TABLE OF CONTENTS

ABSTRACT

In today's rapidly evolving industrial landscape, digital transformation is essential for improving efficiency, accuracy, and productivity. Traditional manual data logging and reporting systems often lead to human errors, inefficiencies, and delays in decision-making. To overcome these challenges, this project introduces an Industrial Data Logging and Reporting System, which integrates real-time data collection, database storage, and automated report generation to streamline factory operations.

The system consists of three primary modules: the Stores Module, Test Booth Module, and Assembly Module. The Stores Module manages inventory-related data such as raw materials, TPL numbers, and purchase orders. The Test Booth Module records crucial test data, including component performance metrics and quality control parameters. Lastly, the Assembly Module tracks essential manufacturing details such as PLC models, VFD configurations, and other assembly-related specifications.

The backend of the system is developed using Node.js and Express.js, ensuring seamless communication with a MySQL database. The frontend, built with HTML, CSS, and JavaScript, provides an intuitive user interface for factory personnel. All collected data is securely stored in a centralized database and can be exported as structured Excel or PDF reports using ExcelJS and PDFKit, making data management more efficient and accessible.

Key features of this system include automated data entry with real-time updates, seamless database integration, automated report generation, improved accuracy, and a user-friendly interface. These features significantly enhance data visibility, traceability, and operational efficiency, ultimately reducing errors and optimizing production workflows.

By implementing this system, manufacturing units can transition to a data-driven and intelligent industrial ecosystem, enabling better decision-making and improved productivity. Future enhancements may include IoT integration, AI-based predictive maintenance, and cloud storage solutions, further enhancing scalability and automation. This project demonstrates the transformative potential of automation in manufacturing and lays the foundation for a more efficient and digitally connected factory environment.

1. INTRODUCTION

In the era of Industry 4.0, digital transformation has become essential for manufacturing industries to enhance efficiency, accuracy, and productivity. Traditional manual data entry and record-keeping methods often lead to human errors, delays, and inefficiencies, making it difficult to track and analyze production data effectively. To address these challenges, this project introduces an Industrial Data Logging and Reporting System, designed to streamline data collection, storage, and reporting in a smart manufacturing environment.

The system consists of three key modules: the Stores Module, which manages inventory by tracking TPL numbers, purchase orders, CFM values, and tank details to ensure efficient material handling; the Test Booth Module, which records performance test data of machinery components, enabling real-time quality checks and historical data analysis; and the Assembly Module, which maintains detailed assembly information, including PLC models, VFD configurations, and fan motor details, ensuring accurate product assembly documentation.

Developed using Node.js and Express.js for backend functionality, the system utilizes MySQL for structured data storage, while the frontend is built with HTML, CSS, and JavaScript, offering an intuitive and user-friendly interface. Additionally, it features automated Excel and PDF report generation using ExcelJS allowing seamless data visualization and analysis.

By integrating real-time data entry, database connectivity, and automated reporting, this system significantly reduces human errors, enhances traceability, and improves decision-making in manufacturing processes. As a smart factory solution, it not only optimizes operations but also lays the foundation for future integration of IoT and AI-driven predictive analytics, making factories more digitally connected and intelligent.

1.1 OVERVIEW OF THE PROJECT

The Automated Digital Factory Data Logging and Reporting System is designed to enhance efficiency, accuracy, and productivity in manufacturing by automating data collection, storage, and reporting. Traditional manual record-keeping methods often lead to inefficiencies, human errors, and delays in decision-making. To address these challenges, this system provides a real-time, automated, and centralized solution that improves data accuracy, traceability, and operational efficiency.

The system is structured around three key modules: the Stores Module, which manages inventory tracking, including TPL numbers, purchase orders, and material details; the Test Booth Module, which records machinery performance test data, enabling real-time quality control and component validation; and the Assembly Module, which maintains detailed assembly specifications such as PLC models, VFD configurations, and motor details to ensure consistency in the production process.

To achieve seamless integration, the backend of the system is built using Node.js and Express.js, allowing efficient data handling and communication with a MySQL database for secure storage. The frontend is developed using HTML, CSS, and JavaScript, providing an intuitive and user-friendly interface for factory personnel. Additionally, the system includes automated report generation using ExcelJS enabling structured data to be exported in Excel and PDF formats for analysis and documentation.

Key features of this system include real-time automated data logging, centralized database connectivity, automated Excel report generation, an intuitive web interface, and improved traceability. These functionalities significantly enhance data visibility, minimize manual errors, and optimize production workflows.

By implementing this system, manufacturing industries can transition toward a smart and digital manufacturing environment, leading to improved efficiency and data-driven decision-making.

1.2 ORGANIZATION PROFILE

YBI Foundation is a Delhi-based not-for-profit edutech organization established in October 2020. Operating as a Section 8 company under the Companies Act, it focuses on empowering youth by providing education and training in emerging technologies. The foundation offers a blend of online and offline learning opportunities, emphasizing a "Learn Anywhere, Learn Anytime" approach to make education accessible to a diverse audience.

YBI Foundation's mission is to bridge the gap between academic knowledge and industry requirements. They provide free online instructor-led courses and internships designed to equip learners with practical skills in areas such as data science, business analytics, machine learning, cloud computing, and big data. The organization caters to a wide range of individuals, including college students, recent graduates, and working professionals from various domains like B.Tech, M.Tech, BCA, MCA, BBA, MBA, and diploma holders.

To date, YBI Foundation has trained over 200,000 interns, transforming them into industry-ready professionals. Their programs are beginner-friendly and emphasize project-based learning, offering hands-on experience through industry capstone projects. They also provide live one-on-one doubt resolution sessions with mentors, resume building assistance, and interview preparation. Upon completion of their programs, participants receive digitally verified internship certificates, which are recognized by both academic institutions and industry employers.

The foundation's commitment to quality education and skill development has positioned it as a trusted brand in the edutech sector, aiming to support learners in achieving their academic and professional goals.

2. SYSTEM SPECIFICATION

2.1 HARDWARE REQUIREMENT

Processor : intel core i3

RAM : 2 GB RAM or more

2.1.1 SOFTWARE REQUIREMENT

- Windows 10
- HTML/CSS/Java Script
- Excel
- MySQL Workbench 8.0 CE

2.2 SOFTWARE DESCRIPTION

- **HTML/CSS/JAVASCRIPT:**
  HTML structures web content, CSS styles it, and JavaScript adds interactivity. Together, they are the fundamental building blocks of web development, enabling the creation of dynamic, visually appealing websites.
- **EXCEL:**
  Microsoft Excel is a powerful spreadsheet tool used for data organization, analysis, and visualization. It provides functions like formulas, pivot tables, and charts to manage and analyse data efficiently in various business and personal contexts.
- **DATABASE MANAGEMENT SYSTEM:**
  A structured relational database is implemented to securely store all manufacturing-related data, ensuring efficient management and retrieval of information. MySQL, a reliable and scalable relational database, is used to handle data for stores, test booths, and assembly stations, providing a robust foundation for the system. Additionally, the MySQL2 library for Node.js facilitates seamless communication between the backend and the database, enabling efficient query execution and data retrieval for real-time processing and reporting.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In traditional manufacturing environments, data logging, inventory management, and quality control are often handled manually using paper records or standalone spreadsheets. This approach leads to several inefficiencies, including human errors, data redundancy, lack of real-time updates, and difficulties in tracking production details. The existing system lacks centralized data storage, making it challenging to retrieve historical records for analysis. Additionally, generating reports manually is time-consuming and prone to inconsistencies. Without automation, decision-making is delayed, and operational inefficiencies hinder productivity and quality assurance.

DISADVANTAGES

Prone to Human Errors – Manual data entry increases the likelihood of mistakes, leading to inaccurate records and potential production issues.

Time-Consuming – Data collection, validation, and report generation require significant time and effort, slowing down decision-making.

Lack of Real-Time Updates – Since data is recorded manually, there is no real-time synchronization, making it difficult to track production and inventory status instantly.

Data Redundancy and Loss – Paper-based or spreadsheet systems are prone to duplication, misplacement, or loss, leading to inconsistencies in records.

Limited Accessibility – Data stored in physical records or standalone files is not easily accessible across departments, affecting collaboration and efficiency.

Inefficient Reporting – Generating reports manually is cumbersome and may lead to inconsistencies, making it difficult to analyze and interpret production trends.

Poor Traceability – Tracking defects, maintenance logs, and historical production data is challenging, impacting quality control and compliance with industry standards.

Scalability Issues – As production scales, managing large volumes of data manually becomes unsustainable, leading to bottlenecks and inefficiencies.

3.2 PROPOSED SYSTEM

The Automated Digital Factory Data Logging and Reporting System is designed to overcome the limitations of the existing manual system by integrating real-time data collection, structured database management, and automated report generation. This system ensures efficiency, accuracy, and reliability in manufacturing operations.

ADVANTAGES

Automated Data Entry – Eliminates human errors by capturing and storing manufacturing data digitally.

Real-Time Updates – Enables instant data synchronization, allowing departments to access the latest production and inventory details.

Centralized Database – Uses MySQL for secure and structured data storage, reducing redundancy and improving accessibility.

Seamless Communication – Developed using Node.js and Express.js for efficient backend processing, ensuring smooth data exchange between modules.

User-Friendly Interface – Built with HTML, CSS, and JavaScript, providing an intuitive and responsive UI for easy data entry and retrieval.

Automated Report Generation – Utilizes ExcelJS to generate structured Excel and PDF reports for better data analysis and decision-making.

Improved Traceability – Tracks inventory, test results, and assembly details, making it easier to monitor defects and production history.

Scalability and Future Expansion – Can be enhanced with IoT integration, cloud storage, and AI-driven predictive analytics for smarter manufacturing.

4. SYSTEM DESIGN

4.1 MODULE DESCRIPTION

The system consists of three key modules, each responsible for specific manufacturing processes:

Login Module

Implements user authentication with role-based access control (RBAC) to ensure secure system access. Users can log in using unique credentials, with different access levels for administrators, supervisors, and operators.

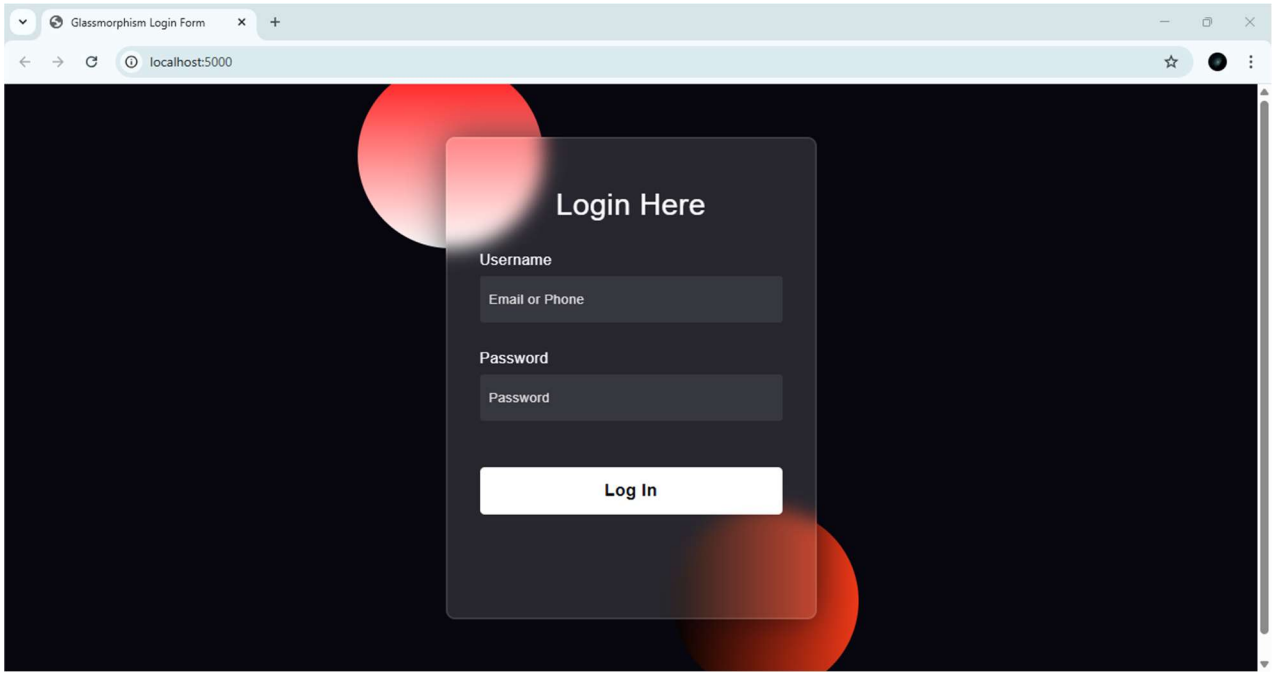


Fig 1: Login Page

The Login Module ensures secure access with user authentication and role-based control. Users enter credentials, which are verified against a MySQL database using Node.js and MySQL2. Roles (Admin, Store Manager, Operator) define access permissions. Sessions are managed via express-session, preventing repeated logins. Security features include password encryption, SQL injection prevention, and account lockout for failed attempts. This module enhances data security, prevents unauthorized access, and maintains system integrity.

Stores Module

Manages inventory details, including TPL numbers, purchase orders (PO), CFM values, and tank/canopy details. Ensures accurate tracking of incoming and outgoing materials.

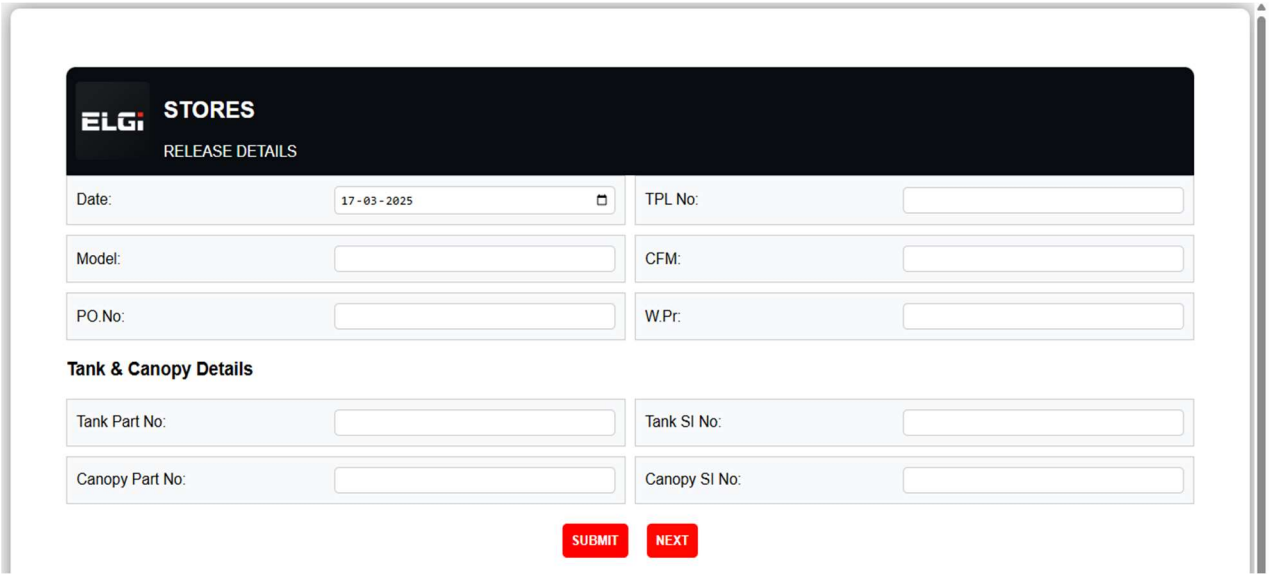


Fig 2: Store Page

The Store Module manages inventory and material tracking in the digital factory system. It records essential details such as TPL numbers, purchase orders (PO), CFM values, and tank/canopy details. Data is stored in a MySQL database, ensuring accurate tracking of incoming and outgoing materials. The module integrates with the frontend (HTML, CSS, JavaScript) for easy data entry and the backend (Node.js, Express.js, MySQL2) for seamless database interactions. It supports real-time inventory updates, data validation, and automated report generation, improving efficiency and reducing manual errors.

Assembly Module

Records detailed assembly data such as PLC models, VFD configurations, and fan motor specifications. Ensures accurate documentation of assembled products and configurations.

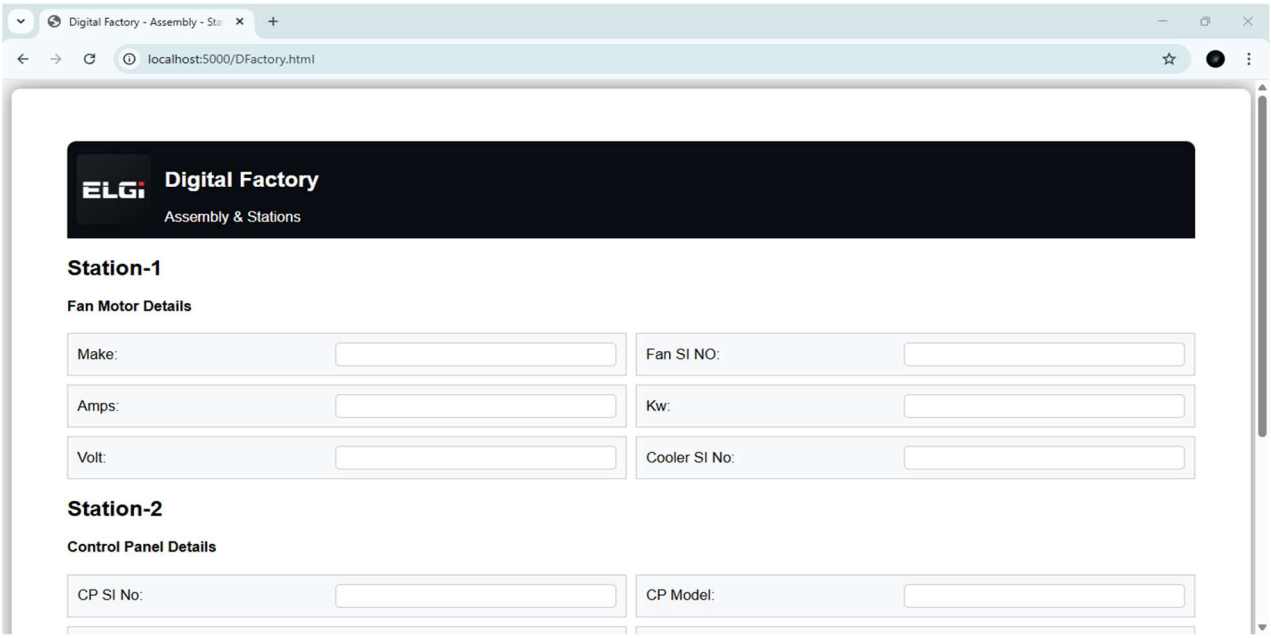


Fig 3: Assembly Page

The Assembly Module records and manages detailed assembly data, ensuring accurate documentation of product configurations. It tracks PLC models, VFD configurations, fan motor specifications, and other essential assembly details. The data is stored in a MySQL database and is accessible through a user-friendly interface built with HTML, CSS, and JavaScript. The backend, developed using Node.js and Express.js, handles data insertion, retrieval, and updates efficiently. This module enables real-time monitoring, reduces manual errors, and supports automated report generation, ensuring a streamlined assembly process.

Test Booth Module

Captures performance test data for components like Airend, Motors, and Tank. Enables real-time quality control and historical test data analysis.

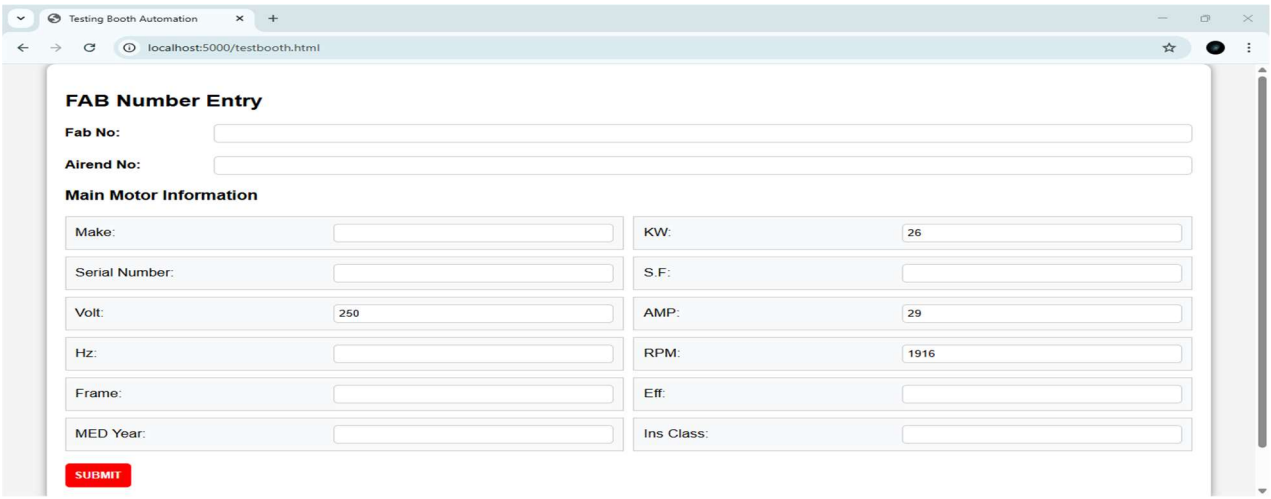


Fig 4: Test booth Page

The Test Booth Module is responsible for capturing and managing performance test data of various components, such as Airend, Motors, and Tanks. It enables real-time quality control and historical test data analysis, ensuring that all tested components meet the required standards before final assembly. Developed with Node.js and Express.js for backend functionality and MySQL for structured data storage, this module securely logs test results. The frontend, built with HTML, CSS, and JavaScript, provides an intuitive interface for users to input and retrieve test data efficiently. Additionally, it supports automated report generation in Excel and PDF formats, ensuring traceability and streamlined decision-making in the testing process.

Database Selection

MySQL Database: Stores structured data related to Stores, Test Booth, and Assembly processes.

MySQL2 (Node.js Library): Enables seamless communication between Node.js backend and the MySQL database for data insertion, retrieval, and updates.

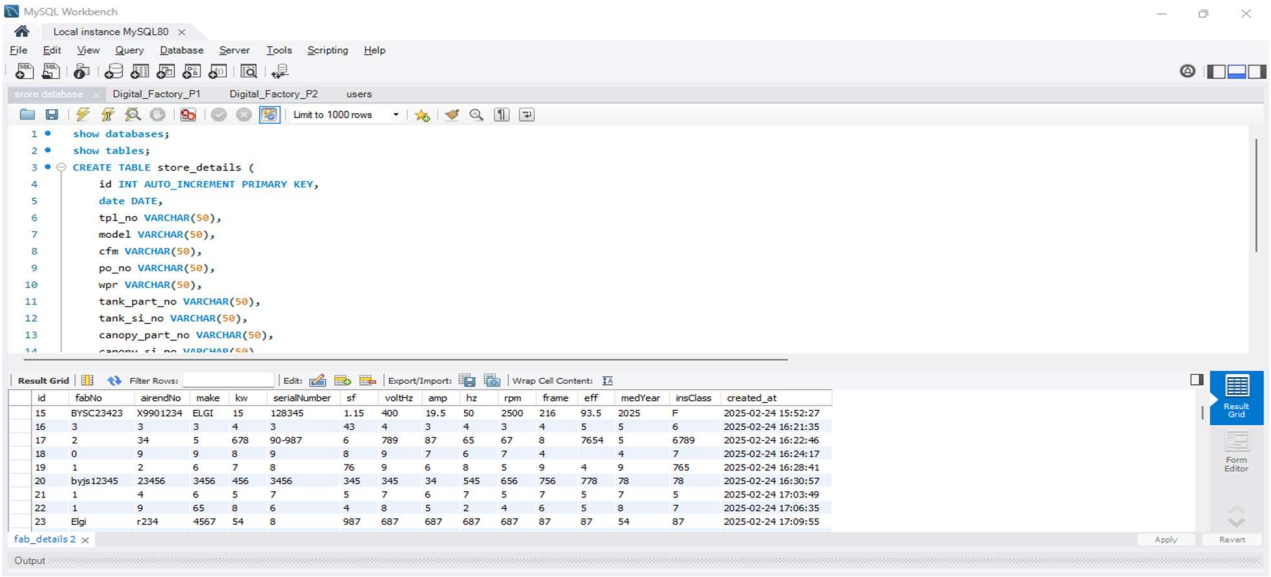
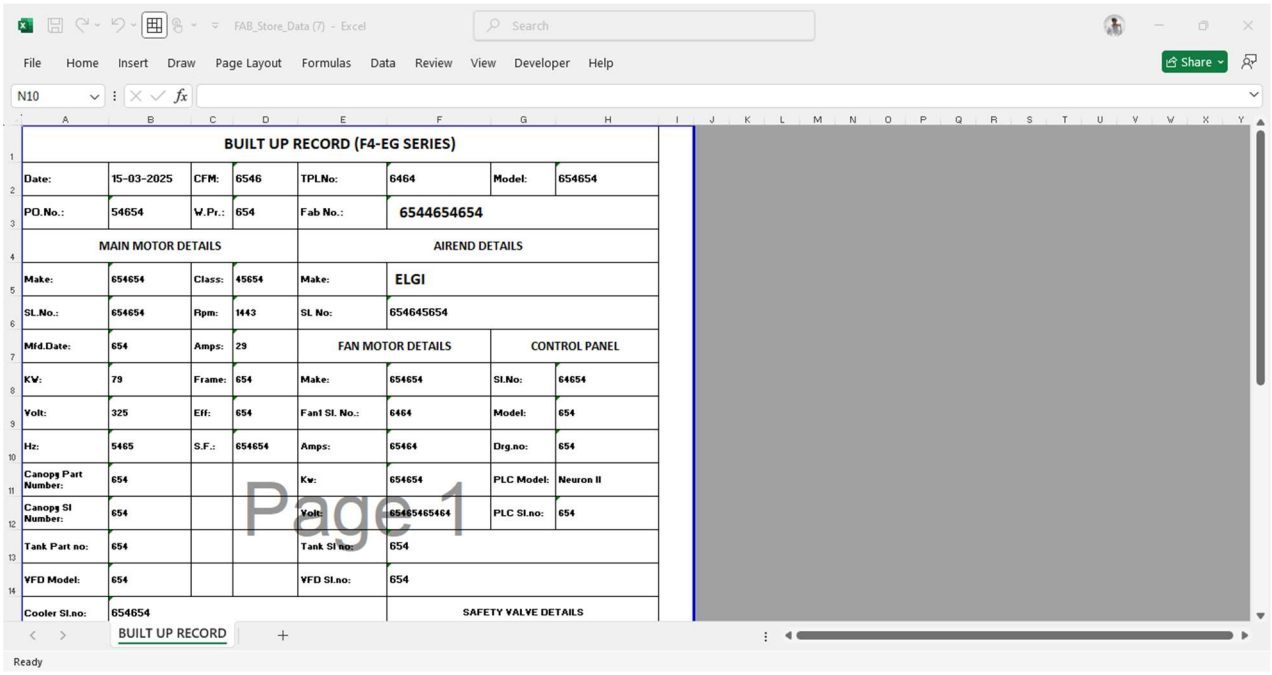


Fig 5: Database Tables

The Database Module is the backbone of the system, responsible for securely storing, managing, and retrieving all manufacturing-related data. It utilizes MySQL as the relational database to maintain structured records for the Stores Module, Assembly Module, and Test Booth Module. The MySQL2 library in Node.js enables seamless interaction between the backend and the database, allowing efficient data insertion, retrieval, and updates. The module ensures data integrity, consistency, and quick access to production data, enabling real-time monitoring and reporting. Additionally, it supports automated data extraction for Excel and PDF report generation, ensuring streamlined record-keeping and decision-making.

Excel Extraction Module

Fetches the latest data from the database. Uses SQL queries to extract relevant information efficiently. Uses the Excel JS library in Node.js to write data into an Excel file. Saves the output as an Excel report, ready for download.



BUILT UP RECORD (F4-EG SERIES)							
Date:	15-03-2025	CFM:	6546	TPLNo:	6464	Model:	654654
PO.No.:	54654	W.Pr.:	654	Fab No.:	6544654654		
MAIN MOTOR DETAILS				AIREND DETAILS			
Make:	654654	Class:	45654	Make:	ELGI		
SL.No.:	654654	Rpm:	1443	SL No:	654645654		
Mfd.Date:	654	Amps:	29	FAN MOTOR DETAILS		CONTROL PANEL	
KW:	79	Frame:	654	Make:	654654	Sl.No:	64654
Volt:	325	Eff:	654	Fan1 Sl. No.:	6464	Model:	654
Hz:	5465	S.F.:	654654	Amps:	65464	Drg.no:	654
Canopy Part Number:	654			Kw:	654654	PLC Model:	Neuron II
Canopy Sl Number:	654			Volt:	65465465464	PLC Sl.no:	654
Tank Part no:	654			Tank Sl no:	654		
VFD Model:	654			VFD Sl.no:	654		
Cooler Sl.no:	654654			SAFETY VALVE DETAILS			

Fig 6: Final Report

The Excel Extraction Module automates the process of retrieving and exporting manufacturing data into structured Excel reports. It fetches the latest records from the MySQL database using optimized SQL queries and utilizes the ExcelJS library in Node.js to generate well-formatted Excel files.This module ensures that critical data from the Stores, Assembly, and Test Booth Modules is accurately transferred to an Excel sheet, maintaining predefined cell placements for better readability. The exported Excel report provides a structured overview of inventory details, assembly configurations, and test results, enabling easy analysis, quality checks, and decision-making.

4.2 DATA FLOW DIAGRAM:

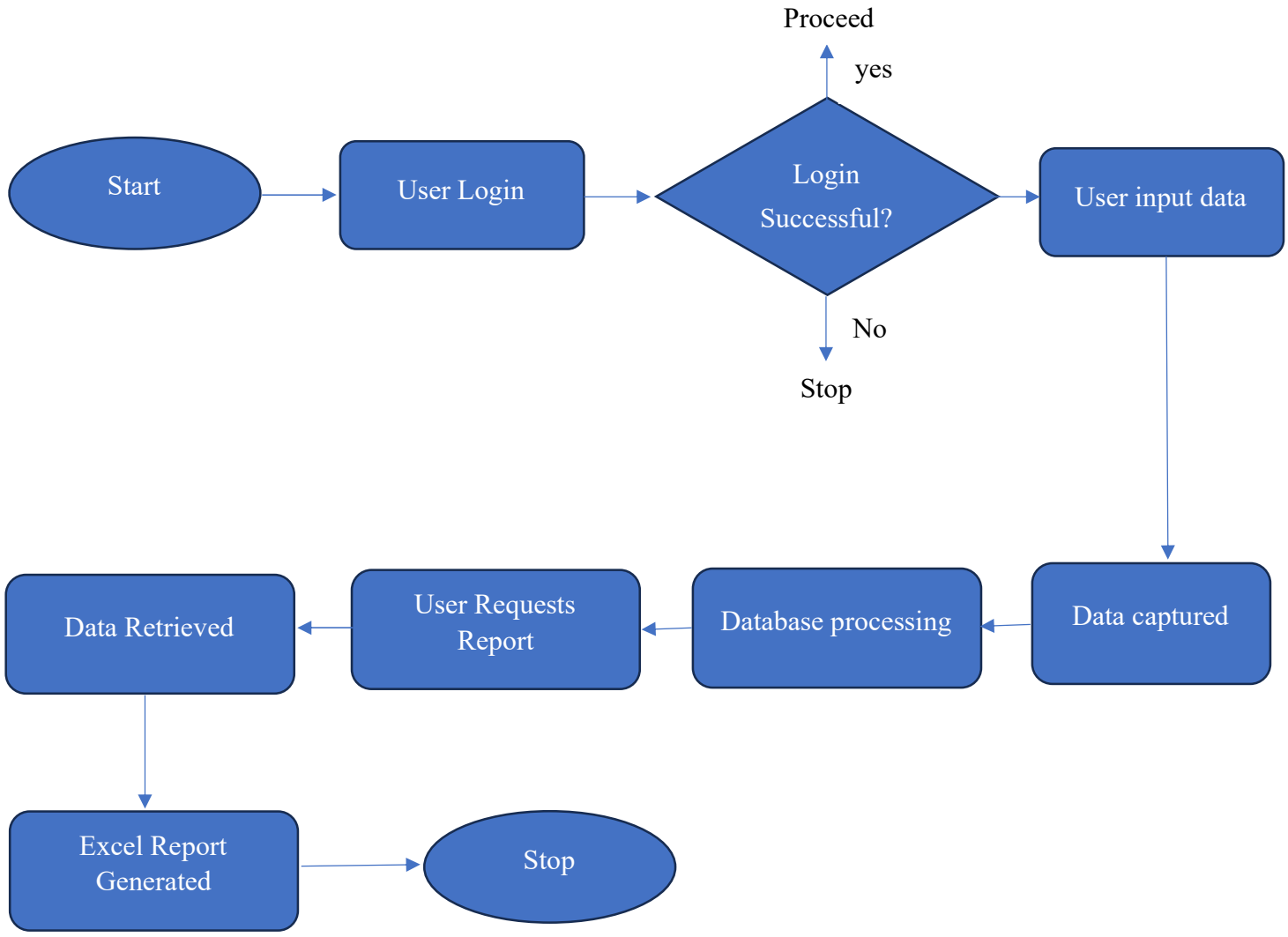


Fig 7: Data Flow Diagram

4.3 INPUT DESIGN

Login page:

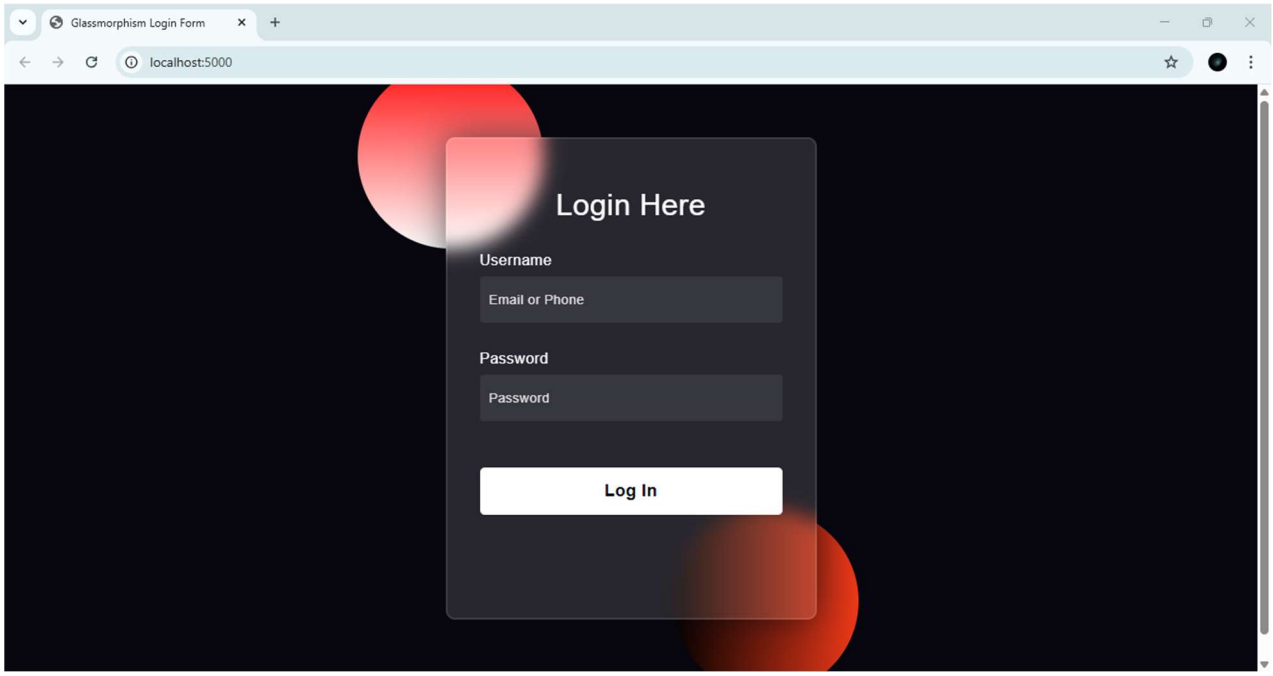


Fig 8: Final Report

The Login Module ensures secure access to the Industrial Data Logging and Reporting System by verifying user credentials before granting access to different modules. The following inputs are required:

Username – The unique identifier assigned to each user.

Password – A secure string required for authentication.

Store Release Details:

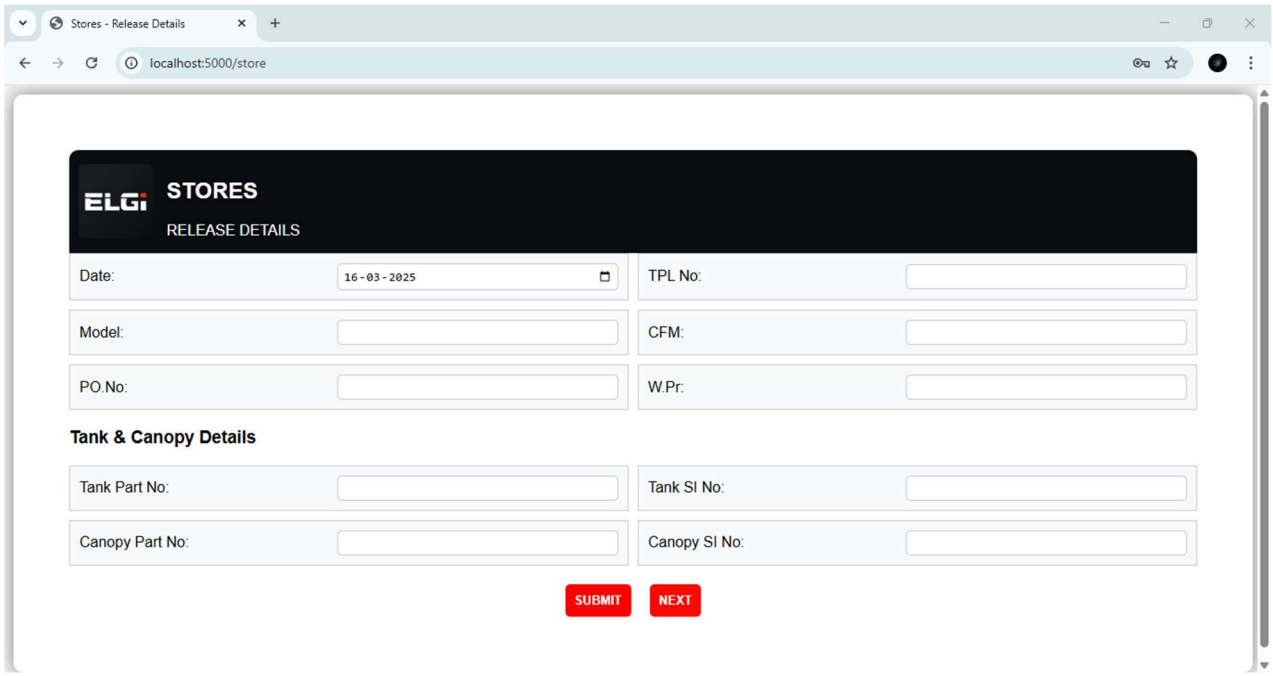


Fig 9: Store Release Page

The Store Details Module is responsible for managing inventory records and ensuring accurate tracking of materials within the factory. It facilitates efficient data logging for materials received, used, and stored.

Assembly & Stations page:

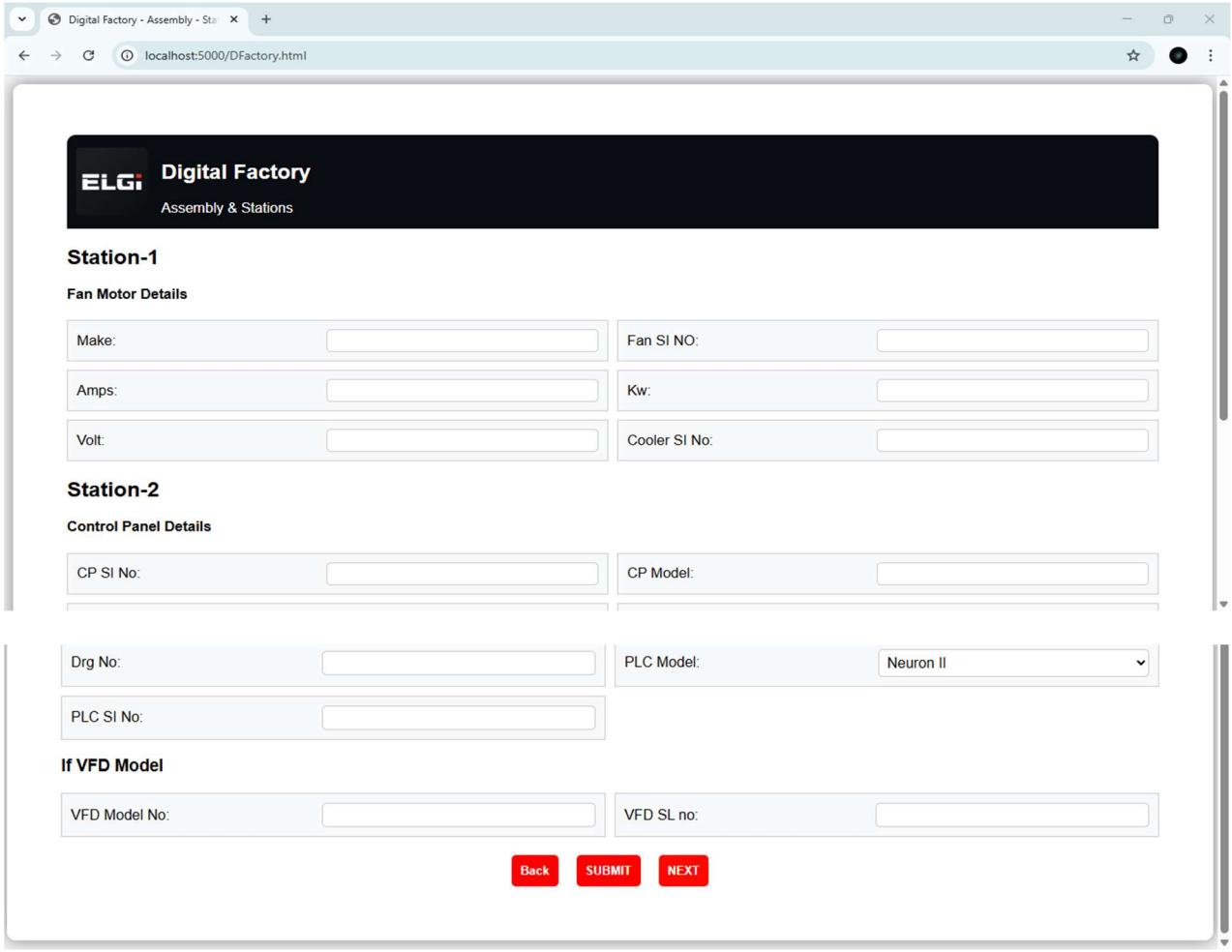


Fig 10: Assembly Page

The Assembly Module records and manages detailed assembly data, ensuring accurate documentation of product configurations.

Testbooth Automation Page:

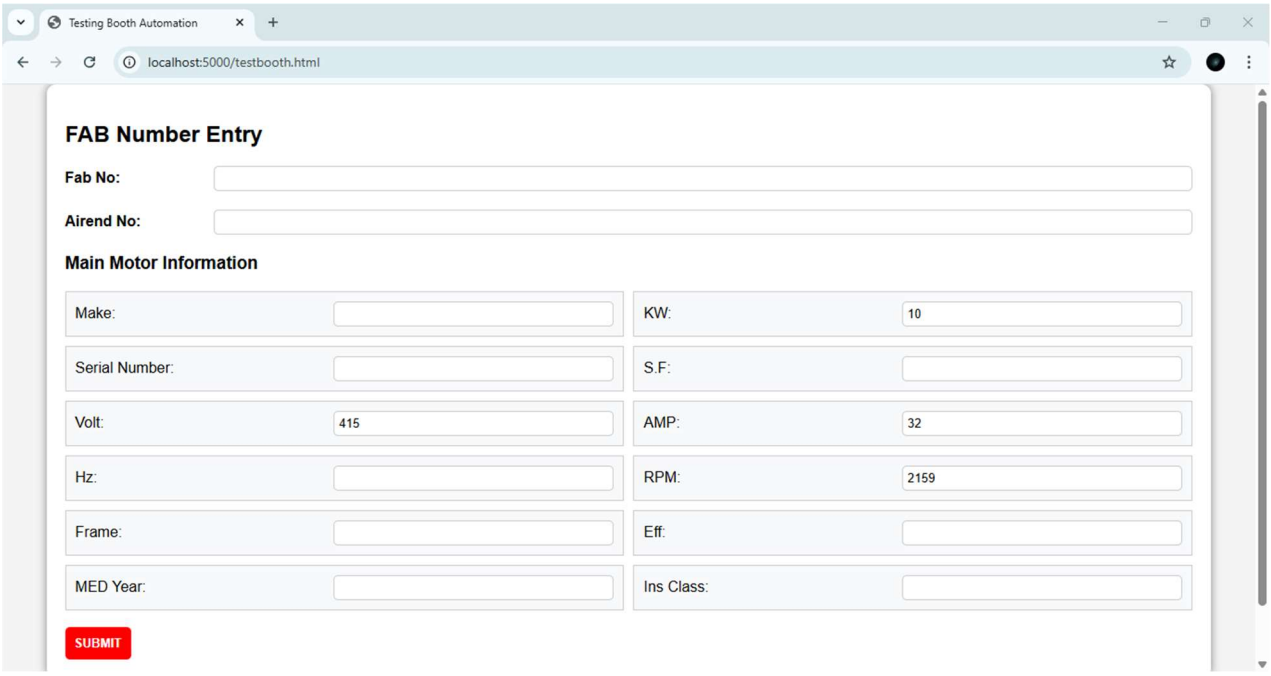
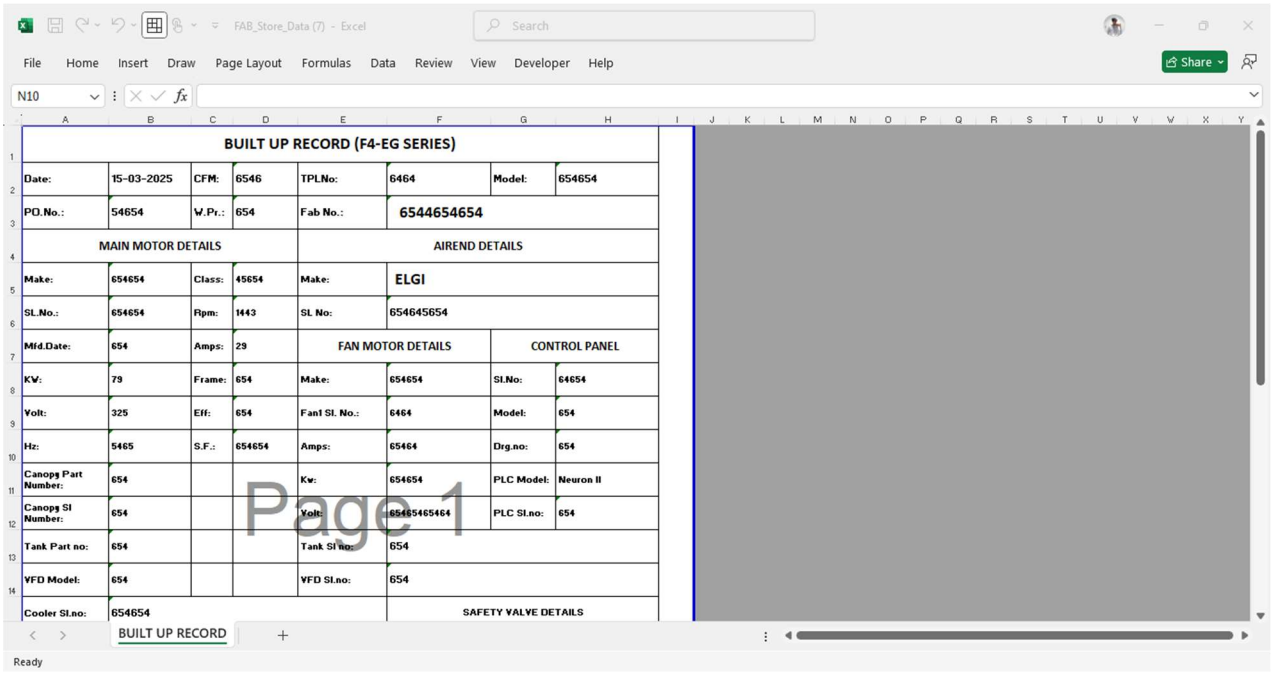


Fig 11:Testbooth Page

The Test Booth Module is responsible for capturing and managing performance test data of various components, such as Airend, Motors, and Tanks. It enables real-time quality control and historical test data analysis, ensuring that all tested components meet the required standards before final assembly.

4.4 OUTPUT DESIGN:

Excel Report:



BUILT UP RECORD (F4-EG SERIES)							
Date:	15-03-2025	CFM:	6546	TPLNo:	6464	Model:	654654
PO.No.:	54654	W.Pr.:	654	Fab No.:	6544654654		
MAIN MOTOR DETAILS				AIREND DETAILS			
Make:	654654	Class:	45654	Make:	ELGI		
SL.No.:	654654	Rpm:	1443	SL No:	654645654		
Mfd.Date:	654	Amps:	29	FAN MOTOR DETAILS		CONTROL PANEL	
KW:	79	Frame:	654	Make:	654654	Sl.No:	64654
Volt:	325	Eff:	654	Fanl Sl. No.:	6464	Model:	654
Hz:	5465	S.F.:	654654	Amps:	65464	Drg.no:	654
Canopy Part Number:	654			Kw:	654654	PLC Model:	Neuron II
Canopy Sl Number:	654			Volt:	65465465464	PLC Sl.no:	654
Tank Part no:	654			Tank Sl no:	654		
VFD Model:	654			VFD Sl.no:	654		
Cooler Sl.no:	654654			SAFETY VALVE DETAILS			

Fig 12: Excel Report

The Excel Extraction Module automates the process of retrieving and exporting manufacturing data into structured Excel reports.

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 SYSTEM TESTING:

FUNCTIONAL TESTING

Functional testing ensures that the Industrial Data Logging and Reporting System performs accurately across all its modules, including Stores, Test Booth, and Assembly. The process starts with verifying user authentication, ensuring only authorized personnel can log in and access different functionalities. Each module undergoes validation, where data entry forms are tested to ensure correct values are stored in the database. The Stores Module is tested for accurate inventory tracking, the Test Booth Module for correct recording of test parameters, and the Assembly Module for capturing assembly-related specifications. Additionally, the Excel export functions are tested to ensure they generate properly formatted reports with accurate real-time data. This testing phase guarantees that all features function as expected without errors.

INTEGRATION TESTING:

Integration testing ensures seamless interaction between different modules and system components. The system involves database connectivity, data retrieval, processing, and export functionalities, which must work together without issues. Testing begins by validating data flow from form submission to the MySQL database, ensuring accurate storage and retrieval. The system then checks data transfer across different modules, verifying that information entered in the Stores Module is accessible to the Test Booth and Assembly Modules as required. The automated Excel generation features are tested to ensure extracted data is correctly formatted and reflects real-time updates. Any inconsistencies in data processing, report generation, or UI interactions are identified and resolved to ensure smooth system operation.

SECURITY TESTING

Security testing is essential to protect sensitive manufacturing data from unauthorized access and potential cyber threats. The login module undergoes authentication testing to prevent unauthorized access and brute-force attacks. Role-based access control is tested to ensure only authorized users can modify or retrieve specific data. Database security testing ensures that sensitive details such as inventory records, test results, and assembly specifications are protected from SQL injection or unauthorized tampering. Additionally, data privacy testing ensures that exported reports do not expose confidential information, preventing leaks in Excel or PDF documents. These measures help maintain the integrity and confidentiality of manufacturing data.

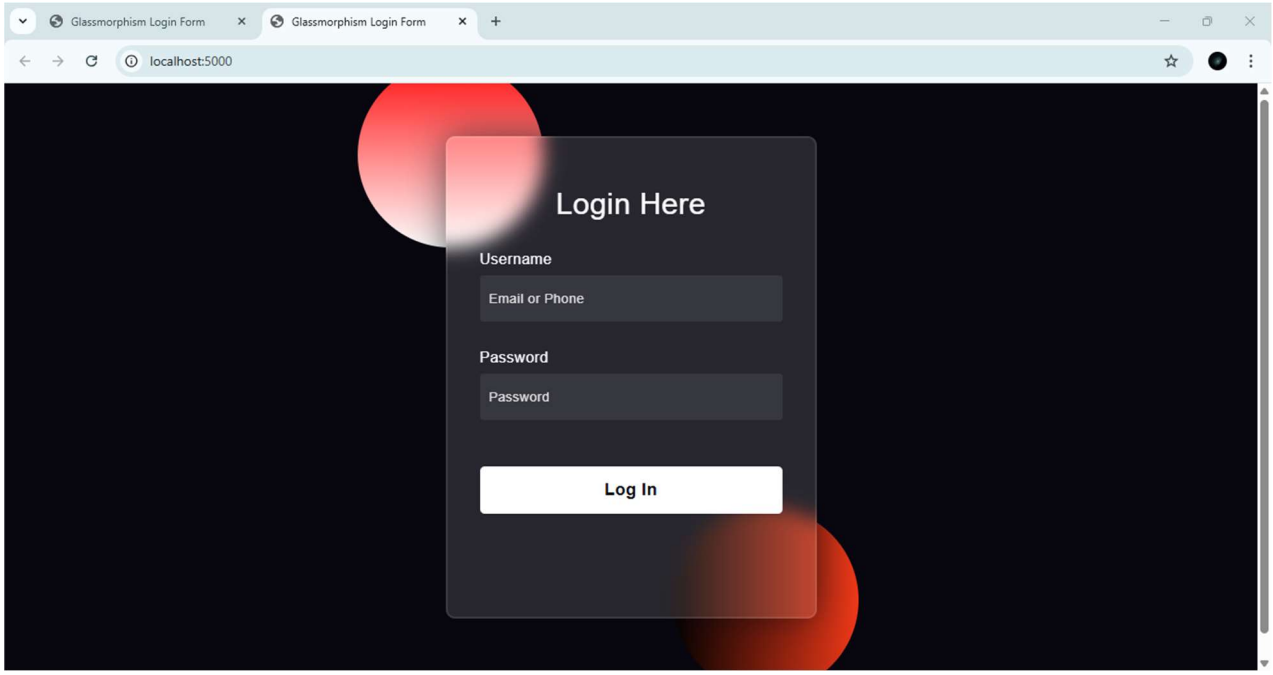


Fig 13: Excel Report

PERFORMANCE TESTING

Performance testing ensures that the Industrial Factory Data Logging and Reporting System operates efficiently under various conditions, including high data loads and concurrent user interactions. Load testing evaluates the system's capability to handle multiple users entering data into the Stores, Test Booth, and Assembly Modules simultaneously, while also assessing database performance during large queries and extensive Excel/PDF exports. Speed testing measures the system's response time for database queries, form submissions, and report generation, ensuring real-time updates and quick processing. Scalability testing determines whether the system can efficiently manage increasing data volumes without performance degradation, with potential future enhancements like cloud storage for improved capacity. Stress testing pushes the system to its limits by simulating heavy concurrent data entry, large-scale data retrieval, and multiple report generations at once to identify bottlenecks and optimize performance. By conducting these tests, the system is fine-tuned for speed, reliability, and scalability, ensuring efficient and seamless data management in a manufacturing environment.

6. CONCLUSION

The Industrial Factory Data Logging and Reporting System enhances efficiency, accuracy, and productivity in manufacturing by integrating automated data entry, real-time database storage, and structured report generation. By replacing traditional manual logging methods, this system reduces human errors, improves traceability, and streamlines decision-making processes. With key modules such as Stores Management, Assembly Tracking, and Test Booth Monitoring, the system ensures that inventory, production, and quality control data are recorded and managed effectively. The use of Node.js, Express.js, MySQL, and ExcelJS enables seamless backend operations, while a user-friendly frontend built with HTML, CSS, and JavaScript allows intuitive interaction for factory personnel. Additionally, the Login Module ensures secure access through authentication mechanisms. By integrating automated Excel and PDF report generation, this system significantly improves data management, making it easier for manufacturing industries to analyze and optimize their operations. Future enhancements may include IoT integration, AI-driven predictive maintenance, and cloud-based storage, ensuring further scalability and automation. This project lays the foundation for a fully digital and connected smart factory environment, revolutionizing traditional manufacturing processes.

SCOPE OF FUTURE ENHANCEMENT

The Industrial Data Logging and Reporting System has been developed to streamline manufacturing processes by integrating real-time data collection, database management, and automated reporting. While the system effectively enhances operational efficiency, several future enhancements can further improve its capabilities, scalability, and intelligence. One of the major enhancements includes IoT integration, which will enable real-time sensor data collection from machinery, allowing for predictive maintenance and reducing downtime. Additionally, the incorporation of AI-based predictive analytics can help analyze historical data for fault prediction, performance optimization, and automated decision-making in production planning and resource allocation.

To ensure scalability and remote accessibility, cloud-based data storage can be implemented using platforms like AWS, Azure, or Google Cloud, allowing factory operations to be monitored remotely. The development of a mobile application will further enhance usability, enabling managers to access real-time data and receive alerts from anywhere. Strengthening security measures with role-based access control (RBAC), multi-factor authentication (MFA), and data encryption will protect sensitive manufacturing data from cyber threats.

For better data analysis, advanced report customization can be introduced, allowing users to generate detailed analytics and visual insights by integrating the system with tools like Power BI, Tableau, or Google Data Studio. Furthermore, blockchain technology can be leveraged to ensure data integrity, enhancing transparency, security, and traceability within manufacturing processes.

PUBLICATIONS

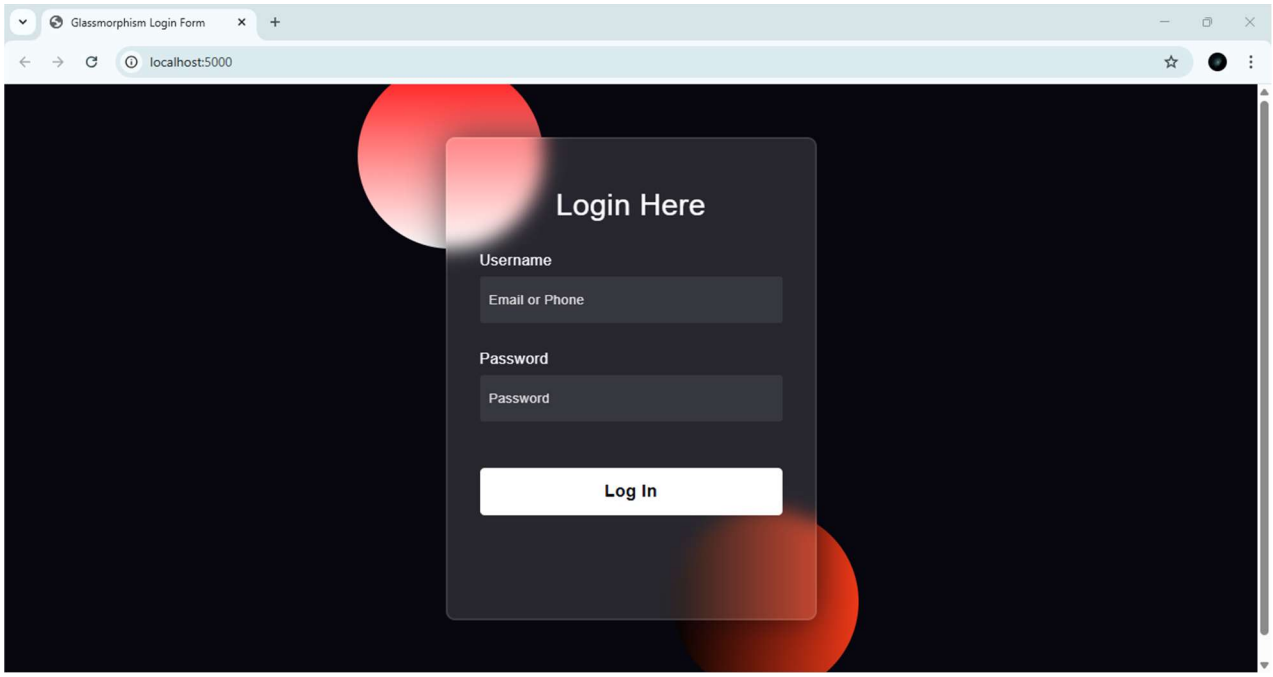
Gowri, K., Aswath, A., Adarsh, AP, (2025). Mastering the Cloud. Amazon Kindel, Retrieved from https://www.amazon.com/dp/B0F1FPBMWF

BIBILIOGRAPHY

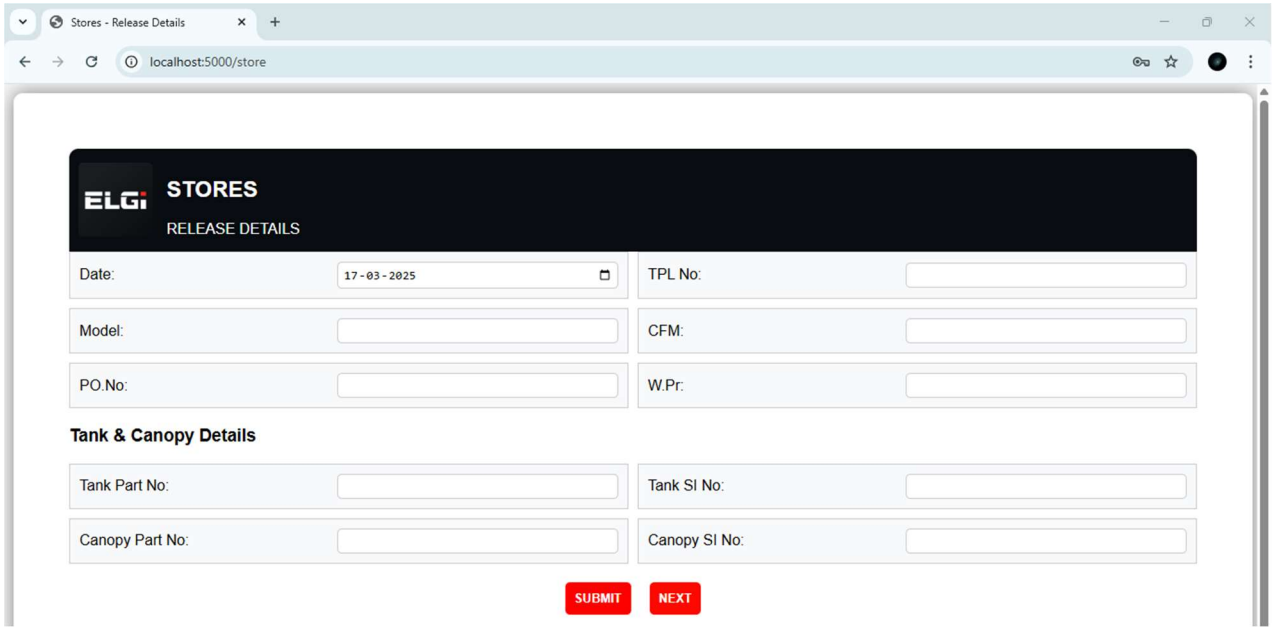
- https://www.opentext.com/what-is/digital-factory
- https://www.augmentir.com/glossary/what-are-digital-sops
- https://4industry.com/blog/everything-about-sop/
- https://www.indeed.com/career-advice/career-development/how-to-make-reports-in-excel
- https://www.jaspersoft.com/articles/what-is-production-reporting
- https://www.elgi.com/in/
- https://pluto-men.com/what-are-digital-sops-and-how-create-them-industrial-training/

APPENDICS

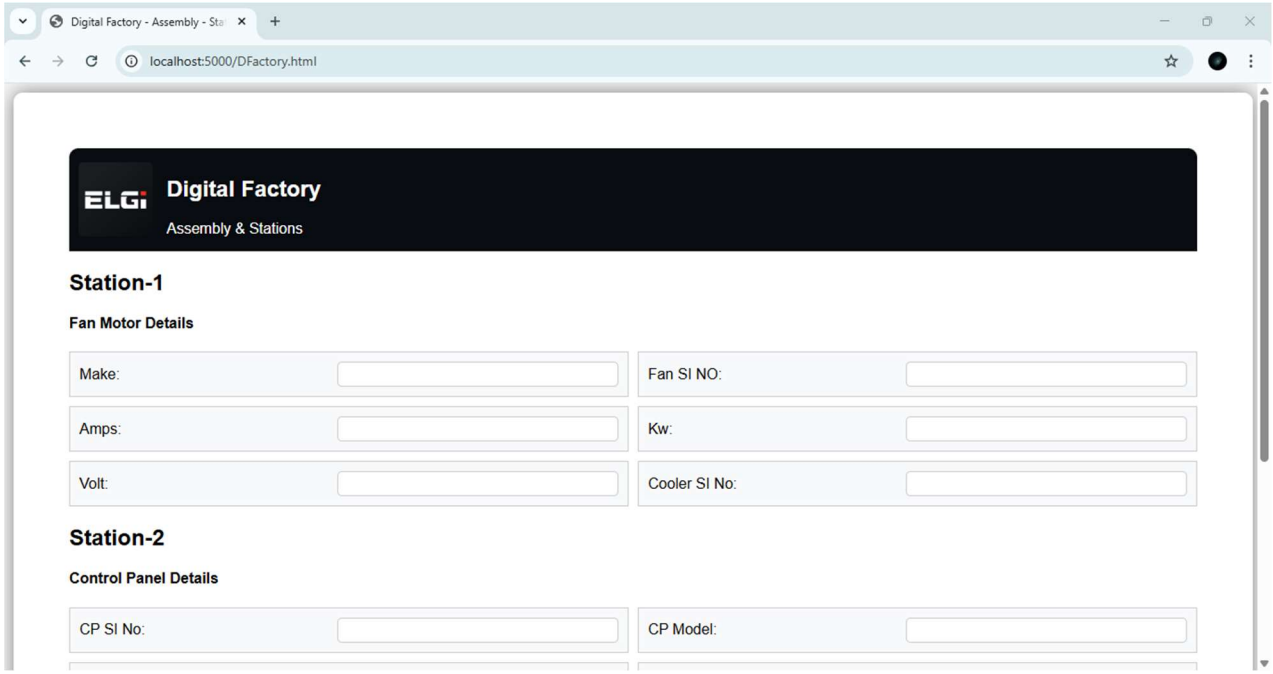
SCREENSHOTS



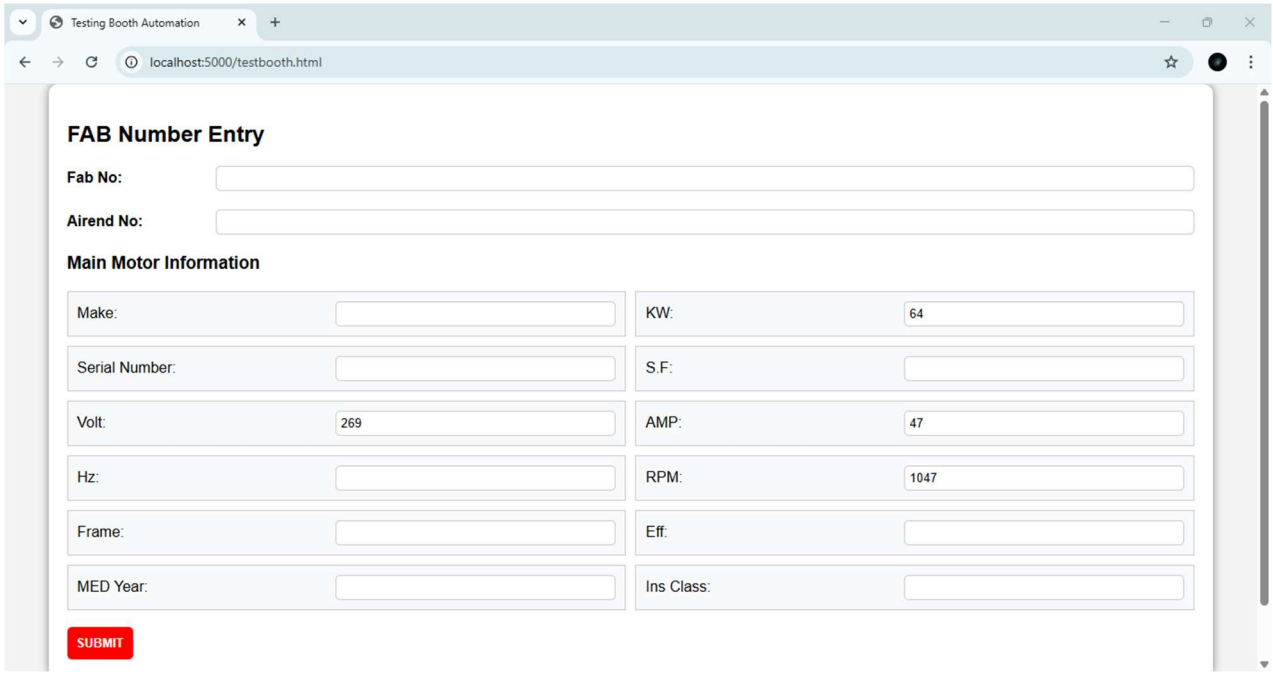
Login Page



Store page



Assembly & Station Page



Test booth Page

SAMPLE CODE

Login page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Glassmorphism Login Form</title>
    <link rel="stylesheet" href="1login.css"> <!-- Link to external CSS file -->
</head>
<body>
    <div class="background">
        <div class="shape"></div>
        <div class="shape"></div>
    </div>
    <form id="loginForm">
        <h3>Login Here</h3>

        <label for="username">Username</label>
        <input type="text" placeholder="Email or Phone" id="username" required>

        <label for="password">Password</label>
        <input type="password" placeholder="Password" id="password" required>

        <button type="submit">Log In</button>

    </form>

    <script src="1loginscript.js"></script> <!-- Link to external JavaScript file -->
</body>
</html>
```

Excel Report:

```
// 📌 Route to export Excel (With Data from `store_details`, `Digital_Factory_P1`, `Digital_Factory_P2`, & `fab_details`)
app.get("/export/excel", async (req, res) => {
  const storeSql = "SELECT * FROM store_details ORDER BY id DESC LIMIT 1";
  const assemblySql = "SELECT * FROM Digital_Factory_P1 ORDER BY id DESC LIMIT 1";
  const valveSql = "SELECT * FROM Digital_Factory_P2 ORDER BY id DESC LIMIT 1";
  const fabSql = "SELECT * FROM fab_details ORDER BY id DESC LIMIT 1";

  try {
    const [storeResults, assemblyResults, valveResults, fabResults] = await Promise.all([
      new Promise((resolve, reject) => {
        db.query(storeSql, (err, results) => {
          if (err) reject(err);
          else resolve(results);
        });
      }),
      new Promise((resolve, reject) => {
        db.query(assemblySql, (err, results) => {
          if (err) reject(err);
          else resolve(results);
        });
      }),
      new Promise((resolve, reject) => {
        db.query(valveSql, (err, results) => {
          if (err) reject(err);
          else resolve(results);
        });
      }),
      new Promise((resolve, reject) => {
```

```
        db.query(fabSql, (err, results) => {
            if (err) reject(err);
            else resolve(results);
        });
    }),
]);

// Debugging: Log the fetched data
console.log("Store Data:", storeResults);
console.log("Assembly Data:", assemblyResults);
console.log("Valve Data:", valveResults);
console.log("Fab Data:", fabResults);

// Check if all data is available
if (storeResults.length === 0) {
    return res.status(404).json({ message: "Store data not found" });
}
if (assemblyResults.length === 0) {
    return res.status(404).json({ message: "Assembly data not found" });
}
if (valveResults.length === 0) {
    return res.status(404).json({ message: "Valve data not found" });
}
if (fabResults.length === 0) {
    return res.status(404).json({ message: "Fab data not found" });
}

const storeEntry = storeResults[0];
const assemblyEntry = assemblyResults[0];
const valveEntry = valveResults[0];
const fabEntry = fabResults[0];
```

```
const templatePath = path.join(__dirname, "Built_Up_Record_Template.xlsx");
const outputExcelPath = path.join(exportsDir, "FAB_Store_Data.xlsx");

const workbook = new ExcelJS.Workbook();
await workbook.xlsx.readFile(templatePath);
const worksheet = workbook.getWorksheet(1);

// ☑ Insert `store_details` Data
worksheet.getCell("B2").value = storeEntry.date || "N/A";
worksheet.getCell("F2").value = storeEntry.tpl_no || "N/A";
worksheet.getCell("H2").value = storeEntry.model || "N/A";
worksheet.getCell("D2").value = storeEntry.cfm || "N/A";
worksheet.getCell("B3").value = storeEntry.po_no || "N/A";
worksheet.getCell("D3").value = storeEntry.wpr || "N/A";
worksheet.getCell("B13").value = storeEntry.tank_part_no || "N/A";
worksheet.getCell("F13").value = storeEntry.tank_si_no || "N/A";
worksheet.getCell("B11").value = storeEntry.canopy_part_no || "N/A";
worksheet.getCell("B12").value = storeEntry.canopy_si_no || "N/A"; // Fix typo: canopy_si_no

// ☑ Insert 'Digital_Factory_P1' Data
worksheet.getCell("F8").value = assemblyEntry.make || "N/A";
worksheet.getCell("F9").value = assemblyEntry.fan_si_no || "N/A";
worksheet.getCell("F10").value = assemblyEntry.amps || "N/A";
worksheet.getCell("F11").value = assemblyEntry.kw || "N/A";
worksheet.getCell("F12").value = assemblyEntry.volt || "N/A";
worksheet.getCell("B15").value = assemblyEntry.cooler_si_no || "N/A";
worksheet.getCell("H8").value = assemblyEntry.cp_si_no || "N/A";
worksheet.getCell("H9").value = assemblyEntry.cp_model || "N/A";
worksheet.getCell("H10").value = assemblyEntry.drg_no || "N/A";
worksheet.getCell("H11").value = assemblyEntry.plc_model || "N/A";
```

```
worksheet.getCell("H12").value = assemblyEntry.plc_si_no || "N/A";
worksheet.getCell("B14").value = assemblyEntry.vfd_model_no || "N/A";
worksheet.getCell("F14").value = assemblyEntry.vfd_sl_no || "N/A";

// ☑ Insert 'Digital_Factory_P2' Data
worksheet.getCell("H16").value = valveEntry.safety_valve_make || "N/A";
worksheet.getCell("H17").value = valveEntry.batch_no || "N/A";
worksheet.getCell("H18").value = valveEntry.safety_valve_range || "N/A";
worksheet.getCell("B17").value = valveEntry.solenoid_valve_make || "N/A";
worksheet.getCell("B18").value = valveEntry.intake_valve_sl_no || "N/A";
worksheet.getCell("E18").value = valveEntry.tv_element || "N/A";
worksheet.getCell("B19").value = valveEntry.aos_batch_no || "N/A";
worksheet.getCell("E19").value = valveEntry.mpv_sl_no || "N/A";
worksheet.getCell("G19").value = valveEntry.oil_filter_batch_no || "N/A";
worksheet.getCell("B20").value = valveEntry.assembly_remarks || "N/A";
worksheet.getCell("B21").value = valveEntry.quality_remarks || "N/A";

// ☑ Insert `fab_details` Data
worksheet.getCell("F3").value = fabEntry.fabNo || "N/A";
worksheet.getCell("B5").value = fabEntry.make || "N/A";
worksheet.getCell("F6").value = fabEntry.airendNo || "N/A";
worksheet.getCell("B6").value = fabEntry.serialNumber || "N/A";
worksheet.getCell("B8").value = fabEntry.kw || "N/A";
worksheet.getCell("B9").value = fabEntry.voltHz || "N/A";
worksheet.getCell("B10").value = fabEntry.hz || "N/A";
worksheet.getCell("D10").value = fabEntry.sf || "N/A";
worksheet.getCell("D7").value = fabEntry.amp || "N/A";
worksheet.getCell("D6").value = fabEntry.rpm || "N/A";
worksheet.getCell("D8").value = fabEntry.frame || "N/A";
worksheet.getCell("D9").value = fabEntry.eff || "N/A";
worksheet.getCell("B7").value = fabEntry.medYear || "N/A";
```

```
        worksheet.getCell("D5").value = fabEntry.insClass || "N/A";

        // ☑ Save Excel File
        await workbook.xlsx.writeFile(outputExcelPath);
        res.download(outputExcelPath, "FAB_Store_Data.xlsx");
    } catch (error) {
        console.error("❌ Error exporting Excel:", error);
        res.status(500).json({ message: "Error exporting Excel file", error: error.message });
    }
});

// Login endpoint
app.post("/login", async (req, res) => {
    const { username, password } = req.body;

    try {
        const sql = "SELECT * FROM users WHERE username = ?";
        db.query(sql, [username], async (err, results) => {
            if (err) {
                console.error("❌ Database error:", err);
                return res.status(500).json({ message: "Database error" });
            }

            if (results.length === 0) {
                return res.status(401).json({ message: "Invalid username or password" });
            }

            const user = results[0];

            const isPasswordValid = await bcrypt.compare(password, user.password);
            if (!isPasswordValid) {
```

```
            return res.status(401).json({ message: "Invalid username or password" });
        }

        // Set session variable
        req.session.user = user;

        // Login successful
        res.json({ message: "Login successful!", redirectUrl: "/store" });
      });
    } catch (error) {
      console.error("❌ Error:", error);
      res.status(500).json({ message: "Internal server error" });
    }
});

// 📌 Start Server
const PORT = 5000;
app.listen(PORT, () => console.log(`🚀 Server running on http://localhost:${PORT}`));
```