AIccountant

DESIGN LOGIC

Overview

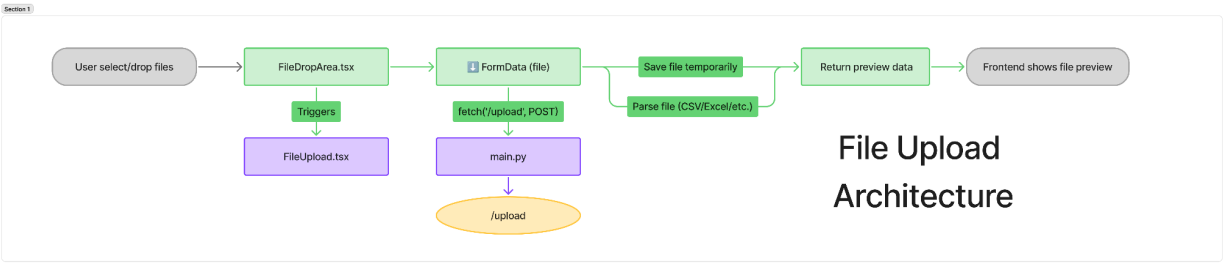
AICOUNTANT is AI-powered application (AI Agent) built to automate financial reconciliation tasks and generate insightful financial summaries. It is designed for finance teams and CPA firms seeking a smarter alternative to manual reconciliation workflows. The system is structured around modular components to ensure clarity, scalability, and adaptability for future enhancements.

Design Objectives

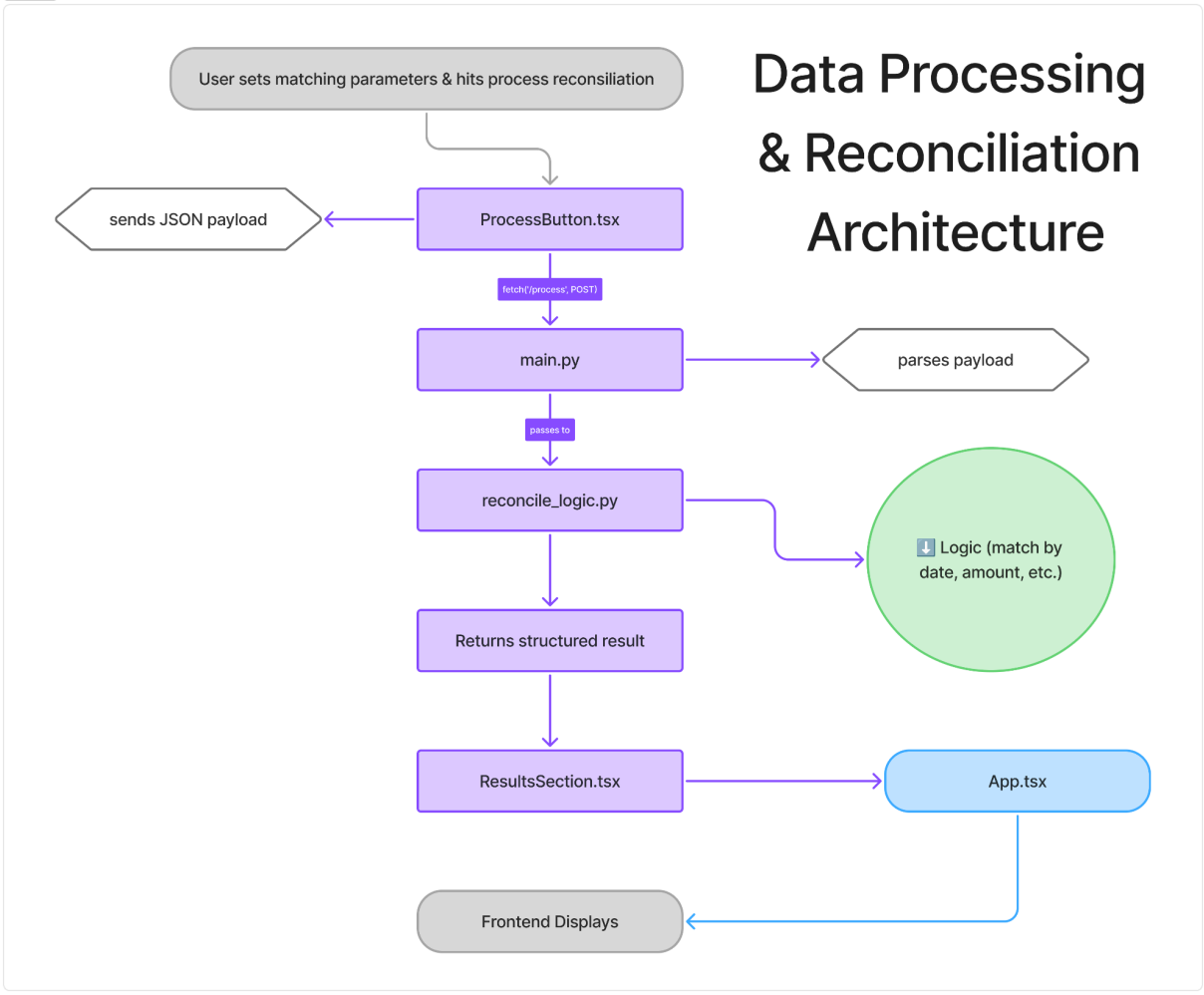
The system was designed with the following key goals in mind:

- **Automation**: Reduce manual reconciliation efforts by leveraging fuzzy matching and AI-based reasoning.
- **User-Friendliness**: Provide a clean, intuitive interface that supports seamless interactions for non-technical users.
- **Intelligence & Adaptability**: Incorporate feedback mechanisms to refine logic over time and generate human-like insights.

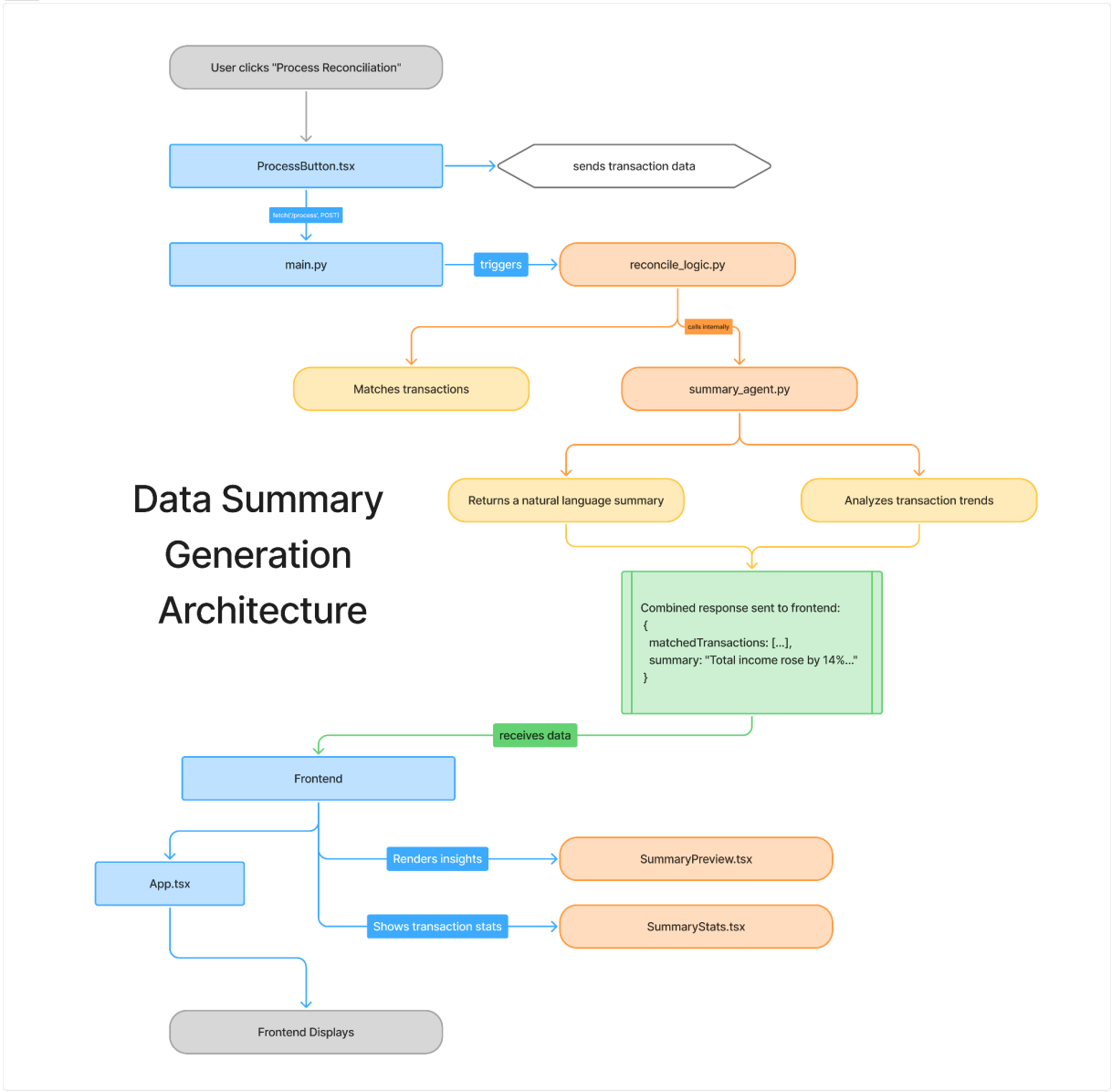
System Architecture

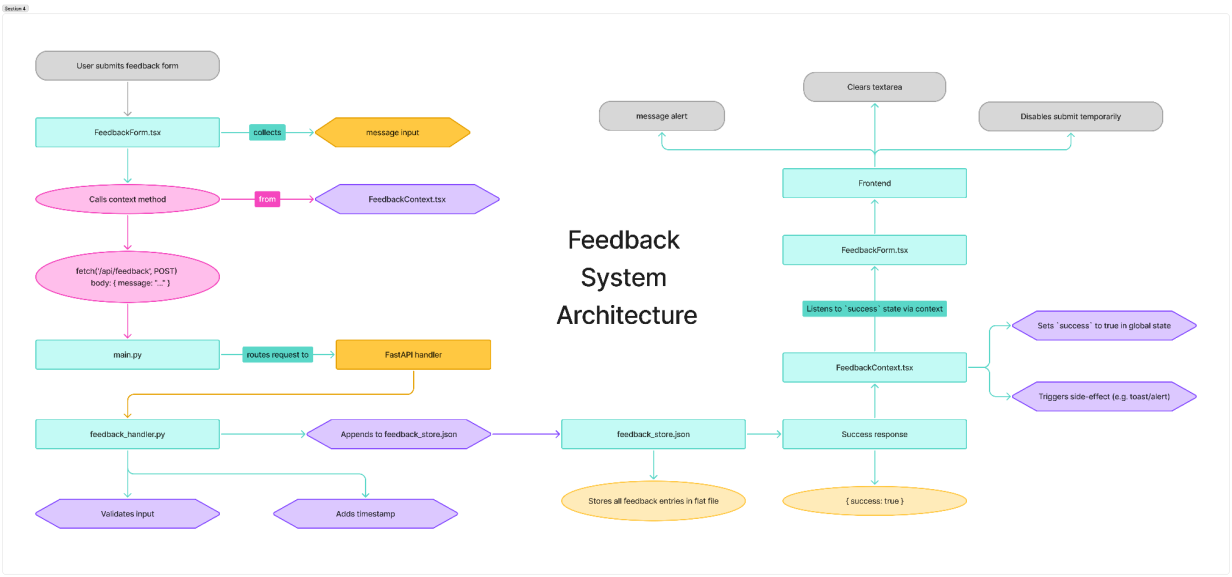


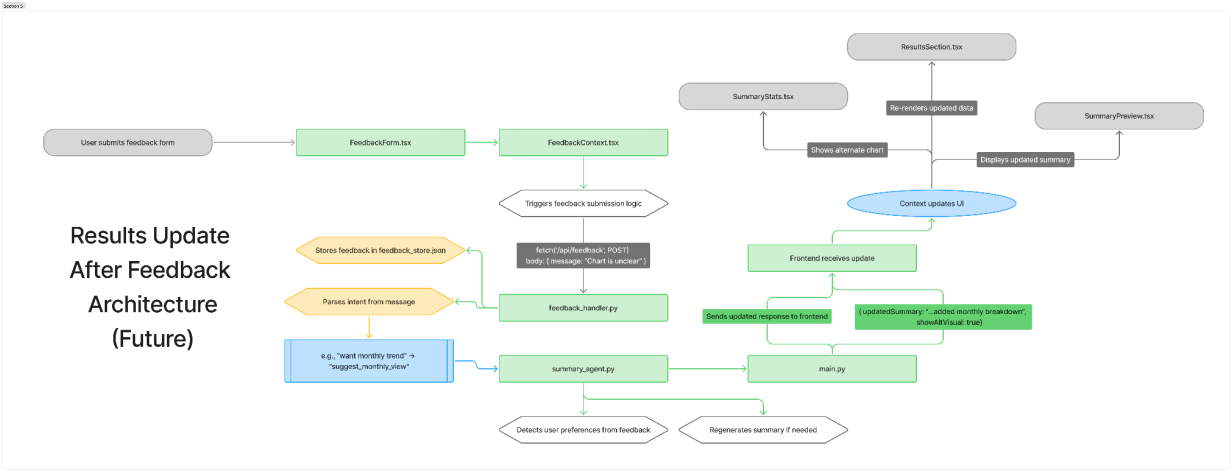
Section 2


Data Processing & Reconciliation Architecture
User sets matching parameters & hits process reconsiliation
ProcessButton.tsx
sends JSON payload
fetch('/process', POST)
main.py
parses payload
passes to
reconcile_logic.py
⬇️ Logic (match by date, amount, etc.)
Returns structured result
ResultsSection.tsx
App.tsx
Frontend Displays

Section 3


User clicks "Process Reconciliation"
ProcessButton.tsx
sends transaction data
fetch('/process', POST)
main.py
triggers
reconcile_logic.py
calls internally
Matches transactions
summary_agent.py
Data Summary Generation Architecture
Returns a natural language summary
Analyzes transaction trends
Combined response sent to frontend:
{
matchedTransactions: [...],
summary: "Total income rose by 14%..."
}
receives data
Frontend
App.tsx
Renders insights
SummaryPreview.tsx
Shows transaction stats
SummaryStats.tsx
Frontend Displays





Core Workflow: Reconciliation

- **User Interaction**:
  Users upload two CSV files (bank and ledger). The frontend sends these files to the backend via a POST request.

- **Backend Processing**:
  The uploaded files are passed to the reconciliation engine, where transactions are compared using:
  - Text similarity (for vendor name or description mismatches)
  - Date proximity

    - Amount thresholds

- **Confidence Scoring**:
  Each potential match is assigned a confidence score. Matches below a predefined threshold are flagged for manual review.

- **Categorization**:
  Transactions are classified into:
    - Confident matches
    - Unmatched transactions
    - Low-confidence matches (suggested but not confirmed)

- **Summary Generation**:
  After reconciliation, the results are forwarded to the `summary_agent.py`, which crafts a plain-language narrative highlighting income/expense trends and anomalies.

- **Frontend Display**:
  The matched data is displayed in a table, while the summary is rendered as a paragraph-style insight.

Feedback System

The application includes a feedback mechanism that allows users to correct inaccurate matches.

- When a user submits feedback, it is stored in a local JSON file (`feedback_store.json`).
- Although feedback is currently stored passively, the future goal is to:
    - Update matching thresholds and rules dynamically.
    - Reprocess reconciliations based on cumulative user feedback.
    - Incorporate feedback into machine learning or rule-based tuning.

Agentic Behavior

AICOUNTANT incorporates basic agentic capabilities:

- **Decision Autonomy**: Automatically performs transaction matching and escalates low-confidence cases.

- **Natural Language Generation**: Creates plain-English summaries tailored to finance teams.
- **Feedback Awareness (planned)**: Learns from past user interactions to improve performance over time.

Design Considerations

- **Simplicity in Storage**: The project uses local file storage for uploaded files and feedback. This avoids database complexity in early stages.
- **Separation of Concerns**: Each functionality (matching, summarizing, feedback) is handled in separate scripts to improve readability and maintainability.
- **Scalability**: The architecture allows easy transition to cloud storage, databases, and external APIs if the system needs to scale.

Design Considerations

With more time and resources, the following improvements are envisioned:

Dynamic Reprocessing: Update summaries and matches instantly when feedback is submitted.
Plugin Integrations: Support importing data from platforms like QuickBooks, Xero, or Zoho Books.
Voice Narration: Add audio narration for the generated summaries for better accessibility.
User Roles & Permissions:

Auditor: View-only mode.
Reconciler: Edit matches and provide feedback.
Admin: Full access including configuration.

Collaborative Features: Add notes and comments on specific transactions for teamwork and discussion.
Version History: Enable rollback and audit trail of reconciliation sessions.
Advanced Vendor Matching: Implement normalization techniques to match different variations of vendor names (e.g., “TATA Motors Ltd.” vs “TATA MOTORS”).

Conclusion

AICOUNTANT was built to streamline the traditionally tedious reconciliation process using AI and automation. Its modular, user-centered design allows for easy maintenance and future enhancements. While the current MVP delivers core reconciliation and summarization features, the foundation is set for developing a fully adaptive, intelligent accounting assistant.