SmartBasket

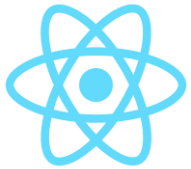
Project Overview & Purpose

- **Overview:** SmartBasket is an AI-driven shopping assistant that predicts users' next likely purchases based on past behavior.
- **Purpose:** To demonstrate how machine learning (ML) can enhance user experience through intelligent item recommendations.
- **Goal:** Build a full-stack application that integrates a neural network model, efficient API design, and a responsive UI.

Objective

- DATA MANAGEMENT: USE A ROBUST DATABASE (POSTGRESQL) FOR STORING USER DATA
- API DESIGN: DEVELOP A REST API USING FASTAPI
- USER INTERFACE: BUILD A MODERN, RESPONSIVE FRONTEND USING REACT
- AI INTEGRATION: DESIGN AND TRAIN A NEURAL NETWORK FOR NEXT-ITEM PREDICTION

System Architecture & Design



FRONTEND:
Pattern
React
ReactQuery
TypeScript

Backend:

FastAPI

Database:

PostgreSQL

Caching:

Redis

ML Service:

Tensorflow
Scikit-Learn
Pandas
Numpy

Design

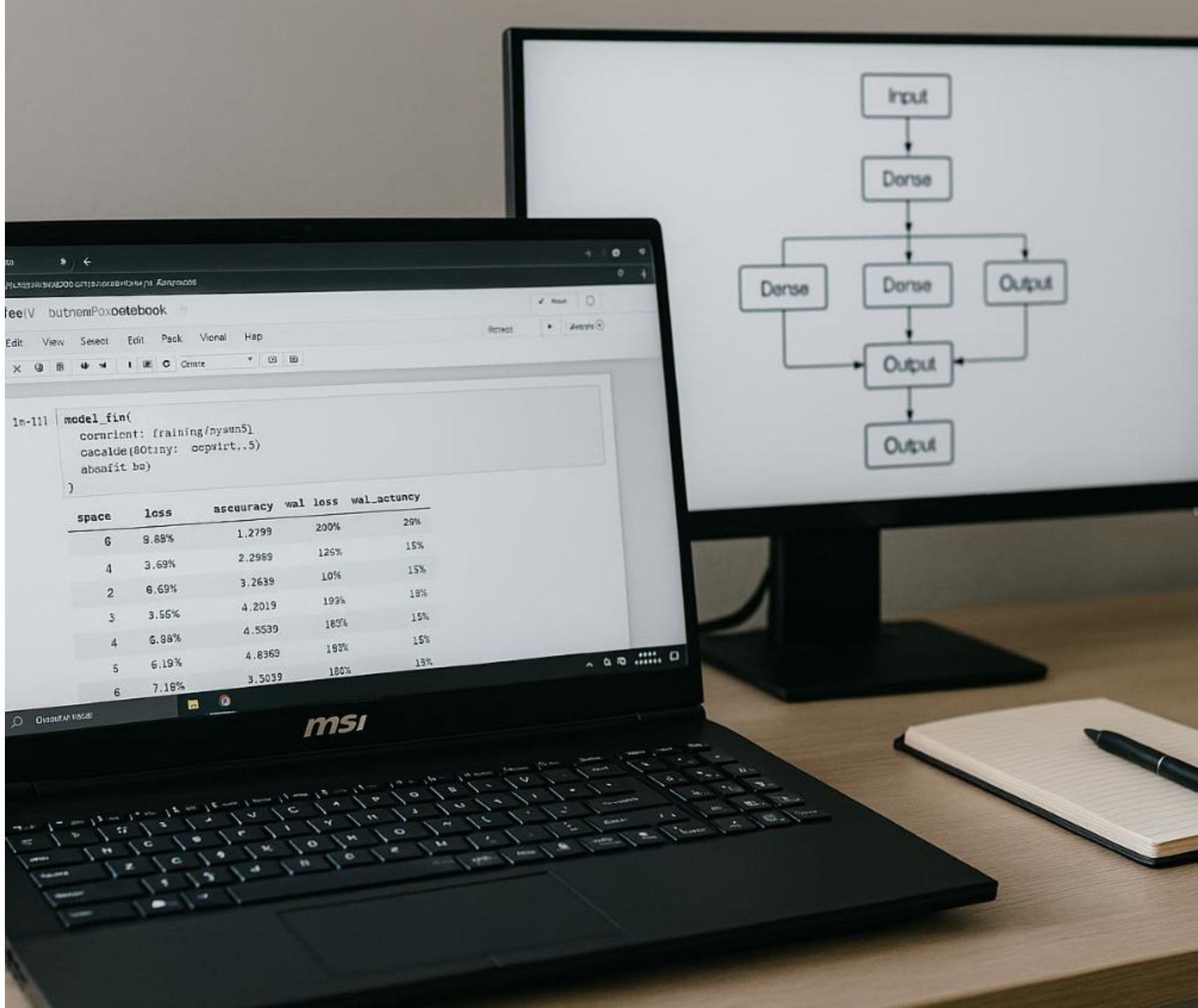
MVC

Data Storage & Management

- PostgreSQL: Robust, Scalable, Structured
- Data Structure: Designed normalized schemas to manage users, shopping sessions, and item logs efficiently, supporting fast queries for predictions.
- Integration: Connected PostgreSQL with the FastAPI backend, enabling smooth data flow between the frontend, backend, and machine learning model.

ML Model

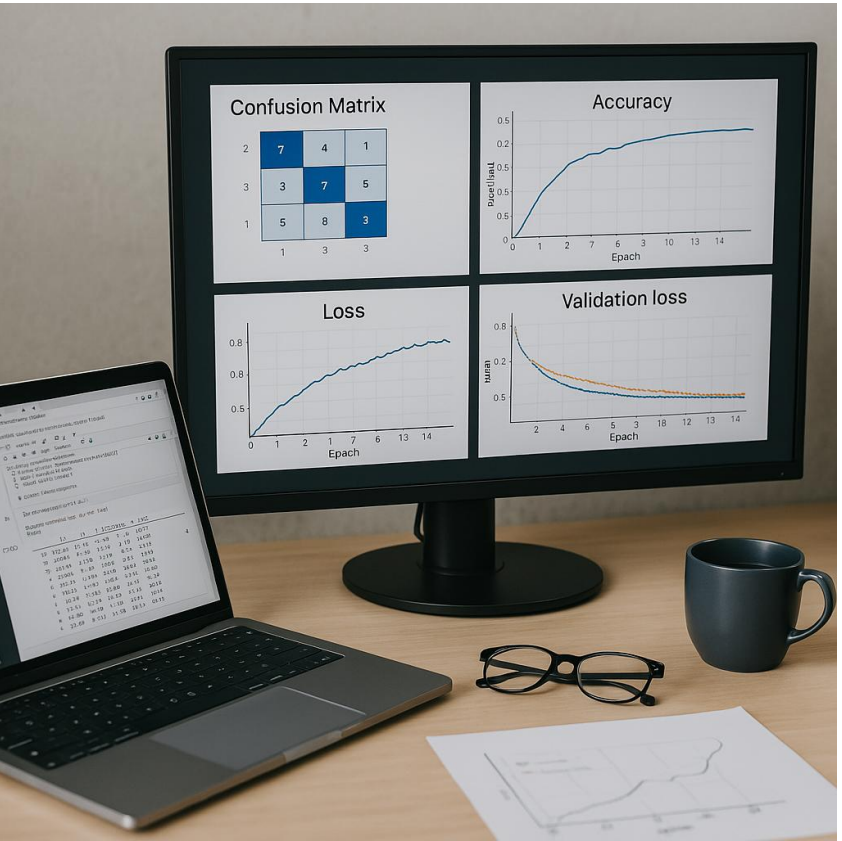
- **Preprocessing:** Cleaned and structured the data into sequential formats, preparing it for model training by handling duplicates and formatting shopping sessions.
- **Model Architecture:** Designed a neural network using TensorFlow to learn item relationships and predict the next likely item.
- **Training:** Trained on user-item interactions with consistent loss reduction across epochs, indicating stable convergence and readiness for deployment.



Evaluation & Performance

MODEL EVALUATION SUMMARY:

- **Final Training Accuracy:** 7.8%
- **Final Validation Accuracy:** 5.5%
- **Final Top-3 Validation Accuracy:** 13.7%
- **Final Training Loss:** 4.17
- **Final Validation Loss:** 4.52



INTERPRETATION:

• **Performance:**
The model shows some early learning but overall prediction accuracy remains low, which is expected given the dataset's size and the complexity of next-item prediction using minimal context (e.g., 1–2 item sequences).

• **Top-3 Accuracy:**
At 13.7%, the model occasionally predicted a relevant item within the top 3 choices. While this is modest, it indicates the model is learning partial associations in shopping patterns.

CHALLENGES IDENTIFIED:

• **Class Imbalance:**
Popular items like "whole milk" dominate the dataset, skewing the model toward repeating common predictions.

• **Sparse Input Representation:**
Predicting the next item based on short item sequences without contextual features (like time, user habits, or categories) limits model performance.

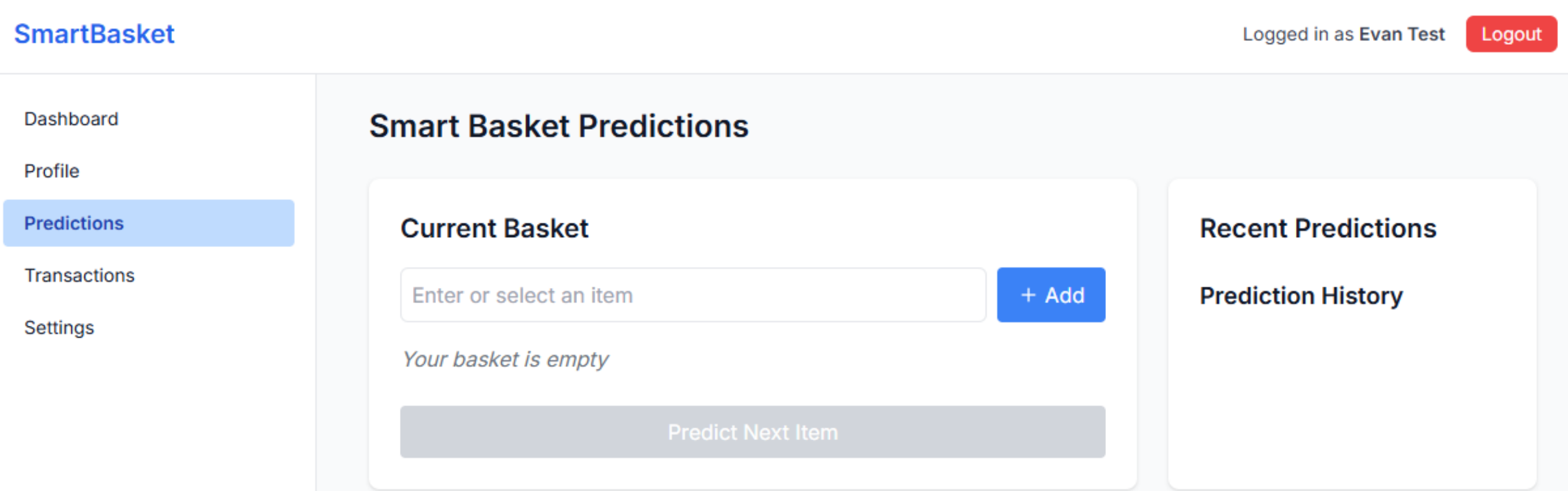
IMPROVEMENT STRATEGIES:

• **Better Data Preprocessing:**
Group similar items (e.g., "whole milk" and "skim milk") to reduce output classes and improve generalization.

• **Model Architecture Tuning:**
Try recurrent or attention-based models (e.g., LSTM, Transformer) for sequential pattern recognition.

• **Input Enrichment:**
Add metadata like time of purchase, day of week, or customer profile info for context-aware predictions.

• **Balanced Sampling or Augmentation:**
Reduce over-represented items and simulate varied transactions for more robust training.



Frontend Implementation & UX

- Dashboard
- Prediction Page
- User Management

Backend & API Integration

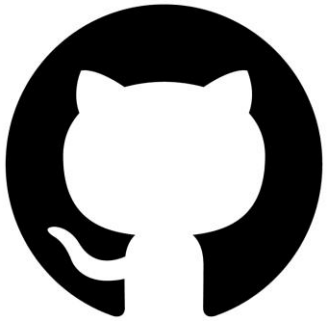
RESTful API

- /token
- /api/v1/users/
- /api/v1/predictions/next-item

Security

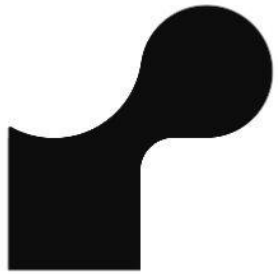
- JWT
- OAuth2

Deployment Strategy & CI/CD

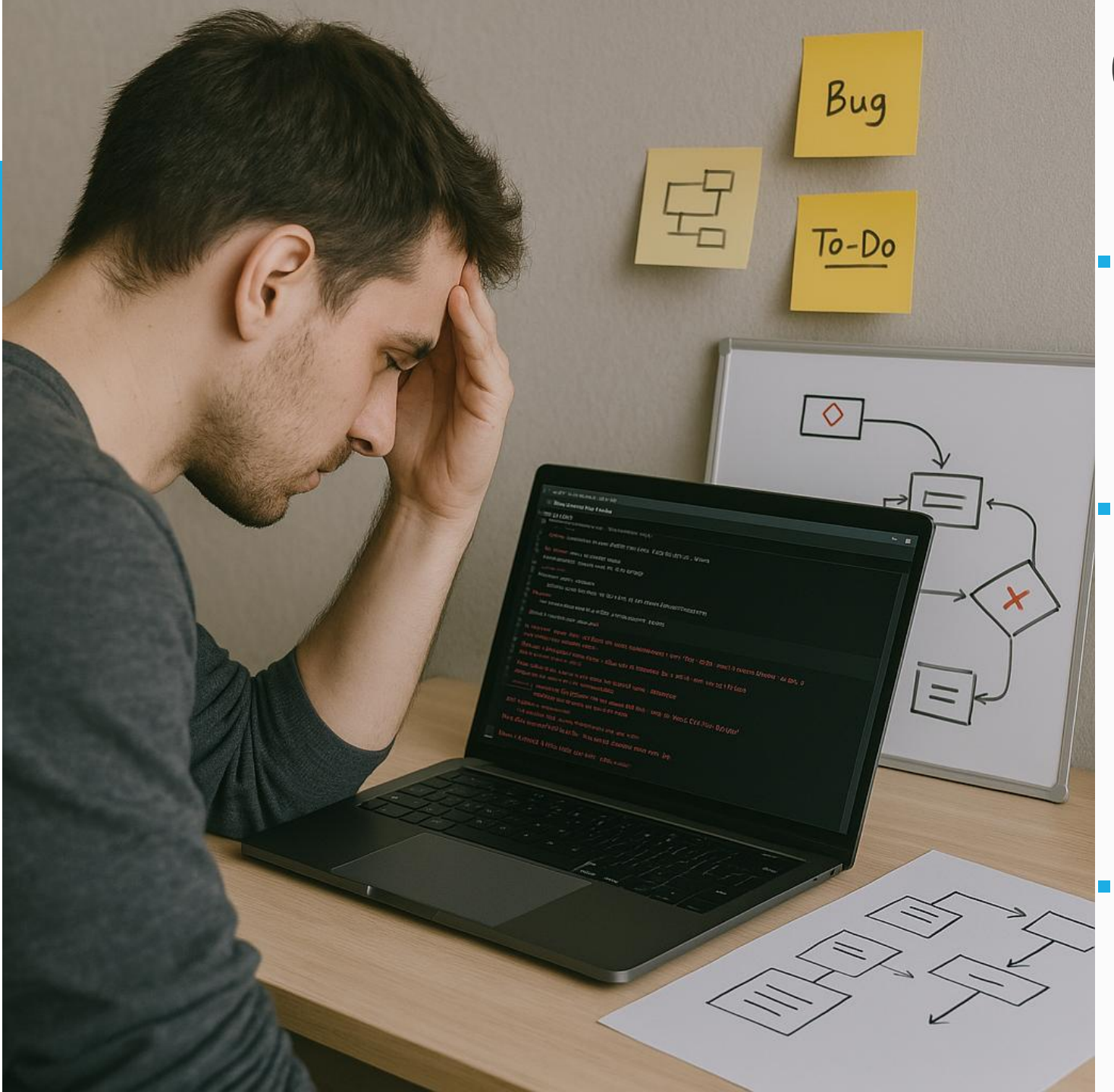


GITHUB
PUSH CHANGES TO GITHUB TO TRIGGER AUTOMATIC DEPLOYMENTS ON RENDER

DOCKER
CONTAINERIZE APPLICATION

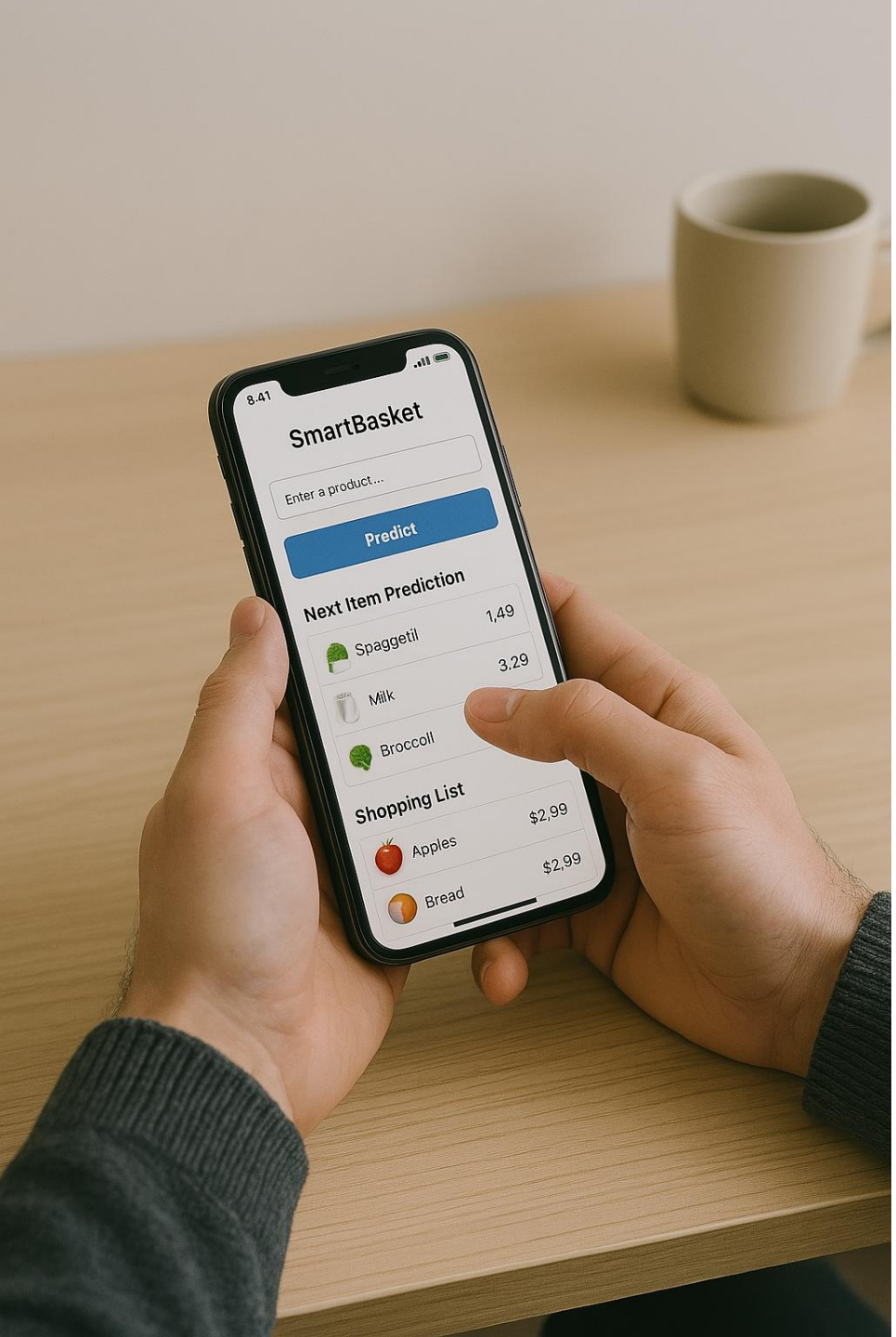


RENDER
WEB SERVICE TO HOST/DEPLOY FULL STACK APPLICATIONS



Challenges & Future Improvement

- **Challenges:**
  Model loading issues and integrating the ML model into the backend.
  Data preprocessing: handling duplicates or improving data source
- **Future Improvements:**
  Enhancing data preprocessing to improve model accuracy
  Expanding the UI with profile editing, transaction analysis, and more prediction options
  Optimizing the neural network (parameter tuning, more sophisticated architecture)
- **Lessons Learned:**
  Importance of robust error-handling and logging
  Challenges of integrating AI components into a full-stack application



Wrap Up

- **Project Summary:** SmartBasket is a full-stack web application designed to predict the next item in a user's shopping list, enhancing the shopping experience through intelligent suggestions. The project integrates a React frontend, FastAPI backend, PostgreSQL database, and a neural network-based machine learning model.
- **Key Achievements:**
  - **Seamless Integration:** Successfully combined multiple technologies to create a cohesive application.
  - **Functional ML Model:** Developed and trained a neural network capable of making next-item predictions.
  - **User-Centric Design:** Focused on creating an intuitive and responsive user interface to facilitate ease of use.
- **Lessons Learned:**
  - **Importance of Data Quality:** Recognized the importance of balanced and high-quality data in training effective machine learning models.
  - **Integration Challenges:** Realized the complexity of integrating machine learning components into a full-stack application.
  - **Continuous Improvement:** Learned that we would need to focus on ongoing refinement and iteration to enhance application performance and user satisfaction.
- **Future Directions:**
  - **Enhanced Personalization:** Incorporate user profiles and preferences to tailor predictions more closely to individual users.
  - **Expanded Features:** Introduce functionalities such as transaction analysis and more diverse prediction options.
  - **Model Optimization:** Continue refining the machine learning model through advanced techniques and expanded datasets.