Design Generated Document:

Okay, here's a detailed technical design document for the automated dispute resolution system, incorporating the specified technologies and addressing the user stories.

**1. Introduction**

**1.1 Overview**

The Automated Dispute Resolution System (ADRS) is designed to streamline and automate the process of resolving billing disputes, specifically focusing on revenue leakage within the billing workflow. The system leverages Google Agent Space, Agentic AI, Generative AI (LLMs), LangChain, Gemini Flash, Gemini Embedding, FAISSDB, and Google BigQuery to provide an intelligent and efficient solution for dispute management.

**1.2 Purpose**

The primary purpose of the ADRS is to:

* Automate the dispute intake and matching process.
* Reduce manual effort in dispute resolution.
* Increase recovery rates from disputes.
* Decrease dispute resolution time.
* Proactively detect and prioritize revenue leakage.
* Identify and target high-impact revenue leakage scenarios.
* Generate basic insights for root cause analysis.
* Improve customer experience by providing timely and accurate updates.

**1.3 Scope**

The initial scope (MVP) of the ADRS will focus on:

* Billing disputes related to invoice discrepancies (amount, quantity, services).
* Data ingestion from Google BigQuery, Excel spreadsheets, and PDF documents.
* Automated dispute matching based on semantic similarity.
* Generation of suggested resolution steps using LLMs.
* Automated email communication with customers.
* Basic reporting on dispute volume and resolution time.
* Efficient handling of data volume and transaction processing for MVP scope.

**1.4 Problem Statement**

Currently, dispute resolution is a manual and time-consuming process, leading to:

* **Cash Flow Delays:** Unresolved disputes tie up revenue, impacting cash flow.
* **High Operational Costs:** Manual investigation and resolution require significant staff time.
* **Billing Inaccuracies:** Underlying billing errors contribute to disputes and revenue leakage.
* **Poor Customer Experience:** Slow resolution times and lack of communication frustrate customers.

**1.5 Business Objectives**

The ADRS aims to achieve the following quantifiable benefits:

* **Reduce Dispute Resolution Time by 50%:** Automating key steps will significantly speed up the process.
* **Increase Dispute Recovery Rate by 20%:** Improved matching and resolution strategies will lead to higher recovery rates.
* **Decrease Operational Costs by 30%:** Automation will reduce the need for manual intervention.
* **Improve Customer Satisfaction (CSAT) Score by 15%:** Faster resolution and better communication will enhance customer experience.
* **Reduce Revenue Leakage by 10%:** Proactive detection and root cause analysis will help prevent future disputes.

**2. System Architecture**

**2.1 High-Level Architecture**

The ADRS is built on a modular and scalable architecture, leveraging Google Agent Space as the central orchestration layer. It integrates various components, including Agentic AI agents, Generative AI (LLMs), LangChain, Gemini Embedding, FAISSDB, and Google BigQuery.

**2.2 Component Interactions**

1. **Data Ingestion:** Data is ingested from various sources (BigQuery, Excel, PDFs, Email) by the **Data Ingestion Agent**.
2. **Data Harmonization:** The ingested data is transformed and harmonized into a consistent format.
3. **Dispute Submission:** Customers can submit disputes through an online form or email.
4. **Dispute Matching:** The **Dispute Matching Agent** uses Gemini Embedding and FAISSDB to match new disputes with existing disputes or known issues.
5. **Dispute Review:** A Dispute Analyst reviews the automatically matched disputes and validates the matches.
6. **Dispute Resolution:** The **Resolution Agent** uses Gemini Flash to generate potential resolution steps.
7. **Resolution Execution:** A Dispute Analyst executes the resolution steps and documents the outcome.
8. **Communication:** The **Communication Agent** automatically generates email communications to the customer regarding the dispute status and resolution.
9. **Root Cause Analysis:** The **Root Cause Analysis Agent** uses Gemini Flash to analyze resolved disputes and identify potential root causes of revenue leakage.
10. **Reporting:** Managers can view reports on dispute volume, resolution time, and recovery rates.

**2.3 Architecture Diagram**

graph LR

subgraph Data Sources

A[Google BigQuery] --> B(Data Ingestion Agent)

C[Excel Files] --> B

D[PDF Documents] --> B

E[Email] --> B

end

subgraph AgentSpace

B --> F(Data Harmonization Layer)

F --> G(Dispute Submission Form)

G --> H(Dispute Matching Agent)

H --> I(FAISSDB)

I --> H

H --> J(Dispute Analyst Review)

J --> K(Resolution Agent)

K --> L(Gemini Flash LLM)

L --> K

K --> M(Dispute Analyst Execution)

M --> N(Communication Agent)

N --> O(Customer)

M --> P(Root Cause Analysis Agent)

P --> L

P --> Q(Reporting Dashboard)

end

subgraph External Components

O --> N

end

style B fill:#f9f,stroke:#333,stroke-width:2px

style H fill:#f9f,stroke:#333,stroke-width:2px

style K fill:#f9f,stroke:#333,stroke-width:2px

style P fill:#f9f,stroke:#333,stroke-width:2px

classDef agentFill fill:#f9f,stroke:#333,stroke-width:2px;

class B,H,K,P agentFill;

**2.4 Scalability, High Performance, and Resiliency**

* **Scalability:** The system is designed to scale horizontally by adding more instances of the Agentic AI agents and the data harmonization layer. FAISSDB is also scalable and can be distributed across multiple nodes. Google BigQuery is inherently scalable.
* **High Performance:** Gemini Embedding and FAISSDB enable efficient similarity search. Caching mechanisms will be implemented to reduce latency. Asynchronous processing will be used for long-running tasks.
* **Resiliency:** The system will be deployed in a highly available configuration with redundancy. Error handling and retry mechanisms will be implemented to handle failures gracefully. Monitoring and alerting will be set up to detect and respond to issues proactively.

**3. Functional Requirements**

Based on the user stories, the system must be able to perform the following actions:

* **Dispute Intake:**
  + Accept dispute submissions through an online form and email.
  + Parse data from invoice PDFs and contact PDFs.
  + Extract data from Google BigQuery.
* **Dispute Matching:**
  + Generate Gemini Embeddings for new disputes.
  + Search FAISSDB for similar disputes.
  + Present potential matches to the analyst with confidence scores.
  + Allow the analyst to accept or reject matches.
* **Dispute Resolution:**
  + Generate a summarized dispute history using Gemini Flash.
  + Generate potential resolution steps using Gemini Flash.
  + Allow the analyst to modify or add resolution steps.
  + Track the execution of resolution steps.
  + Record the outcome of each step.
  + Update the dispute status automatically.
* **Communication:**
  + Generate personalized email communications to the customer.
  + Send emails automatically based on predefined triggers.
* **Root Cause Analysis:**
  + Analyze resolved disputes using Gemini Flash.
  + Identify potential root causes of revenue leakage.
  + Provide supporting evidence for each potential root cause.
* **Reporting:**
  + Generate reports on dispute volume, resolution time, and recovery rates.
  + Allow filtering reports by date range, dispute type, and other criteria.
  + Export reports in various formats (CSV, PDF).
* **Administration:**
  + Configure system parameters (e.g., thresholds for automatic matching, email templates).
  + Validate input parameters.
  + Log all configuration changes.

**4. Non-Functional Requirements**

* **Scalability:** The system must be able to handle 1000 disputes per day in the MVP phase and scale to 10,000 disputes per day in the future.
* **Performance:**
  + Dispute submission: Response time should be less than 2 seconds.
  + Dispute matching: Similarity search should complete in less than 1 second.
  + Report generation: Reports should be generated in less than 5 seconds.
* **Availability:** The system should be available 99.9% of the time.
* **Maintainability:** The system should be designed with modular components and well-documented code to facilitate maintenance and updates.
* **Security:** The system must protect sensitive financial information and comply with relevant financial regulations.
* **Usability:** The user interface should be intuitive and easy to use for billing analysts and customer service representatives.
* **Reliability:** The system should be reliable and accurate in processing disputes and generating reports.
* **Testability:** The system should be designed to be easily testable, with unit tests, integration tests, and end-to-end tests.

**5. User Interface**

The user interface will be built using **Streamlit** for rapid prototyping and deployment.

**5.1 User Roles**

* **Customer:** Submits disputes and receives updates.
* **Dispute Analyst:** Reviews disputes, validates matches, executes resolution steps, and documents outcomes.
* **Manager:** Views reports and monitors system performance.
* **Administrator:** Configures system parameters.

**5.2 Key Screens**

* **Dispute Dashboard:**
  + Displays a list of disputes with their status, priority, and assigned analyst.
  + Allows filtering and sorting of disputes.
  + Provides a summary of key metrics (dispute volume, resolution time, recovery rate).
* **Dispute Resolution Interface:**
  + Displays dispute details, including invoice information, customer information, and dispute description.
  + Shows potential matches with confidence scores.
  + Presents a summarized dispute history generated by Gemini Flash.
  + Displays potential resolution steps generated by Gemini Flash.
  + Allows the analyst to modify or add resolution steps.
  + Provides a form to record the outcome of each step.
  + Updates the dispute status automatically.
* **Root Cause Analysis Tool:**
  + Displays a list of potential root causes of revenue leakage, ranked by likelihood.
  + Provides supporting evidence for each potential root cause.
  + Allows the analyst to investigate and validate the root causes.
* **Reporting Dashboard:**
  + Displays customizable reports on dispute volume, resolution time, and recovery rates.
  + Allows filtering reports by date range, dispute type, and other criteria.
  + Provides options to export reports in various formats.

**5.3 User Flow**

1. **Dispute Submission:** Customer submits a dispute through the online form or email.
2. **Dispute Matching:** The system automatically matches the dispute with existing disputes or known issues.
3. **Dispute Review:** A Dispute Analyst reviews the matched disputes and validates the matches.
4. **Dispute Resolution:** The analyst uses the resolution interface to execute resolution steps and document the outcome.
5. **Communication:** The system automatically sends email communications to the customer regarding the dispute status and resolution.
6. **Root Cause Analysis:** The system analyzes resolved disputes to identify potential root causes of revenue leakage.
7. **Reporting:** Managers view reports to monitor system performance and identify areas for improvement.

**6. Software Components**

* **6.1 Google Agent Space:**
  + **Role:** Central orchestration layer for managing and coordinating the Agentic AI agents.
  + **Responsibilities:**
    - Agent lifecycle management (creation, deployment, monitoring).
    - Inter-agent communication and coordination.
    - Resource allocation and management.
    - Security and access control.
* **6.2 Agentic AI Agents:**
  + **Design Principles:** Autonomy, proactivity, social ability.
  + **Agents:**
    - **Data Ingestion Agent:** Extracts data from various sources (BigQuery, Excel, PDFs, Email).
    - **Dispute Matching Agent:** Matches new disputes with existing disputes or known issues using Gemini Embedding and FAISSDB.
    - **Resolution Agent:** Generates potential resolution steps using Gemini Flash.
    - **Communication Agent:** Automatically generates email communications to the customer.
    - **Root Cause Analysis Agent:** Analyzes resolved disputes and identifies potential root causes of revenue leakage using Gemini Flash.
* **6.3 Generative AI (LLMs): Gemini Flash**
  + **Purpose:** Intelligent processing, insights, and natural language understanding.
  + **Use Cases:**
    - Understanding natural language in dispute descriptions.
    - Generating potential resolution steps or communication drafts.
    - Summarizing dispute history.
    - Providing insights for root cause analysis.
* **6.4 LangChain**
  + **Role:** Orchestrating LLM calls, managing agent interactions with external tools/APIs.
  + **Functionality:**
    - Chaining together LLM calls to perform complex tasks.
    - Connecting agents to external tools and APIs (e.g., BigQuery, email service).
    - Managing agent memory and context.
* **6.5 Vector Database (FAISSDB)**
  + **Purpose:** Storing and querying Gemini embeddings of dispute characteristics for efficient similarity matching.
  + **Functionality:**
    - Indexing Gemini embeddings.
    - Performing similarity searches based on cosine similarity.
    - Returning the most similar disputes with confidence scores.
* **6.6 Data Harmonization Layer**
  + **Purpose:** Transforming and harmonizing data from various sources into a consistent format.
  + **Functionality:**
    - Extracting data from Google BigQuery, Excel, and PDF.
    - Cleaning and transforming the data.
    - Mapping the data to a common schema.
* **6.7 BigQuery Tables**
  + **Disputes Table:**
    - dispute\_id (INT64, PRIMARY KEY)
    - customer\_id (INT64, FOREIGN KEY referencing customers.customer\_id)
    - invoice\_id (INT64, FOREIGN KEY referencing invoices.invoice\_id)
    - dispute\_date (DATE)
    - dispute\_amount (NUMERIC)
    - dispute\_reason (STRING)
    - dispute\_description (STRING)
    - dispute\_status (STRING)
    - resolution\_date (DATE)
    - resolution\_amount (NUMERIC)
    - resolution\_notes (STRING)
    - embedding (BYTES) // Gemini Embedding
  + **Customers Table:**
    - customer\_id (INT64, PRIMARY KEY)
    - customer\_name (STRING)
    - customer\_email (STRING)
    - customer\_phone (STRING)
  + **Invoices Table:**
    - invoice\_id (INT64, PRIMARY KEY)
    - customer\_id (INT64, FOREIGN KEY referencing customers.customer\_id)
    - invoice\_date (DATE)
    - invoice\_amount (NUMERIC)
    - invoice\_due\_date (DATE)
  + **Payments Table:**
    - payment\_id (INT64, PRIMARY KEY)
    - invoice\_id (INT64, FOREIGN KEY referencing invoices.invoice\_id)
    - payment\_date (DATE)
    - payment\_amount (NUMERIC)
  + **Dispute History Table:**
    - history\_id (INT64, PRIMARY KEY)
    - dispute\_id (INT64, FOREIGN KEY referencing disputes.dispute\_id)
    - timestamp (TIMESTAMP)
    - action (STRING)
    - user (STRING)
    - notes (STRING)
  + **Revenue Leakage Events Table:**
    - leakage\_id (INT64, PRIMARY KEY)
    - customer\_id (INT64, FOREIGN KEY referencing customers.customer\_id)
    - invoice\_id (INT64, FOREIGN KEY referencing invoices.invoice\_id)
    - leakage\_date (DATE)
    - leakage\_amount (NUMERIC)
    - leakage\_type (STRING)
    - leakage\_description (STRING)
    - root\_cause (STRING)

**7. Database Schema**

(See BigQuery Tables section above for detailed schema)

**8. Security**

* **Data Security:**
  + Sensitive financial information will be encrypted at rest and in transit.
  + Access to data will be restricted based on user roles and permissions.
  + Data masking and anonymization techniques will be used to protect sensitive data.
* **Authentication and Authorization:**
  + Users will be authenticated using Google Cloud Identity and Access Management (IAM).
  + Authorization will be based on user roles, with granular permissions for each role.
  + Multi-factor authentication will be enforced for all users.
* **Network Security:**
  + The system will be deployed in a Virtual Private Cloud (VPC) with network security rules to restrict access.
  + Firewalls will be used to protect against unauthorized access.
  + Regular security audits will be conducted to identify and address vulnerabilities.
* **API Security:**
  + APIs will be secured using API keys and OAuth 2.0.
  + Rate limiting will be implemented to prevent abuse.
  + Input validation will be performed to prevent injection attacks.
* **Compliance:**
  + The system will comply with relevant financial regulations, such as PCI DSS and GDPR.
  + Regular compliance audits will be conducted to ensure adherence to regulations.

**9. Conclusion**

The Automated Dispute Resolution System provides a comprehensive solution for streamlining and automating the dispute resolution process. By leveraging Google Agent Space, Agentic AI, Generative AI (LLMs), LangChain, Gemini Embedding, FAISSDB, and Google BigQuery, the system can significantly reduce manual effort, increase recovery rates, decrease resolution time, and improve customer experience.

**9.1 Important Considerations**

* **Data Quality:** The accuracy and completeness of the data are critical for the success of the system. Data validation and cleansing processes should be implemented.
* **LLM Fine-Tuning:** The performance of the LLMs can be further improved by fine-tuning them on dispute-specific data.
* **Agent Training:** The Agentic AI agents should be continuously trained and improved based on feedback and performance data.
* **Monitoring and Alerting:** Robust monitoring and alerting mechanisms are essential for detecting and responding to issues proactively.

**9.2 Possible Future Improvements**

* **Integration with other systems:** Integrate with CRM, ERP, and other relevant systems.
* **Advanced analytics:** Implement advanced analytics to identify trends and patterns in disputes.
* **Predictive modeling:** Use machine learning to predict the likelihood of disputes and proactively address potential issues.
* **Automated resolution:** Automate the resolution of simple disputes without human intervention.

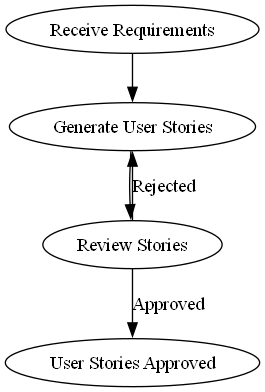
**9.3 Known Risks and Mitigation**

* **Data Quality Issues:** Implement data validation and cleansing processes.
* **LLM Accuracy:** Fine-tune LLMs on dispute-specific data and implement human review for critical decisions.
* **Security Vulnerabilities:** Conduct regular security audits and implement security best practices.
* **Scalability Issues:** Design the system with a scalable architecture and monitor performance closely.

**9.4 Technical Debt and Dependencies**

* **Dependency on Google Cloud Services:** The system is heavily dependent on Google Cloud services, which could be a risk if there are outages or changes to the services.
* **Technical Debt:** The initial implementation may have some technical debt due to the rapid prototyping approach. This should be addressed in future iterations.

**Generated Diagram**



Design Diagram