Milestone 2: Data Preparation & Initial Modeling/Prototyping

Group 5: Aliyyah Jackhan, Mohammed Aadil and Jonathan Chacko

1. Data Status

Acquisition & Cleaning Steps:

- Successfully integrated image upload via Discord bot.
- Data extraction using Google Cloud Document AI (OCR).
- Structured data stored in Firebase Firestore under:
    - DATA/RAW_DATA
    - DATA/RECEIPTS
    - DATA/SUMMARISED_DATA

Challenges Encountered:

- OCR inconsistencies in receipt formats.
- JSON serialization issues resolved by sanitizing data for Firebase compatibility.
- Utilizing the Ollama Models + Langchain + Langraph, we tried to create a classifier, encountering:
    - **Performance Issues**: Slow response times, especially from high-quality models.
    - **Accuracy Issues**: Incorrect computational results (e.g., classifying a bill and returning clearly incorrect totals like 20 + 30 = 90), which are unacceptable for practical use.

2. Feature Engineering

Selected Features:

- **Item Categories**: Derived via LLM classification (Groceries, Fast Food, Electronics, etc.)
- **Receipt Total**: Extracted and validated using multiple keys (total_amount, net_amount, total_tax_amount).
- **Line Item Structuring**: Standardized format ({"Item": {"QTY": Quantity, "PRICE": Price}}).

3. Baseline Model / Core Prototype

Model Implementation:

- Classification and summarization of receipt items using LangChain with Ollama models (llama3, mistral, llava).

Initial Results:

- Accurate classification demonstrated.
- Confidence scoring mechanism implemented to ensure reliability.

4. Updated Technical Approach

Revised Components:

- Confirmed use of Flask for backend (API fully functional).
- Firebase confirmed as primary database solution.
- API designed for versatile integration, enabling usage across various platforms.

Planned Next Steps:

- Implement regression model for predictive analytics (forecast weekly/monthly spend).
- Enhance Discord bot response times and interaction quality.
- Begin frontend implementation with designed mockups.

5. Progress & Next Steps

Milestone 1 Plan Recap:

- **Bot Integration**: Implement and test Discord bot for receipt image uploads.
- **API Routing**: Backend API design to route uploads and metadata.
- **OCR & Parsing**: Setup and utilize Google Vision API → transition to Vertex AI for improved accuracy.
- **Data Cleaning & Schema Definition**: Standardize data structures and remove redundant data.
- **NLP Categorization**: Initial NLP model testing (BERT/spaCy), item categorization logic.
- **LLM Integration**: Integration of LLM via Langchain/Ollama for classification and predictive analytics.
- **Regression Model Development**: Aggregate weekly user data, forecast spending.
- **Frontend Prototyping**: Initial dashboard mockups with Streamlit/Dash.
- **CI/CD Setup**: Explore containerization and automation via Docker, Jenkins, Kubernetes.

Actual Milestone 2 Progress:
Completed Steps:

- Discord Bot Integration fully operational.
- Backend API fully functional with OCR integrated via GCP Vertex AI.

- Data Cleaning & Schema storage structured in Firebase.
- LLM Classification implemented with LangChain/Ollama.
- Confidence Scoring mechanism established.
- Feature Engineering clearly defined.
- Cross-platform API structured for versatile integration.

Partially Completed or Revised Steps:

- Frontend mockups defined, actual implementation pending.
- Regression Modeling scheduled as immediate next task.

Not Yet Implemented:

- Traditional NLP (BERT/spaCy) replaced by LLM classification.
- CI/CD Automation (Docker, Jenkins, Kubernetes) planned for future milestones.

Key Adjustments Made from Milestone 1:

- Transitioned from traditional NLP to LLM-based classification for greater flexibility.
- Frontend tooling adjusted for further planning.

Evaluation Summary:
Overall, progress closely aligns with Milestone 1 intentions, with adjustments for improved technology and timing.
Immediate Priority Tasks:

- Finalize regression model for predictive analytics.
- Initiate frontend mockup implementation.
- Enhance Discord bot responsiveness and accuracy.

Detailed Task Breakdown (Next 3 Weeks):

- Week 1: Finalize regression model prototype (time-series forecasting).
- Week 2: Begin frontend implementation, integrate basic API endpoints.
- Week 3: Deploy chatbot enhancements, complete first iteration of the frontend dashboard.

6. Team Roles & Adjustments

- **Jonathan Chacko**: Discord Bot, Backend API, LLM integration.
- **Aliyyah Jackhan**: Data normalization, regression modeling, LLM classification logic refinement.
- **Mohammed Aadil**: Frontend mockups, dashboard development, chatbot summary testing.

7. Risks & Mitigation

- **OCR Errors**: Further custom training on Vertex AI.
- **Frontend Delay**: Early prototyping and frequent check-ins.
- **Cloud Costs**: Monitor GCP usage, evaluate alternative hosting if costs escalate.

Appendix

- GitHub Repository: Spendify Repository
- API Code: main_api.py
- Bot Logic: bot.py
- Classification Logic: receipt_classifier.py