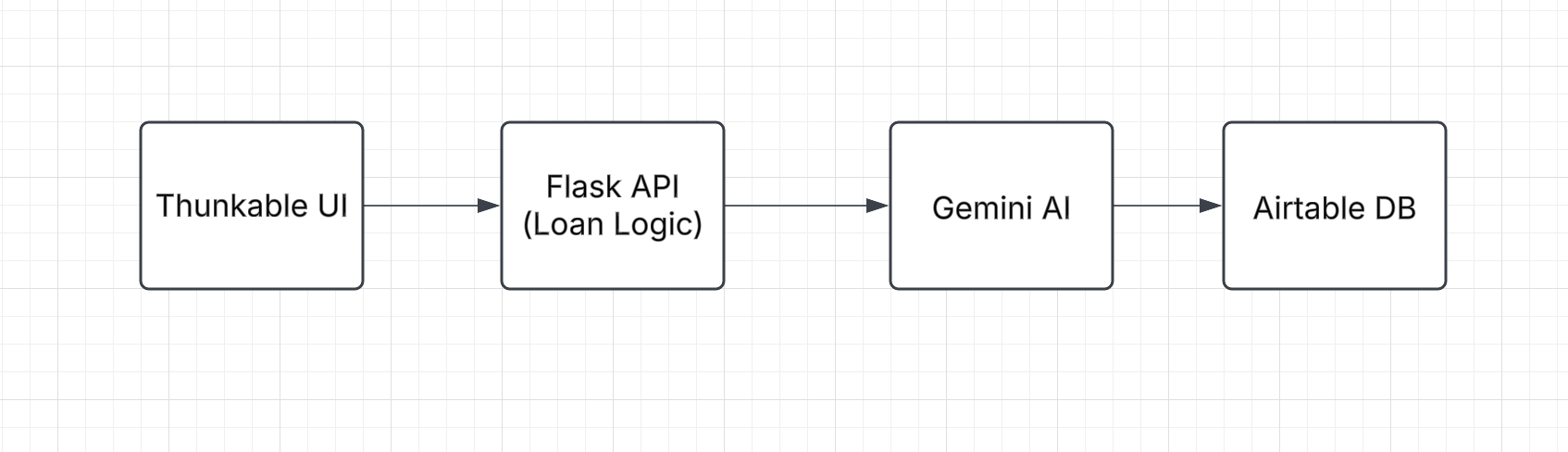
**Task 2 : Building a Full Stack AI Powered Customer Portal**

**System Architecture Diagram:**

****

**MVP Design & Implementation**

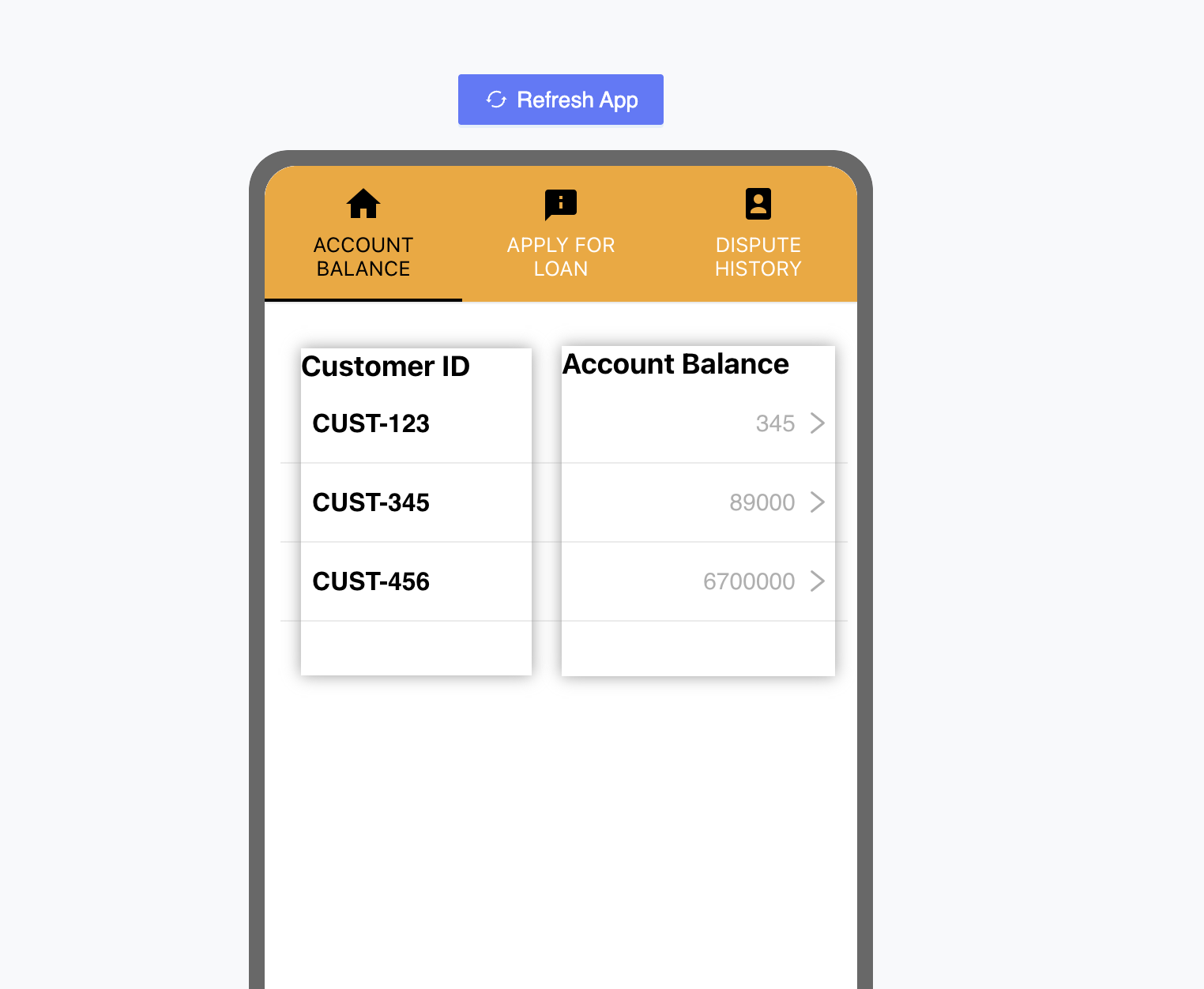
**Banking Application -** [**Link**](https://x.thunkable.com/copy/47e7560626eea410e74c733700028ac7)

I used Thunkable to build my full stack web application as it offers the integration with Airtable database and also it has the feature to connect with APIS using the concept of blocks which is provided in the platform.

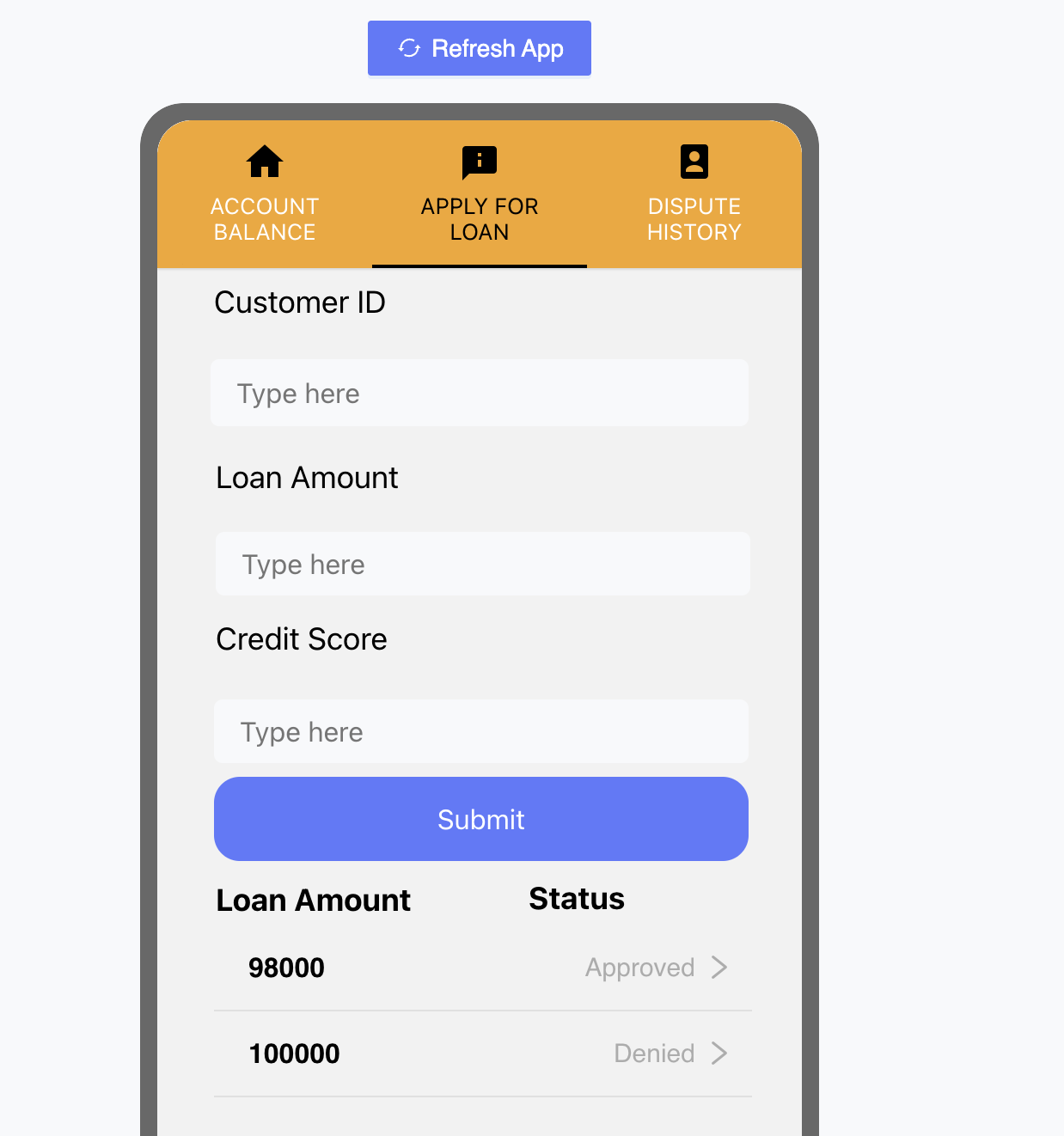
**UI Interface:**

Here it is pretty much simple to design the frontend interface and I made three different options for the user to check his **Account balance**, **Apply for loan** and also track his **Dispute history.**

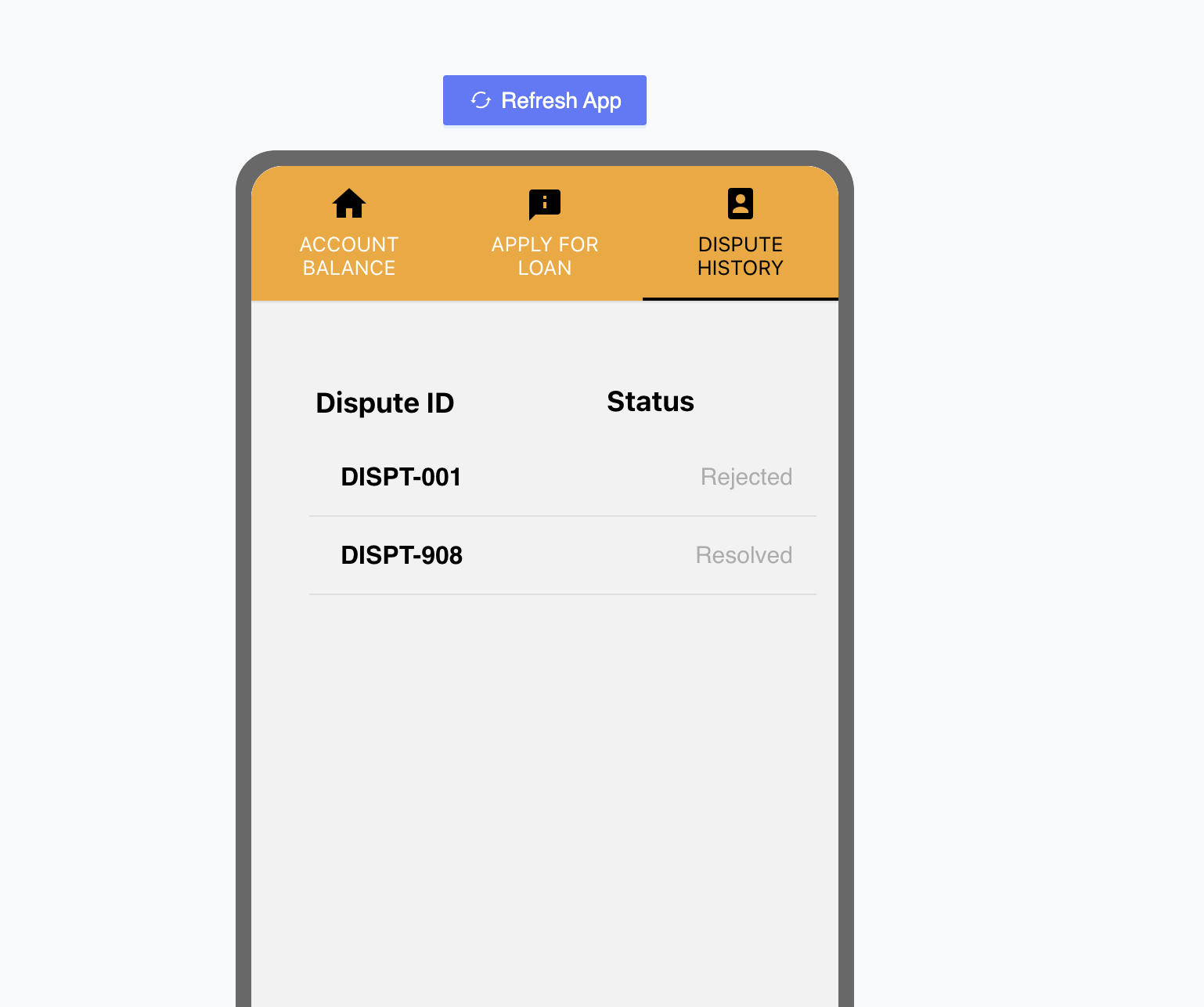
**Account Balance:**

****

**Apply for Loan:**

****

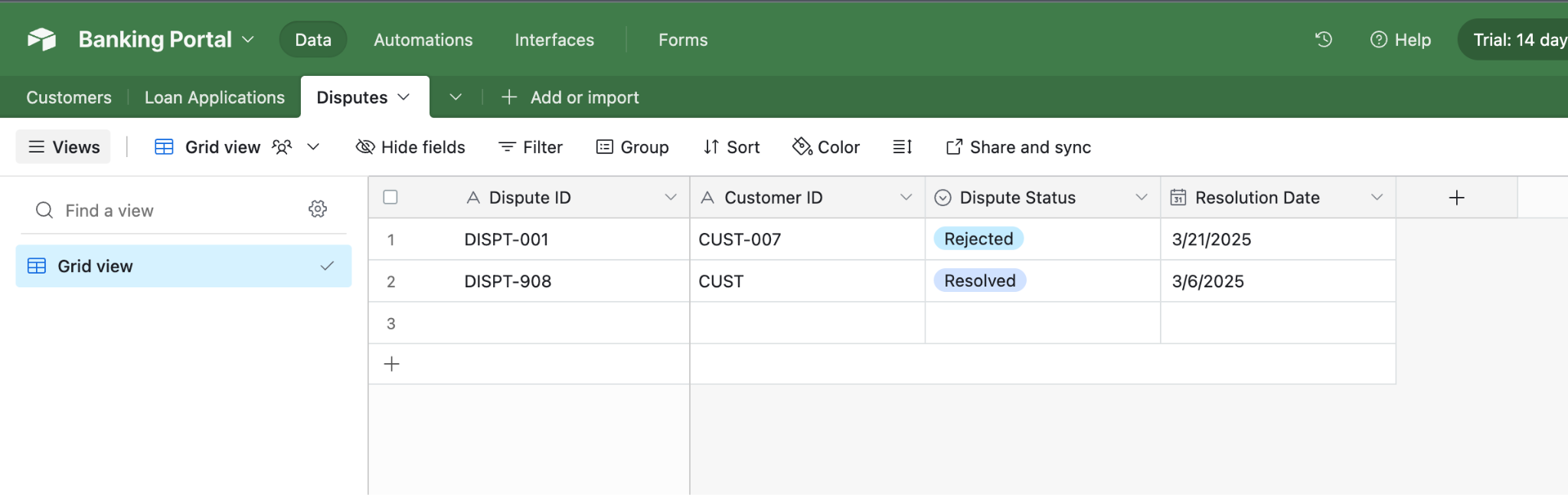
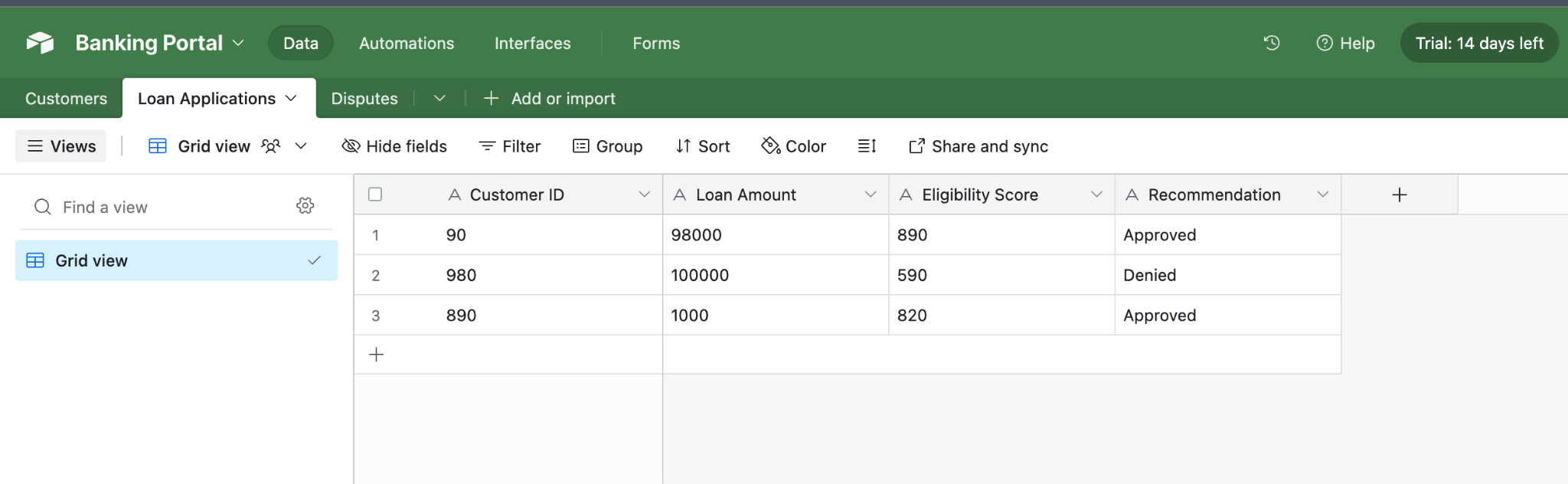
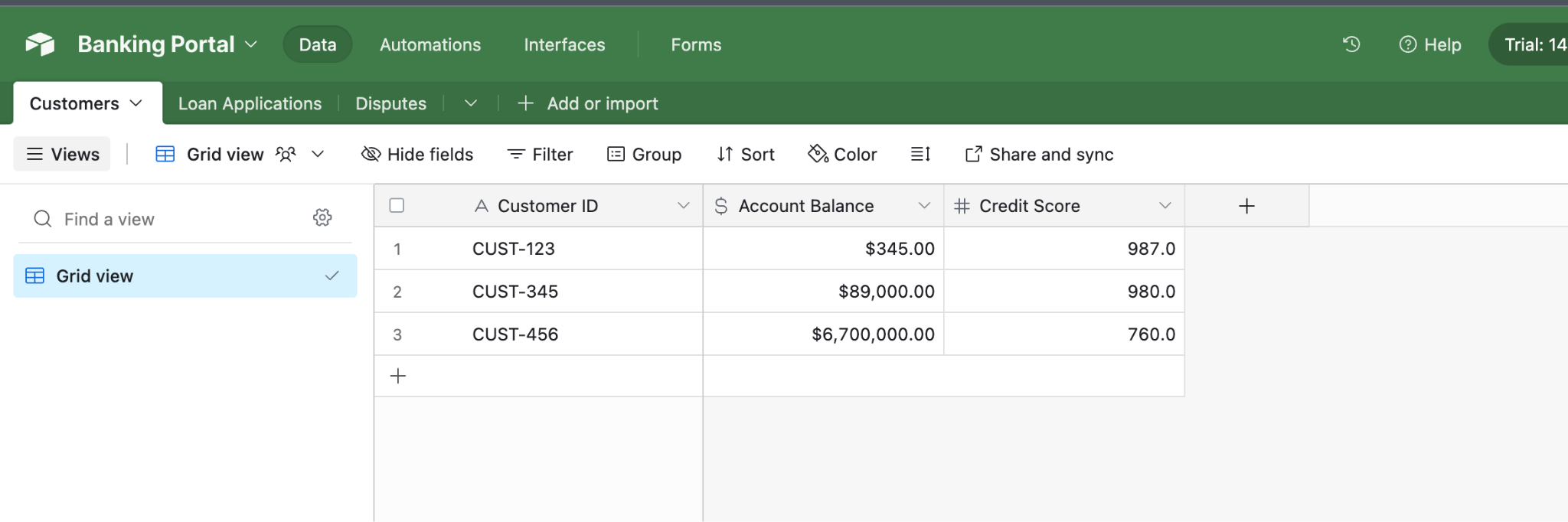
**Dispute History**

****

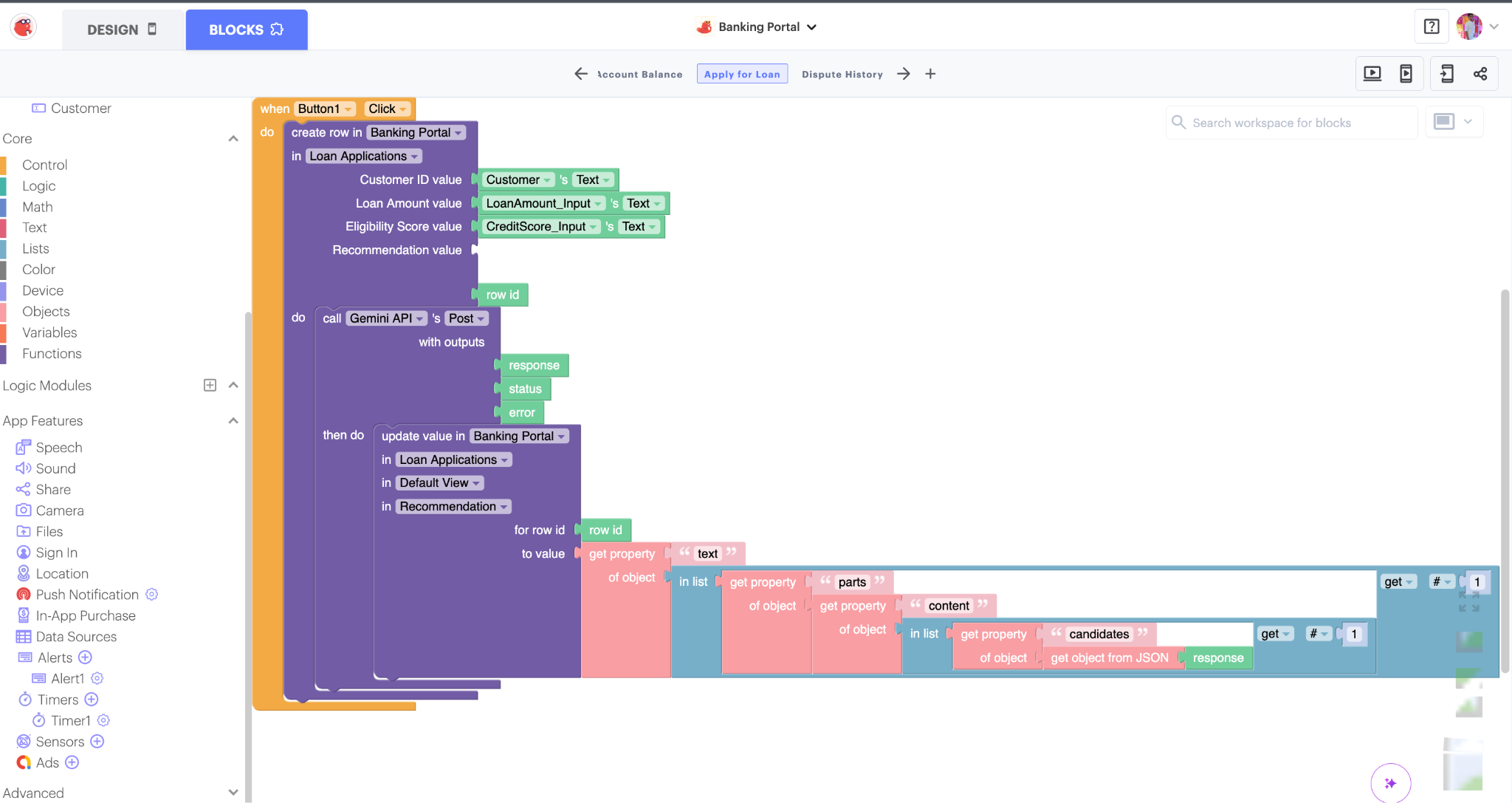
**Backend Implementation:**

The Backend implementation is like a combination of both Blocks ( Thunkable feature and Airtable for storage). Thunkable offers integration with databases and we just have to implement the business logic to connect the UI and Database using blocks.

**Databases:**

****

**Business logic to connect Storage and UI:**

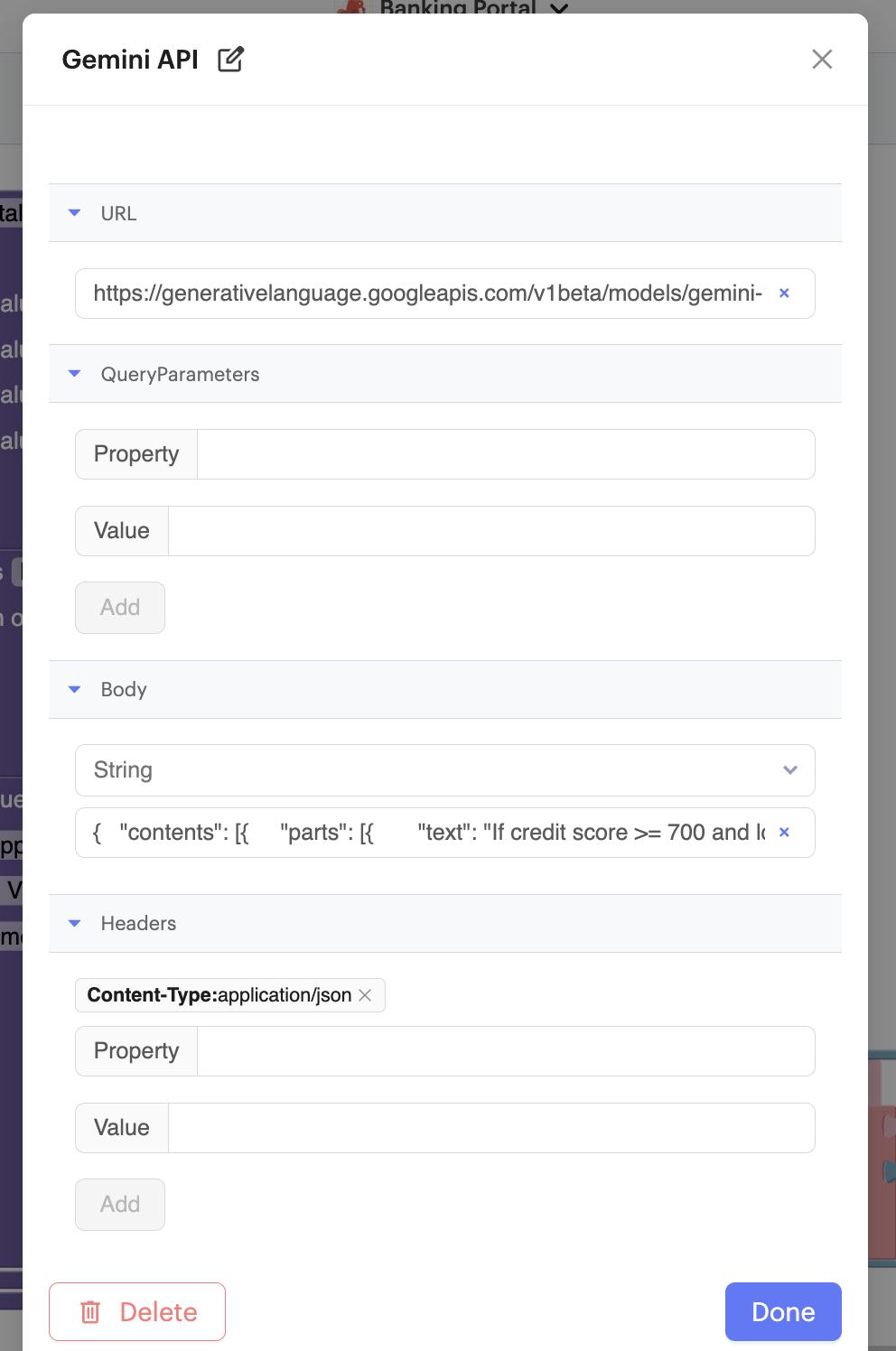
****

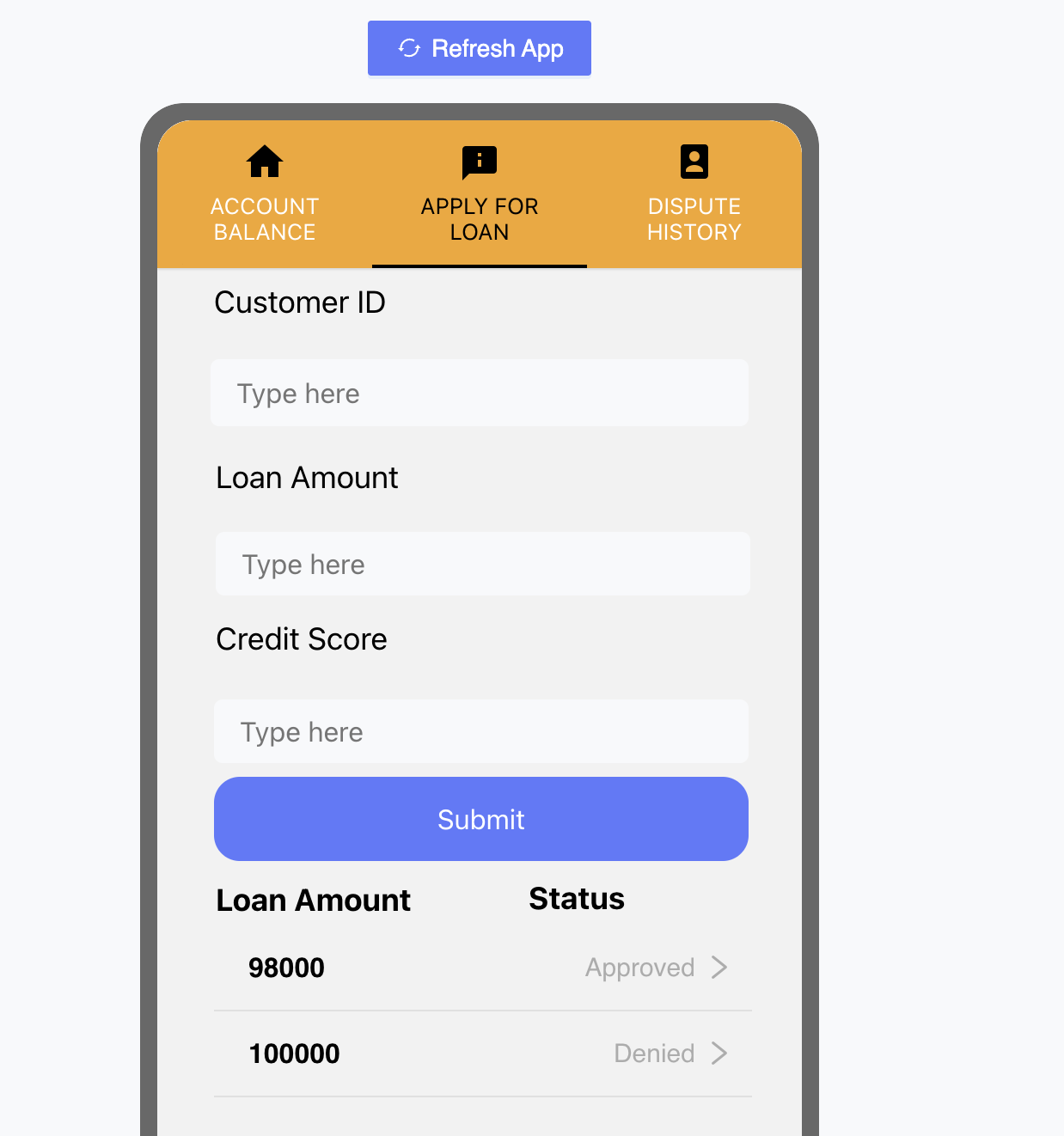
**You can make API calls, update the result in Databases and also reflect the changes in the UI.**

Here these automated tools help in building a full stack MVP with all the necessary features that a basic Application requires.

AI Generated responses for Loan Eligibility:

We have used the Gemini API where it recommends the Banking application whether a loan needs to be approved for a particular user or not. We pass dynamic content of the **credit score** **loan amount** that a user enters while applying for a loan and it reflects back with a recommendation of approval or rejection.

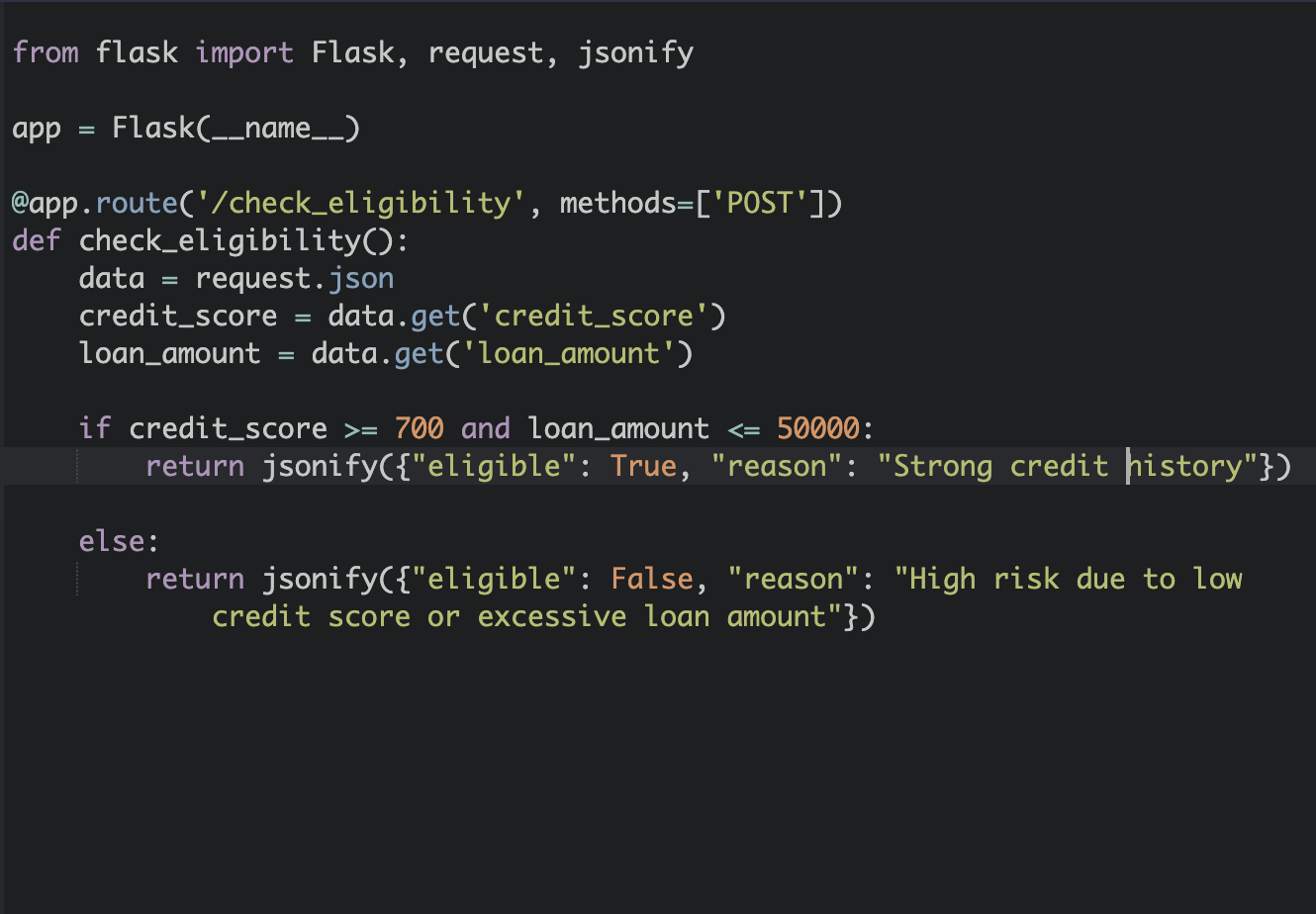


****

I specifically used the blocks feature as it offers for Gemini API integration and also we have easily updated the data among the UI and Database as well.

I even developed small API where you can deploy to flask and call from thunkable blocks:

API Implementation Details:



**Error Handling and Security:**

**I**nput validation blocks in Thunkable ensure users enter numeric loan amounts and valid credit scores. Failed API calls to Gemini trigger retries and user-facing error messages.

We want to mask Sensitive data (e.g., credit scores) in Airtable, and API keys are stored in Thunkable’s encrypted environment variables.

**Rapid MVP Development in 24 Hours**

**Step 1** (4 Hours):

Build UI mockups on Thunkable with placeholder data.

**Step 2** (6 Hours): Set up Airtable tables and Thunkable blocks to connect UI ↔ AI API.

**Step 3** (8 Hours): Train a basic AI model (e.g., Google AutoML or a decision tree) for eligibility checks.

**Step 4** (6 Hours): Test end-to-end flow and refine error handling.

**Why This Works:**

No-code tools handle 80% of the work, letting you focus on AI integration.

**Trade-off**:

Thunkable was prioritized over Retool for its drag-and-drop mobile-first UI capabilities, while Airtable was chosen over Firebase for its spreadsheet-like interface, which simplifies data management during rapid prototyping.

**Low-Code Acceleration**

**Frontend**: Use Retool for pre-built banking UI components (e.g., balance displays).

**Backend**: Use Firebase for real-time data syncing and authentication.

**AI**: Embed Gemini API for dynamic loan eligibility explanations.