Report 1

- By Rishav Kumar(U20220072)

Smart Inventory Management System

Introduction

Efficient inventory management is a critical component of any business, particularly for small enterprises that often struggle with balancing stock levels, tracking sales trends, and minimizing financial losses due to overstocking or stockouts. Traditional inventory management methods rely on manual tracking or basic software solutions, which can be time-consuming, error-prone, and inefficient. Without real-time insights and data-driven decision-making, businesses face challenges such as excessive inventory holding costs, missed sales opportunities, and difficulties in predicting demand fluctuations.

To address these challenges, an **AI-powered Smart Inventory Management System** has been developed, integrating automation, advanced data analytics, and intelligent decision-making capabilities. This system is designed to optimize inventory operations by automating routine tasks, improving forecasting accuracy, and providing actionable insights based on historical data and real-time stock levels.

Project Overview

The Smart Inventory Management System is structured into four distinct phases, each focusing on progressively enhancing the system's capabilities:

1. **Static Website Development:** The initial phase involved creating a structured, mobile-responsive website that serves as the system's user interface. This phase included designing navigation menus, product catalogue pages, inventory display sections, and standardized input forms for product data.
2. **Database Integration:** The second phase focused on establishing a robust data management system to store and track inventory details, transaction history, supplier information, and stock movements. Role-based user access controls and automated backup mechanisms were also implemented to enhance data security and reliability.
3. **Large Language Model (LLM) Integration:** In the third phase, AI-driven natural language processing (NLP) capabilities were added, enabling the system to process complex queries, generate detailed inventory reports, and assist users with intelligent recommendations.
4. **Retrieval-Augmented Generation (RAG) System:** The final phase incorporated advanced AI features such as pattern recognition, demand forecasting, and decision support, ensuring a proactive approach to inventory management. This system analyses past sales data, supplier performance, and seasonal trends to provide predictive insights and inventory optimization strategies.

By combining AI with inventory management, this system empowers small businesses to operate more efficiently, minimize losses, and enhance their decision-making capabilities. The following sections of this report provide a detailed breakdown of each development phase, outlining the methodologies, implementation processes, and impact of the system.

Week-4(Learnings from others Presentation):

Report 1

- By Rishav Kumar(U20220072)

One of the major takeaways from these presentations was the importance of understanding design terminology when working with LLMs. Effective communication with AI models significantly improves the accuracy of responses, allowing for better UI/UX design suggestions, structured code generation, and enhanced automation in web development.

For example, understanding terms like **"visual hierarchy," "grid systems," and "responsive breakpoints"** ensures that AI-generated designs align with industry best practices. Instead of vague requests like *"make the layout better,"* using precise instructions such as *"adjust the typographic scale to improve readability"* or *"align elements according to an 8pt grid system"* leads to more refined and professional results.

A specific tool that stood out during these discussions was **Vo.ai**, which offers a range of AI-powered features to assist in website development. From these presentations, I learned how Vo.ai can:

- **Generate Design Ideas and Templates:** By providing structured UI suggestions, Vo.ai helps streamline website layouts, navigation structures, and visual aesthetics.
- **Assist in Client-Side Functionality Development:** The tool can generate interactive UI components, including search bars, form validations, and dynamic inventory displays.
- **Write and Execute Code in JavaScript and Python:** This feature is particularly useful for automating repetitive tasks, integrating API calls, and handling user interactions dynamically.
- **Build Visual Diagrams for System Architecture:** It simplifies complex programming concepts by generating flowcharts and diagrams, aiding in better visualization and debugging.

Week-5(Building pages using Api):

Searched Website and Found Interesting Ideas

While exploring various websites, I came across several design elements that stood out, particularly in the **navigation bar and body layout**. To better understand their structure, I inspected the website's code using **browser developer tools**. After identifying the parts I liked, I copied the relevant portion of the code and took a **screenshot of the inspected elements**.

To generate a **comprehensive prompt** for **LLM-assisted UI generation**, I provided the screenshot along with the extracted code to GPT. The goal was to translate the existing structure into a well-defined, reusable **UI component** with a structured prompt.

File Changes and Tagging Approach

To ensure clarity and better implementation, I structured the request by specifying the **exact files that needed modifications**. Using proper **HTML, CSS, and JavaScript segmentation**, I defined where changes should be applied, making it easier to integrate the generated code. The structured approach included:

- **HTML file:** For the overall structure, including navigation and body elements.

Report 1

- By Rishav Kumar(U20220072)

- **CSS file:** To refine styles and ensure visual consistency.
- **JavaScript file:** For interactivity and dynamic behaviour.

What Did Not Work?

Initially, the process faced challenges due to an **unclear goal and improper prompt structuring**. The main issues were:

- **Not specifying the overall goal clearly** – The AI-generated output lacked alignment with the intended design due to vague instructions.
- **Not dividing the request using appropriate tags** – Without separating elements (e.g., `<nav>`, `<section>`, `<header>`), the generated output was inconsistent and lacked modularity.
- **Lack of image description** – The absence of a clear explanation of the screenshot led to incomplete or incorrect AI interpretations.

Prompt Comparison- Bad Prompt:

```
Task: Enhance an existing web page using **HTMX + Eleventy

Goal: Create a modular, scalable structure that allows **f

### **Page Elements**

#### **Navigation Bar (Before Sign-In)**
- Display a **Sign In** button in the navbar.
- When clicked, show a **Sign In Form** with:
  - **Username** field
  - **Password** field
  - **Submit button**
- Ensure the form structure allows future backend integrat

#### **Post Sign-In (Dashboard Page)**
- Upon submitting valid details (static validation for now
- The **navigation bar** should now contain:
  - **Home**
  - **Products**
  - **Sales**
  - **Inventory**
```

Report 1

- By Rishav Kumar(U20220072)

Good Prompt:

```
<Task> Enhance an existing web page using **HTMX + Eleventy (11ty) +

### **<Subtask> Page Elements </Subtask>**

#### **<Subtask> Navigation Bar (Before Sign-In) </Subtask>**
- Add a **Sign In** button in the navbar.
- When clicked, open a **new Sign In page** containing:
  - `<input type="text" placeholder="Username">`
  - `<input type="password" placeholder="Password">`
  - `<button> Submit </button>`
- Ensure the form structure allows **future backend integration**.

#### **<Subtask> Post Sign-In (Dashboard Page) </Subtask>**
- Upon clicking **Submit**, redirect the user to a **new dashboard pag
- The **navigation bar** of this page should contain:
  - `<a href="/"> Home </a>`
  - `<a href="/products"> Products </a>`
  - `<a href="/sales"> Sales </a>`
  - `<a href="/inventory"> Inventory </a>`
```

By refining the approach and **structuring prompts effectively**, I was able to generate a more **accurate and reusable UI layout** with improved AI understanding.

Week-5 (Building Database Schema using API)

To streamline the process of database creation, I first designed the pages that required structured data storage. Once the pages were finalized, I provided all the files to the **LLM API**, ensuring that each file was properly tagged with its respective file name. The instruction to the model was to generate a **Python script** capable of:

- **Creating a database schema** based on the provided page structures.
- **Adding dummy data** to test the integrity of the database.

Challenges and Optimization

Initially, I had to experiment with the structure of the request, deciding which files and data needed to be included for accurate database generation. The process required some trial and error to fine-tune the prompt for better results.

However, **Cursor** significantly improved efficiency by generating the required database script **in a single prompt**, eliminating the need for multiple refinements.

Report 1

- By Rishav Kumar(U20220072)

based on the files i have in the directory create a sqlite schema in 2NF databaase schema so that it can easily be integraded to have all the functionality.

Entities & Relationships

- **Users** – Handles authentication and user roles.
- **Products** – Stores product details, including predicted prices and stock levels.
- **Vendors** – Maintains vendor contact information.
- **Product Vendors** – Many-to-Many relationship between Products and Vendors.
- **Inventory Transactions** – Logs stock inflow and outflow.
- **Sales** – Tracks sales transactions.
- **Sale Items** – Many-to-Many relationship between Sales and Products.
- **Invoices** – Generates invoices for completed sales.
- **Predictions** – AI-based demand and price predictions.