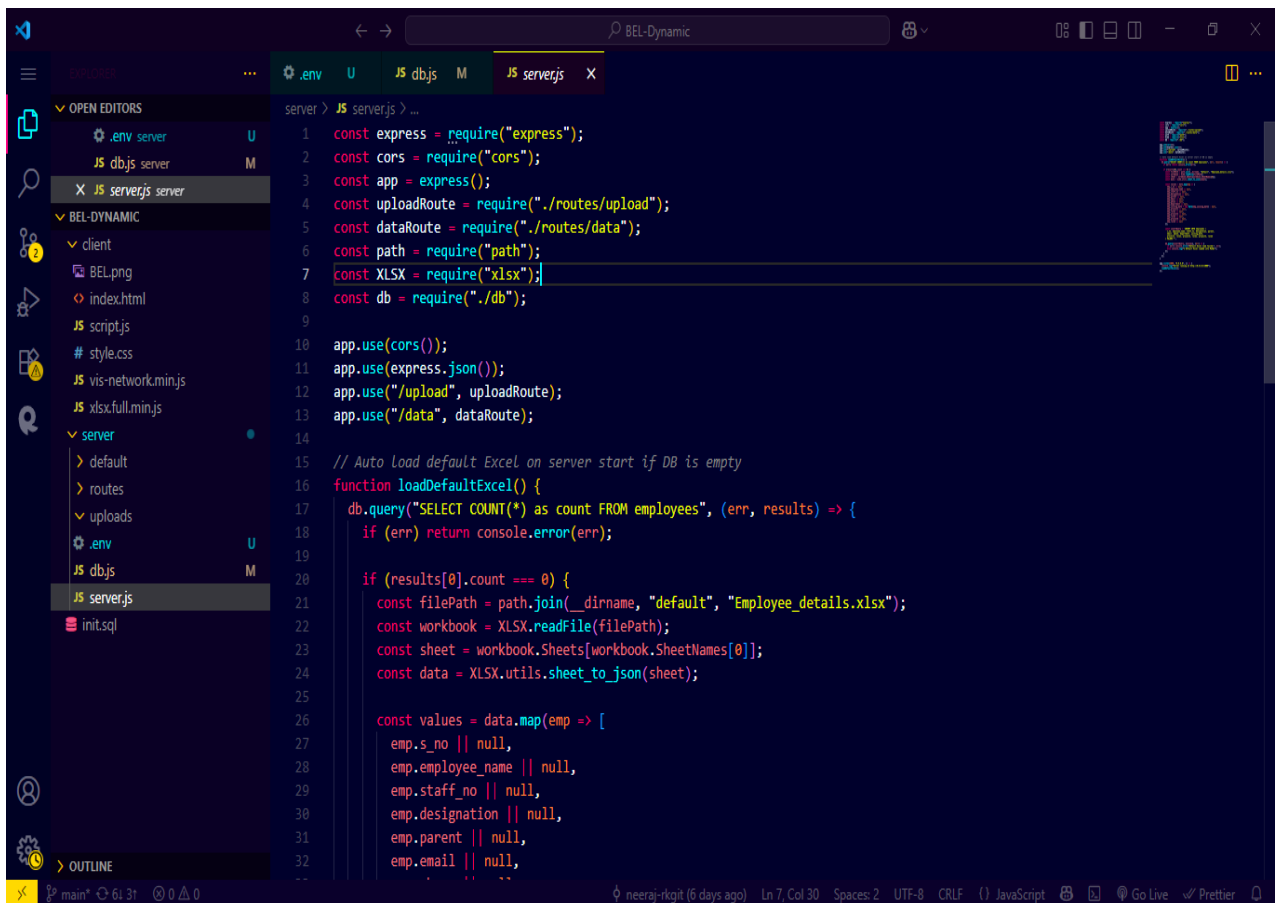
Designation: GM
Staff No: 210001
Reports To: None
S No: 1

Project-1: ✓ Project Manager
Project-2: —
Project-3: ✓ Lead

Department: None
Phone No: —

Figure 2: Employee Detail Popup Modal

12. Code Snippets:



```

server > JS server.js > ...
1  const express = require("express");
2  const cors = require("cors");
3  const app = express();
4  const uploadRoute = require("./routes/upload");
5  const dataRoute = require("./routes/data");
6  const path = require("path");
7  const XLSX = require("xlsx");
8  const db = require("./db");
9
10 app.use(cors());
11 app.use(express.json());
12 app.use("/upload", uploadRoute);
13 app.use("/data", dataRoute);
14
15 // Auto Load default Excel on server start if DB is empty
16 function loadDefaultExcel() {
17   db.query("SELECT COUNT(*) as count FROM employees", (err, results) => {
18     if (err) return console.error(err);
19
20     if (results[0].count === 0) {
21       const filePath = path.join(__dirname, "default", "Employee_details.xlsx");
22       const workbook = XLSX.readFile(filePath);
23       const sheet = workbook.Sheets[workbook.SheetNames[0]];
24       const data = XLSX.utils.sheet_to_json(sheet);
25
26       const values = data.map(emp => [
27         emp.s_no || null,
28         emp.employee_name || null,
29         emp.staff_no || null,
30         emp.designation || null,
31         emp.parent || null,
32         emp.email || null,
33         ...

```

Code Snippet 1: Upload API Route

```

server > JS db.js
1  const mysql = require("mysql2"); // ← MySQL Connector here
2  require("dotenv").config();
3
4  const db = mysql.createConnection({
5    host: process.env.DB_HOST,
6    user: process.env.DB_USER,
7    password: process.env.DB_PASSWORD,
8    database: process.env.DB_NAME,
9  });
10
11 db.connect((err) => {
12   if (err) throw err;
13   console.log("✓ Connected to MySQL");
14 });
15
16 module.exports = db;
17

```

Code Snippet 2: Database Connection Setup (db.js)

```

<body>
  <div class="header">
    <h2>Upload Excel to View BEL Organization Hierarchy</h2>
    <label class="theme-switch" title="Toggle Day/Night Mode">
      <input type="checkbox" id="toggleTheme" />
      <span class="slider"></span> 🌙
    </label>
  </div>

  <input type="file" id="upload" value="Loading... .xlsx" title="Upload Excel File" />
  <input type="text" id="searchBox" placeholder="Search employee..." title="Type employee name or ID" />
  <button onclick="searchNode()">Search</button>
  <button onclick="resetView()">Reset View</button>
  <button onclick="zoomIn()">Zoom In</button>
  <button onclick="zoomOut()">Zoom Out</button>
  <button onclick="toggleFullScreen()">Full Screen</button>
  <button onclick="downloadExcel()">Download Excel</button>

  <div id="network"></div>

  <div id="popup" class="modal">
    <div class="modal-content">
      <span id="close"></span>
      <div id="popupLogo"></div>
      <div id="popupDetails"></div>
    </div>
  </div>

  <script src="xlsx.full.min.js"></script>
  <script src="vis-network.min.js"></script>
  <script src="script.js"></script>
</body>

```

Code Snippet 3: User Interface Structure

13. Conclusion:

With the backend and database integration, the project reached a production-ready stage where it can be deployed in real organizational setups. Data can be maintained reliably and synced across users and sessions.