Infosys Springboard Virtual Internship 6.0 Completion Report

Team Details

This project has been developed collaboratively by the following team members, focusing on designing and implementing an intelligent inventory management and optimization solution for retail operation

Batch Number2: **SmartStock Inventory Optimization for Retail Stores**

Start date : 10-11-25

Names:

1.Girija Pavani

2.Aman Kumar

3.DurgaPrasad

4.Smiriti

5. Vaishnavi

Internship Duration: 40-days (12 weeks)

1. Project Title

Smart Stock Inventory Optimization For Retail Stores

2. Project Objective

Smart Stock Inventory Optimization for Retail Stores is an intelligent system designed to automate and enhance inventory management in retail environments.

Its core objectives are to:

- Analyze historical sales data to forecast future product demand using data-driven techniques.
- Identify overstock and out-of-stock situations by continuously monitoring inventory levels.
- Optimize stock quantities to reduce inventory holding costs and minimize product wastage.
- Recommend optimal reorder quantities and replenishment schedules for efficient stock control.
- Support retail managers with analytical insights to improve inventory planning and decision-making.

The project improves traditional inventory management practices by replacing manual and estimation-based processes with an automated, accurate, and scalable solution that ensures product availability while reducing operational costs for retail businesses.

3. Project description in detail

The **Smart Stock Inventory Optimization for Retail Stores** system is a data-driven web-based application designed to automate and improve inventory management processes in retail environments.

The system allows retailers to upload historical sales data, after which it:

- Collects and processes historical sales and inventory data.
- Performs data cleaning and preprocessing to handle missing values and inconsistencies.
- Analyzes sales trends, demand patterns, and seasonal variations using statistical and predictive techniques.
- Forecasts future product demand to support proactive inventory planning.
- Monitors stock levels to identify potential overstock and out-of-stock conditions.

The system further:

- Calculates optimal reorder quantities based on forecasted demand and current stock levels.
- Suggests appropriate replenishment timings to maintain continuous product availability.

- Provides actionable insights through analytical summaries and visual representations such as charts and indicators.

The application enhances usability by presenting clear inventory status dashboards and decision-support metrics. This project supports retail managers by enabling faster, data-driven inventory decisions, reducing operational costs, minimizing wastage, and improving overall supply chain efficiency.

4. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Requirement analysis and problem understanding	Retail inventory challenges analyzed, problem definition and objectives finalized
Week 2	Technology stack selection and system architecture design	Python backend structure finalized, environment and configuration setup completed
Week 3	Sales and inventory data collection and handling	Data loading and storage mechanisms implemented and verified
Week 4	Data preprocessing and cleaning	Missing values handled, data standardized for forecasting
Week 5	Demand forecasting model design	Initial forecasting logic implemented and tested
Week 6	Inventory optimization logic development	Reorder points, stock thresholds, and optimization rules implemented
Week 7	Alert system integration and dashboard development	Stock-out and overstock alerts integrated, UI dashboards functional
Week 8	Testing, validation, and documentation	System tested with sample datasets, documentation completed, project prepared for evaluation

5a. Key Milestones

Milestone	Description	Date Achieved

Project Kickoff	Initiated the project by defining objectives, scope, functional requirements, team roles, and success criteria for structured execution.	14-Nov-2025
Prototype / First Draft	Developed the initial working prototype implementing core features such as resume parsing, skill extraction, matching logic, and basic UI flow.	20-Nov-2025
Mid-Term Review	Conducted a progress review to assess implementation status, identify gaps, refine the project approach, and incorporate feedback.	04-Dec-2025
Final Submission	Completed and submitted the fully functional Budgetwise project with all features, testing, and documentation.	20-Dec-2025
Presentation	Prepared and delivered the final presentation showcasing system architecture, key functionalities, outcomes, and project impact.	30-Dec-2025

5b. Project execution details.

The **Smart Stock Inventory and Optimization System** was developed using a modular and layered architecture to ensure scalability, maintainability, and ease of integration. The backend was implemented in Python, while the frontend was developed using standard web technologies.

System configuration and setup were carefully managed to define environment settings and dependencies. The application was designed with a clear entry point to manage overall execution and workflow.

Data handling and persistence were implemented to efficiently load, preprocess, and manage sales and inventory data. Database operations, data models, and validation logic were structured to ensure accuracy, consistency, and reliability.

Demand forecasting and inventory optimization logic were developed to analyze historical sales data, predict future demand, and support optimal stock-level decisions. Authentication and security mechanisms were incorporated to ensure secure access and proper authorization.

Alert and notification mechanisms were implemented to automatically notify users of overstock and stock-out situations. Real-time communication channels were integrated to ensure timely alerts.

The frontend interface was designed to be responsive and user-friendly, providing dashboards to monitor inventory status, alerts, and forecasts, along with clear data visualizations to support decision-making.

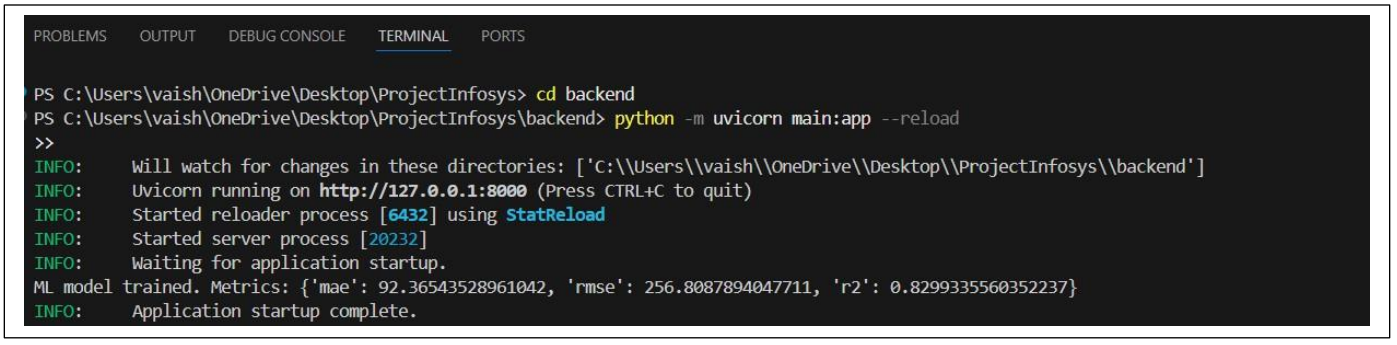
Overall, the system was iteratively tested, refined, and validated to ensure accuracy, reliability, and usability, resulting in a robust solution capable of supporting intelligent retail inventory management

6. Snapshots / Screenshots

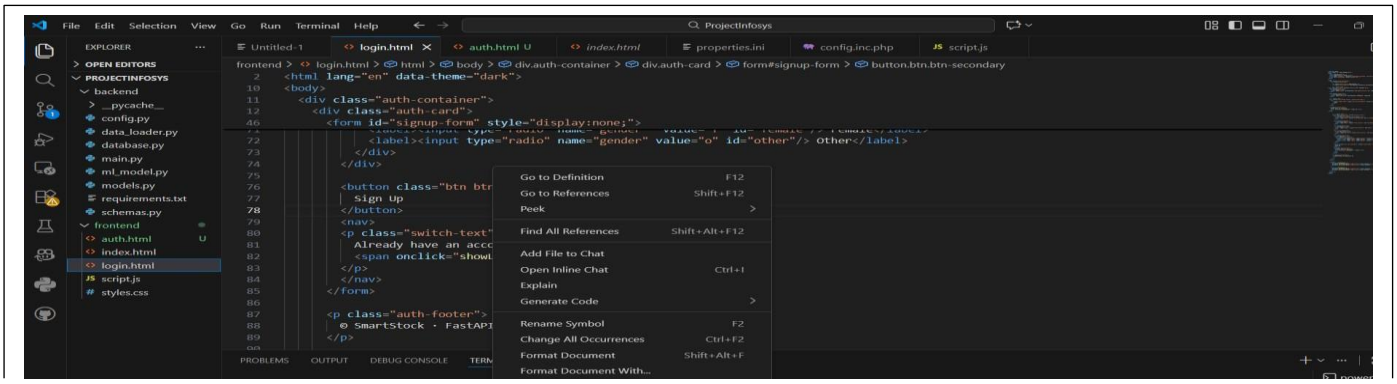
Navigate to GitHub and clone the project repository to your local system. Open the cloned repository in Visual Studio Code to begin the execution process.

The GitHub repository link is: https://github.com/Girijarajarapu/smart-stock-inventory-optimization-for-retail-stores-

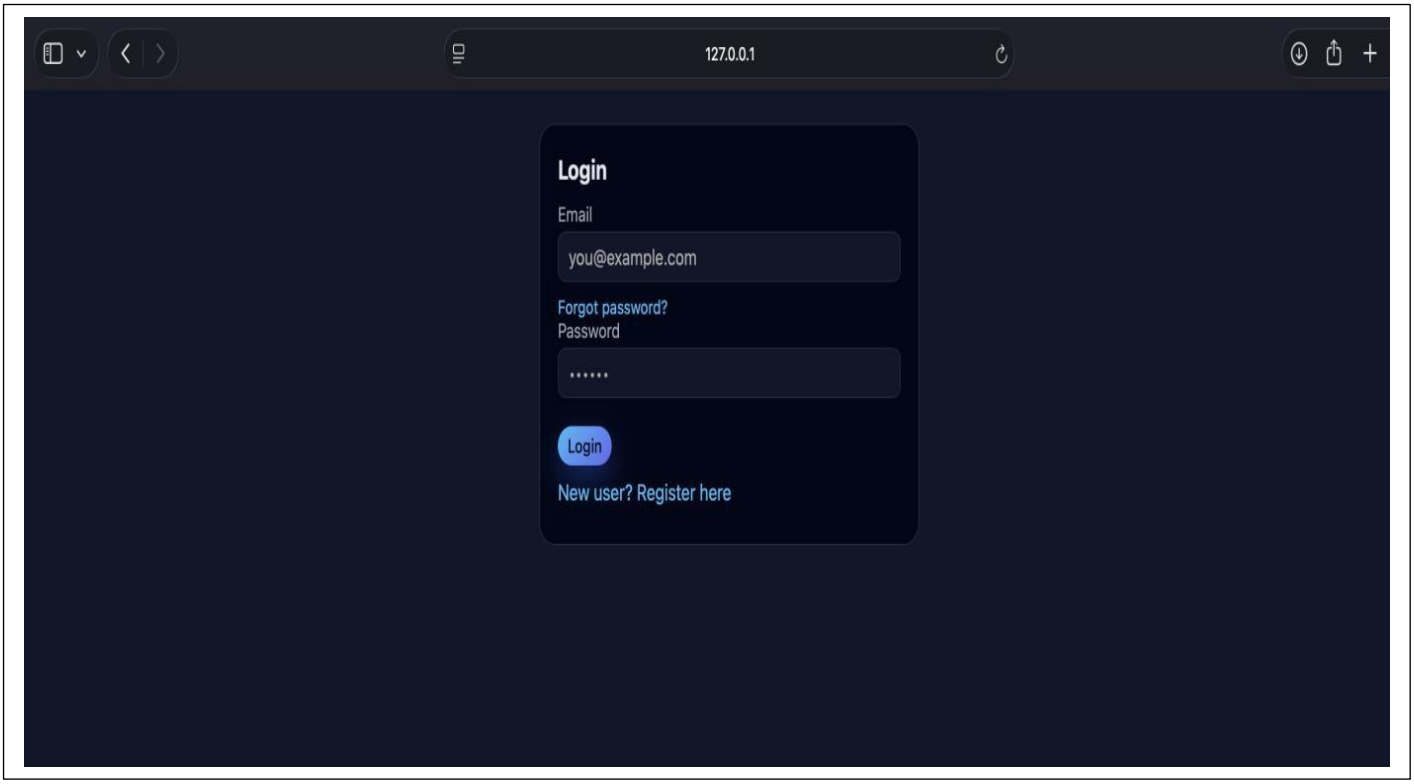
Start the with cd backend & python -m uvicorn main:app --reload



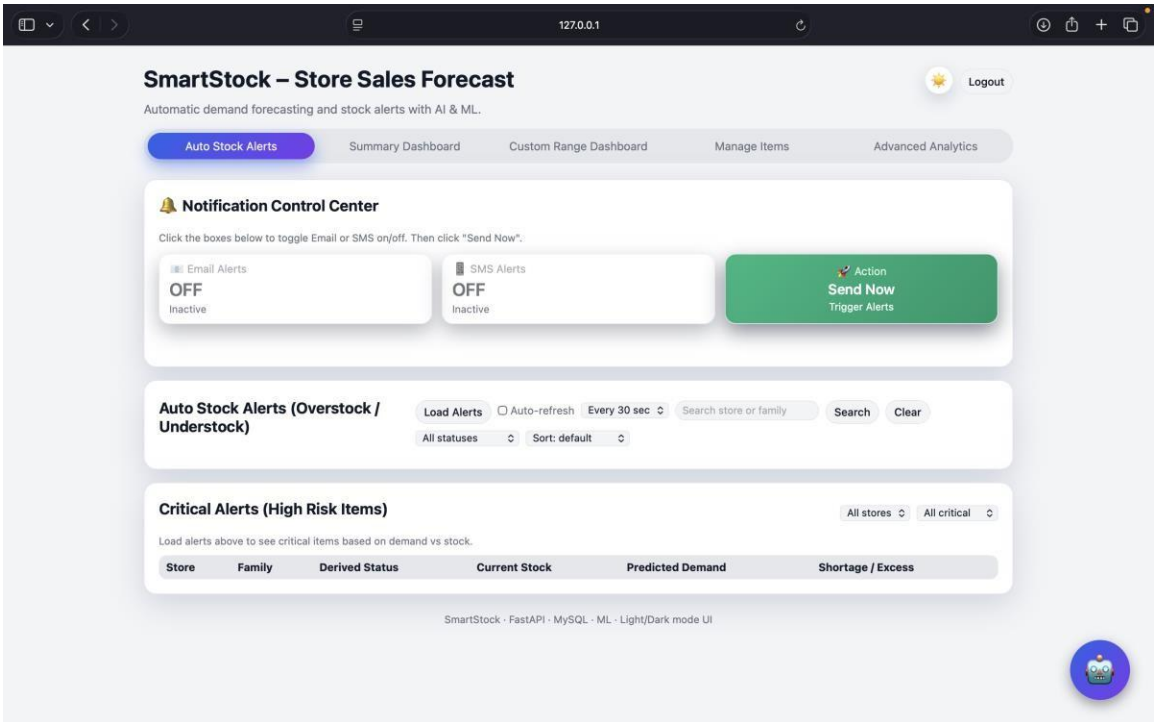
After cloning the repository and opening the project in Visual Studio Code, navigate to the frontend folder and locate the **login.html** file. Right-click on the file and select **"Open with Live Server"**. This will launch the login page in the web browser, allowing users to access the application interface. Running the file through Live Server ensures proper loading of HTML, CSS, and JavaScript files and enables a smooth and interactive user experience during execution.



After clicking on Login/Signup here user need to register and log in for Authentication.

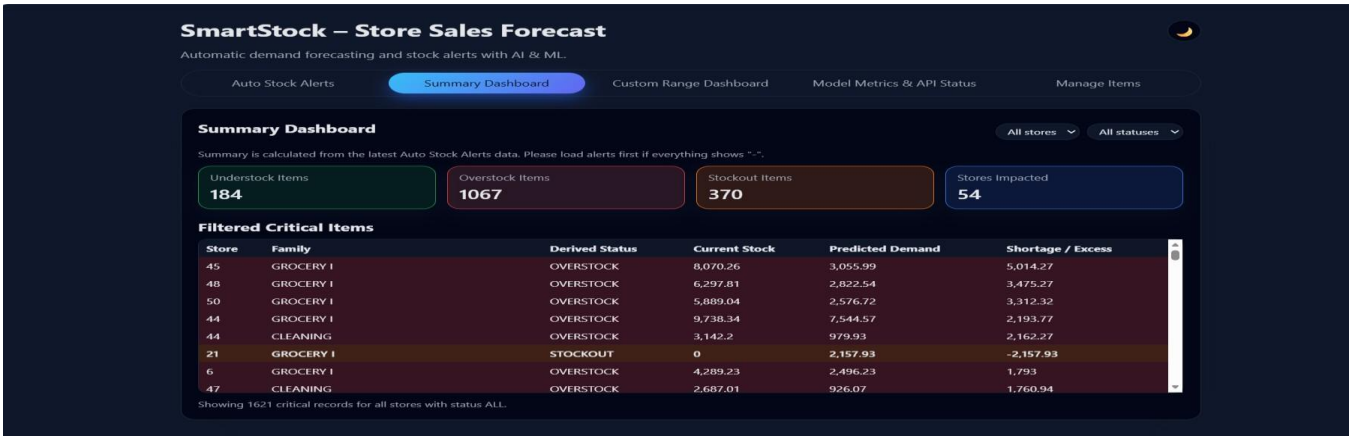


After successfully opening the login page and completing authentication, the system redirects the user to the **Auto Stock Alerts** section. This screen displays real-time alerts related to inventory conditions, including stock-out warnings and overstock notifications. The alerts are generated based on demand forecasts and predefined inventory thresholds, helping users quickly identify critical stock situations. This feature enables timely decision-making by highlighting items that require immediate replenishment or optimization.

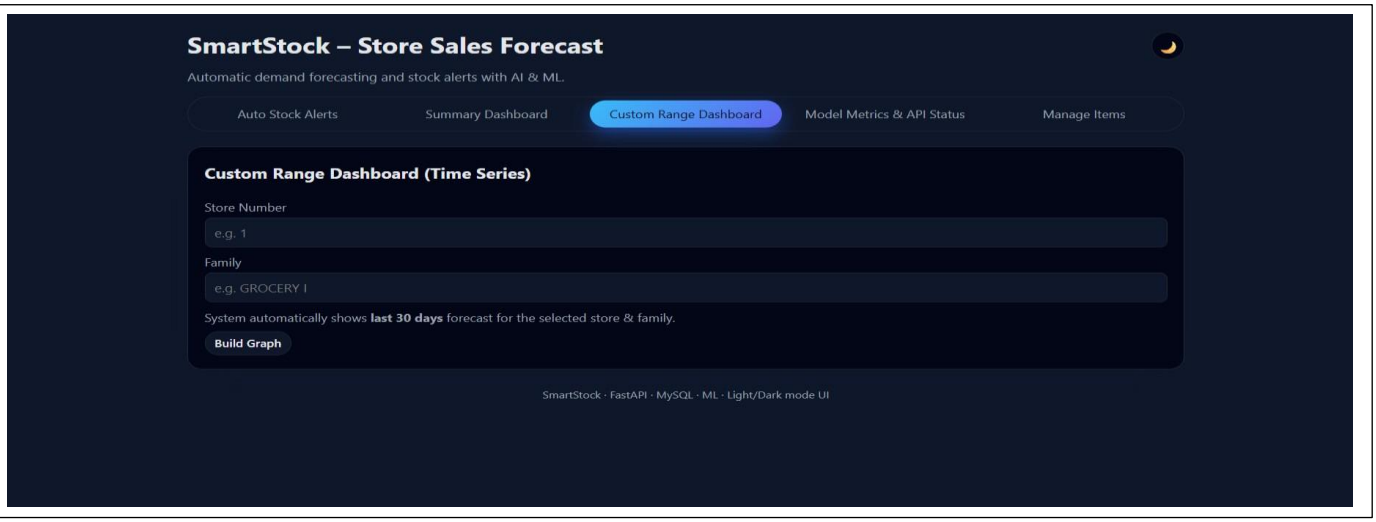


When the user wants to view alerts, they can click the **Load Alerts** button, which fetches and displays the latest inventory alert information.

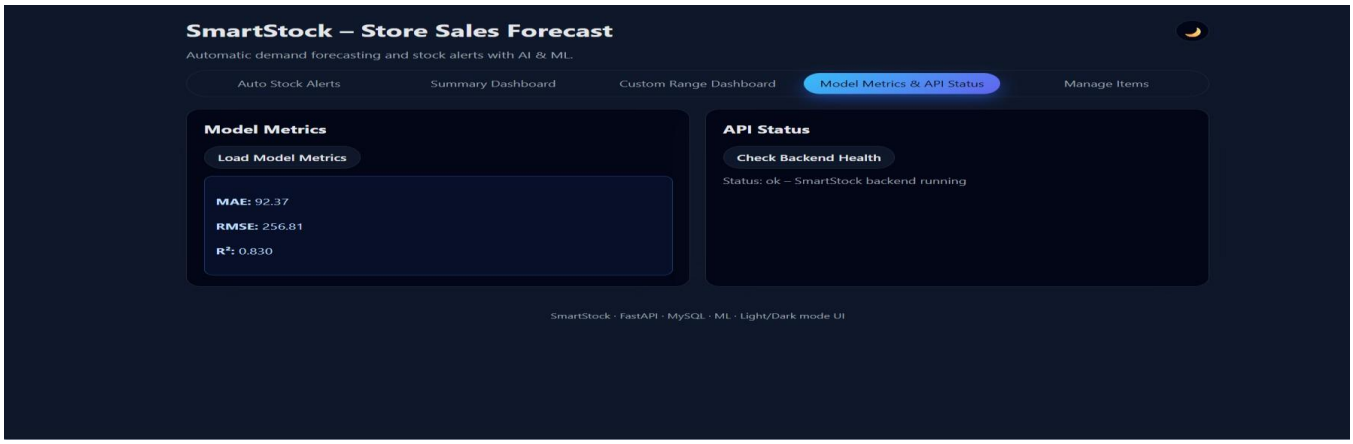
After this, the system navigates to the **Summary Dashboard**, which presents an overall snapshot of inventory status and key performance insights..



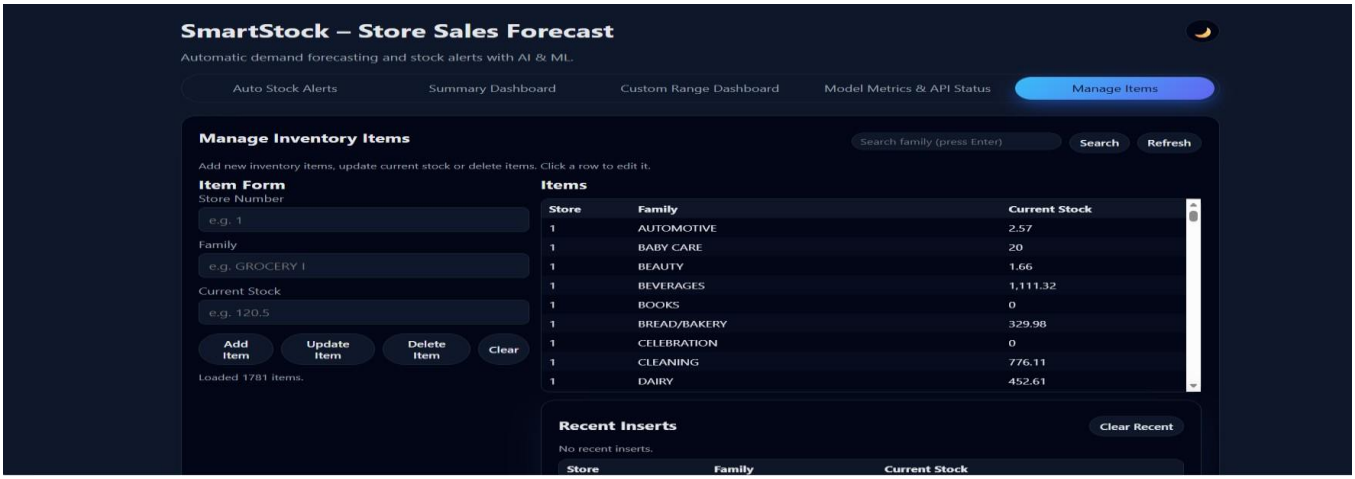
Next, the application moves to the **Custom Range Dashboard**, where users can analyze inventory and sales data for a selected time period.



After that, the system navigates to the **Model Metrics & API Status** page, which displays machine learning performance metrics and confirms the operational status of backend APIs.



Next, the application opens the **Manage Items** page, allowing users to add, update, and manage inventory item details efficiently.



7. Challenges Faced

During the development of the Smart Stock Inventory and Optimization System, several technical and operational challenges were encountered:

Data Quality and Inconsistency: Retail sales and inventory data were obtained in varying formats and levels of completeness. Missing values, inconsistent product naming, and irregular sales entries affected forecasting accuracy, requiring additional data preprocessing and validation.

Demand Forecasting Accuracy : Initial forecasting results did not accurately capture real-world demand patterns due to seasonal variations, sudden demand fluctuations, and limited historical data. Multiple iterations were required to fine-tune model parameters and improve prediction reliability.

Inventory Optimization Threshold Calibration: Defining optimal reorder points and stock thresholds was challenging. Early configurations resulted in frequent or delayed alerts, necessitating refinement of optimization logic to balance sensitivity and accuracy.

Alert Trigger Reliability: Ensuring timely and accurate alerts for overstock and stock-out conditions required careful handling. Variations in sales velocity occasionally caused false alerts, requiring adjustments to alert rules and validation mechanism.

Backend Module Integration: Integrating multiple backend components such as data loading, machine learning models, database operations, authentication, and alert services required careful coordination. Minor inconsistencies in data flow led to debugging and refactoring efforts.

Frontend Responsiveness and Visualization: Dynamic dashboards, stock indicators, and alert notifications sometimes misaligned across different screen sizes. CSS restructuring and repeated UI testing were required to maintain responsive design and a consistent user experience.

Performance and Scalability Constraints: Processing large datasets or extended sales histories introduced delays in forecasting and optimization tasks. Performance improvements were required through efficient data handling and code optimization.

Dependency and Environment Compatibility: Library version mismatches and dependency conflicts caused execution issues during development. These were resolved through careful dependency management and environment configuration.

Testing and Validation Effort: Validating system behavior across multiple product categories and sales patterns required extensive testing. Ensuring consistent optimization outcomes across diverse datasets was time-consuming and required repeated evaluation cycles

8. Learnings & Skills Acquired

Backend Development : Gained hands-on experience in developing a modular Python-based backend, managing configurations, handling data flow, and integrating multiple system components efficiently.

Data Preprocessing and Analysis: Learned to clean, preprocess, and analyze sales and inventory data, handle missing values, and standardize datasets to improve forecasting accuracy.

Machine Learning and Forecasting : Acquired practical knowledge in implementing demand forecasting models, tuning parameters, and evaluating model performance to support data-driven inventory decisions.

Inventory Optimization Techniques: Developed an understanding of reorder point calculation, stock threshold definition, and optimization strategies to reduce overstock and stock-out situations.

Alerting and Automation: Learned to design and implement automated alert mechanisms for inventory conditions, including rule-based triggers and notification workflows.

Frontend Development and UI Design: Gained experience in building responsive user interfaces using HTML, CSS, and JavaScript, focusing on usability, visualization clarity, and cross-device compatibility.

System Integration and Testing: Enhanced skills in integrating backend services with frontend components, performing functional and usability testing, and debugging system-level issues.

Performance Optimization : Learned techniques to improve application performance through efficient data handling, optimized logic, and scalable system design.

Professional and Collaborative Skills: Improved teamwork, problem-solving, time management, documentation, and communication skills while working in an industry-oriented internship environment.

9. Testimonials from team

Team Member 1 :

“This project enhanced my skills in building responsive and user-friendly interfaces. I gained hands-on experience in designing dashboards, integrating visual elements, and ensuring smooth interaction between the frontend and backend systems.”

Team Member 2 :

“I developed a strong understanding of UI layout design, responsiveness, and visualization of inventory data. Working on dashboards and alert displays improved my ability to translate system data into clear and intuitive user experiences.”

Team Member 3 :

“This project strengthened my backend development skills, particularly in handling data processing, system logic implementation, and module integration. It improved my confidence in building and managing scalable backend workflows.”

Team Member 4 :

“I gained practical experience in demand forecasting, model tuning, and analyzing sales trends. The project improved my analytical thinking and helped me understand how machine learning supports real-world inventory optimization.”

Team Member 5:

“This project helped me develop a strong understanding of database design, query optimization, and efficient data management. I gained experience in structuring and handling inventory and sales data to support analytics and system performance.”

10. Conclusion

The **Smart Stock Inventory and Optimization System** successfully demonstrates how data analytics, forecasting models, and automation can enhance retail inventory management. The system efficiently analyzes sales and stock data, predicts future demand, optimizes inventory levels, and generates real-time alerts — delivering operational efficiency and business value to retail stores.

Throughout development, the team strengthened its expertise in end-to-end system design, from data preprocessing and model implementation to backend development, database management, and frontend integration. The project validated that intelligent inventory solutions can significantly reduce stock-outs, minimize overstock, improve decision-making accuracy, and support efficient resource management.

Overall, the system stands as a **practical, scalable, and industry-aligned solution** with potential for further enhancement, such as advanced forecasting models, predictive analytics dashboards, and enhanced UI features — showcasing the team’s innovation mindset, technical skills, and readiness for real-world implementation.

11. Acknowledgements

We express our sincere gratitude to **Infosys Springboard** for providing an excellent learning environment, industry-level curriculum support, and access to resources that enabled the successful development of this project.

We extend our heartfelt thanks to **Shakthi Gopalakrishnan**, our mentor, for her continuous guidance, encouragement, and support throughout the development lifecycle.

Her insights, patience, and motivating approach played a pivotal role in refining our methods, validating our solutions, and enhancing our learning experience. She has been very encouraging and extremely supportive throughout the project.

We also sincerely appreciate the support from our peers, colleagues, and family, whose motivation, collaboration, and valuable suggestions helped us stay committed and successfully deliver the **Smart Stock Inventory and Optimization System**.