Progress Report 2

Intelligent Business Analytics System

- for Maximizing Revenue and Efficiency

Student Name: Seungyeol Chae

Student Number: 300362271

Work Logs

Date	Hours Worked	Description of Work
March 1	6	Fixed CORS issues between FastAPI and Flask, ensuring frontend requests to backend services are processed correctly.
March 3	5	Debugged PostgreSQL database persistence issues; verified orders table structure and fixed missing data problem.
March 7	7	Refactored revenue_forecasting.py to ensure file uploads trigger AI forecasting properly; Flask API now integrates correctly with FastAPI.
March 8	6	Fixed frontend integration; ensured AI forecasting results and original analytics (revenue trends, best-selling items) display correctly in Dashboard.js.
March 9	5	Resolved GitHub push errors due to large files; excluded venv/ from repository and successfully committed changes.
March 11	7	Conducted final testing of all endpoints, fixing database insert issues, and validating AI forecast accuracy.

Description of Work Done

This phase focused on **backend stability, frontend integration, AI forecasting integration & improvements, and repository management**. The main challenges included **CORS errors, missing database data, and frontend visualization issues**, all of which were resolved.

1. Backend Fixes & Data Persistence

- **Fixed PostgreSQL table persistence issues**: Ensured uploaded files correctly insert order data into the orders table.
- **Refactored revenue_forecasting.py**: Ensured uploaded files trigger AI forecasting correctly and Flask communicates smoothly with FastAPI.
- **Resolved CORS conflicts**: Allowed frontend access to both FastAPI and Flask APIs by configuring the correct middleware settings.

2. API Debugging & File Upload Handling

- **Fixed analytics API (/analytics/)**: Ensured the API correctly retrieves stored order data from the database.
- **Refactored upload.py to process files correctly**: FastAPI now forwards uploaded files to Flask, which saves them in a structured format for forecasting.
- **Verified data persistence**: Ensured file_processing.py correctly inserts uploaded data into PostgreSQL.

3. Frontend Fixes & Integration

- **Ensured previous analytics visualizations render properly**: Fixed Dashboard.js so that revenue trends, best-selling items, and customer insights display correctly.
- **Fixed API calls in api.js**: Prevented duplicate requests and ensured correct response handling.
- **Improved UI feedback in FileUpload.js**: Added error handling and success messages for file uploads.

4. GitHub & Repository Management

- **Resolved GitHub push issues**: Excluded large venv/ files from the repository to avoid exceeding GitHub's 100MB limit.
- **Successfully committed project updates**: Cleaned up unnecessary tracked files and optimized the repository.

Repository Check-in of Implementation Completed

1. Backend Services

- main.py → FastAPI service, now correctly initializes both analytics and AI forecasting APIs.
- revenue_forecasting.py → Flask service now correctly processes uploaded files and returns AI forecasts.
- upload.py → Fixed to correctly forward uploaded files between FastAPI and Flask.

2. Frontend Components

- Dashboard.js → Fully functional, displaying **both AI forecasts and business analytics data**.
- api.js → API calls structured properly, reducing duplicate requests.
- FileUpload.js → Improved user feedback messages for uploads.

3. Database & File Processing

- file_processing.py → Fixed missing database inserts, ensuring all uploaded data is stored persistently.
- config.py → Revised database connection settings and improved logging.

Next Steps & Improvements

1. Enhancing AI Forecasting Accuracy

- **Fine-tune Prophet forecasting model** to improve revenue prediction accuracy.
- **Compare with LSTM forecasting** to test alternative methods for time-series prediction.

2. Real-Time Data Integration

- **Connect live weather API** to analyze its impact on revenue fluctuations.
- **Enable dynamic updates** so that the system processes new order data at any time.

3. AI-Driven Inventory Forecasting for Small Businesses

- **Develop an AI-based inventory management feature** to help businesses predict demand for specific items.
- **Integrate historical sales data** to **forecast stock levels**, ensuring businesses order optimal inventory amounts.
- **Reduce overstocking & shortages** by aligning inventory decisions with AI-driven revenue predictions.
- **Enhance business efficiency** by providing **data-backed recommendations** on purchasing and restocking cycles.

Conclusion

Since the Midterm Check-in, I have been working to enhance the complexity of the system as suggested by the professor, aiming to develop a more advanced and functional program by integrating AI capabilities. A major focus has been on incorporating AI into revenue forecasting, exploring different approaches to improve prediction accuracy and overall system performance.

Although the AI forecasting functionality is not yet fully stabilized, significant progress has been made, and future efforts will be directed toward refining the AI models and ensuring seamless integration. Additionally, I plan to expand AI-driven features beyond forecasting, such as inventory prediction, to further assist businesses in optimizing their operations.

During this phase, I encountered issues with GitHub repository management, particularly with large file uploads, which resulted in fewer commits than expected. Most of the work has been done offline to address these challenges, and I appreciate your understanding regarding the limited repository update logs. Moving forward, I will ensure that all improvements are properly documented and uploaded as the system development progresses.