AIG Capstone: Milestone 1 (Week 3) – Project Definition & Planning

1. Project Charter

Modifications to Initial Idea: Expanded scope to include WhatsApp/Discord bot integration, GCP Vertex AI processing, predictive analytics, and web frontend.

Problem Statement: Manual receipt management is inefficient and lacks intelligence. Users need automated tools to track expenses, categorize spending, and gain personalized insights.

Project Goal: Build an AI-based web and chatbot-integrated system that extracts structured data from receipts and provides categorized expense tracking with predictive financial tips.

In-Scope:
- Scanning and processing printed receipts
- Cloud-hosted API for pipeline management
- Google Vision/Vertex Document AI for extraction
- NLP-based categorization and visualization
- AI agent-based predictive insights
- WhatsApp/Discord bot for uploads and summaries
- Web dashboard for insights and future receipt uploads
- Webhook-based triggers for automation and event-driven processing
- Support for multiple input formats including PDFs, scanned and unscanned images

Out-of-Scope: Handwritten receipt recognition, mobile apps, full-scale user authentication system, duplicate checks, dashboard, etc.

2. Data Acquisition & Initial Exploration

Data Sources:
- Dummy data from public receipt datasets
- Planned personal receipt uploads via bot or site

Initial Data Findings:
- Mostly English
- Inconsistent formats
- Requires normalization and noise filtering

Cleaning Plan:
- Remove noise (e.g., offers, irrelevant lines)
- Normalize formats and unify schema
- Map fields for AI training (store, items, price, tax)
- Ensure consistent tagging and extraction of unique Purchase ID for each receipt

3. Proposed Technical Approach

Models/Algorithms:
- OCR: Google Vision API → Vertex Document AI (custom-trained)
- Categorization: BERT/spaCy for item grouping
- Prediction: Langchain or Ollama LLMs via AI agents

Tech Stack:
- Backend: Flask/Django (TBD), Docker, GCP/AWS/Hostinger
- Storage: MySQL/Firebase (TBD)
- Bot: Discord/WhatsApp integration with webhook triggers
- Visualization: Streamlit/Dash for frontend dashboard

Architecture Highlights:
- End-to-end automation via CI/CD and cronjobs
- Document parsing + classification → semantic tagging
- Weekly insights + on-demand queries via bot
- Support for user-specific API behaviors (e.g., simple acknowledgment for Discord, structured data response for business clients)
- Semantic layer for total spend per category, date-tagging, and purchase timeline alignment
- Integration of Jenkins, Git, and Kubernetes for CI/CD and scalable deployment

4. Project Plan & Team Roles

Image Upload & Bot Integration
- Aliyyah: Implement and test weekly summaries via bot
- Jonathan: Set up Discord bot and pipeline trigger integration

Data Ingestion & Routing
- Jonathan: Design backend API for routing uploads to GCP Bucket
- Jonathan: Explore and document data formats and bot submission flows

OCR & Document Processing
- Aadil: Run OCR accuracy tests on dummy/bot data
- Jonathan: Set up Google Vision API and explore transition to Vertex Document AI

Data Cleaning & Preprocessing
- Aadil: Normalize receipt text data and define schema
- Aliyyah: Remove irrelevant or redundant receipt data

NLP Categorization & Semantic Structuring
- Aadil: Select and experiment with NLP models (spaCy/BERT)
- Aadil: Develop initial categorization logic and category mapping

AI Agent & Predictive Layer
- Jonathan: Integrate NLP output with AI agent (Langchain/Ollama) via API

- Aliyyah: Research and document LLMs for predictive analytics
- Aadil: Assist in LLM exploration and prototype predictive model
- Aliyyah: Document predictive model performance and challenges

Frontend Development
- Aliyyah: Create mockups with Streamlit for dashboard insights
- Aadil: Research and test Dash for dashboard visualizations
- Jonathan: Develop login and dashboard functionality for web frontend

Project Management & Collaboration
- All: Use Notion for central documentation
- All: Track progress and assign tasks using Jira-alternative tools
- All: Meet weekly to review tasks and roadblocks
- All: Collaborate on deployment strategy (GCP vs AWS vs Hostinger)

5. Risks and Mitigation
- OCR Errors: Use of trained Vertex model, multiple test cases
- Bot Downtime: Use established SDKs and modular bot design
- Model Generalization: Diverse training samples, fine-tuning
- Cloud Cost: GCP free tier, fallback to AWS or Hostinger as needed
- Webhook failures or scheduling conflicts will be addressed by redundancy and fallback routines

6. Professionalism & Engagement
Preparedness:
- Weekly Notion logs & timeline using Jira alt
- Clear role delegation and technical exploration paths

Engagement:
- Team syncs and milestone reviews are scheduled
- Exploration of hosting, frameworks, and APIs in parallel
- Early prototyping of Discord/WhatsApp interaction modes

7. Process Flow (End-to-End)
Step 1: Image Capture & Upload
- Users take a picture or upload a receipt.
- Upload can occur via Discord, WhatsApp Bot, or other integrated platforms.

Step 2: API Handling & Routing
- The bot/API collects the image, user metadata, and optional custom prompts (e.g., source context).
- API is hosted on a cloud platform (TBD: AWS or Hostinger) and routes the image to a Google Cloud Storage Bucket.

Step 3: Document AI Parsing

- The image is sent to a pre-trained custom GCP Vertex Document AI model.
- The model extracts structured data such as:
  - Store information & metadata
  - Item names, quantities, unit amounts
  - Tax and total amounts
  - Unique Purchase ID (1 per bill)

Step 4: Data Categorization via AI Agent
- Structured data is forwarded to an AI agent (e.g., Langchain or Ollama).
- AI categorizes each item into broader categories, calculates total spend per category, tags the date of purchase, and appends the Purchase ID for linkage.

Step 5: Storage and Summary
- The categorized output is stored in a database (TBD: MySQL or Firebase).
- Summary data is prepared for visualization or messaging.

Step 6: Output Delivery
- Discord/WhatsApp users receive status confirmation (e.g., "It’s processed") or a summarized insight.
- Company platforms receive a structured JSON response with detailed results.

Step 7: Automation & Infrastructure
- CI/CD workflows, webhooks, and cronjobs manage deployments and scheduled insight deliveries.
- Tools include Docker, Git, Jenkins, and Kubernetes for containerization and orchestration.

FLOWCHART OF SPENDIFY