# Service Booking System (Using Flask + Firestore)

## Problem Statement

Build a small backend system using Flask as the server framework and Firestore as the database. The system should manage a set of services (like appointments, room bookings, or haircut slots). Each service has a list of available time slots. Users can pick a slot and book it. The backend should store services, slots, and bookings in Firestore, and all operations should be handled through Flask.

## Solution Overview

To solve this problem, a Flask-based REST backend was implemented with Google Firestore as the database.

The system allows:

- Creation and retrieval of services with predefined time slots
- Slot-based booking by users
- Validation against Firestore to prevent duplicate bookings
- Clean and structured storage of service and booking data

All logic is handled at the backend level to ensure data consistency and correctness.

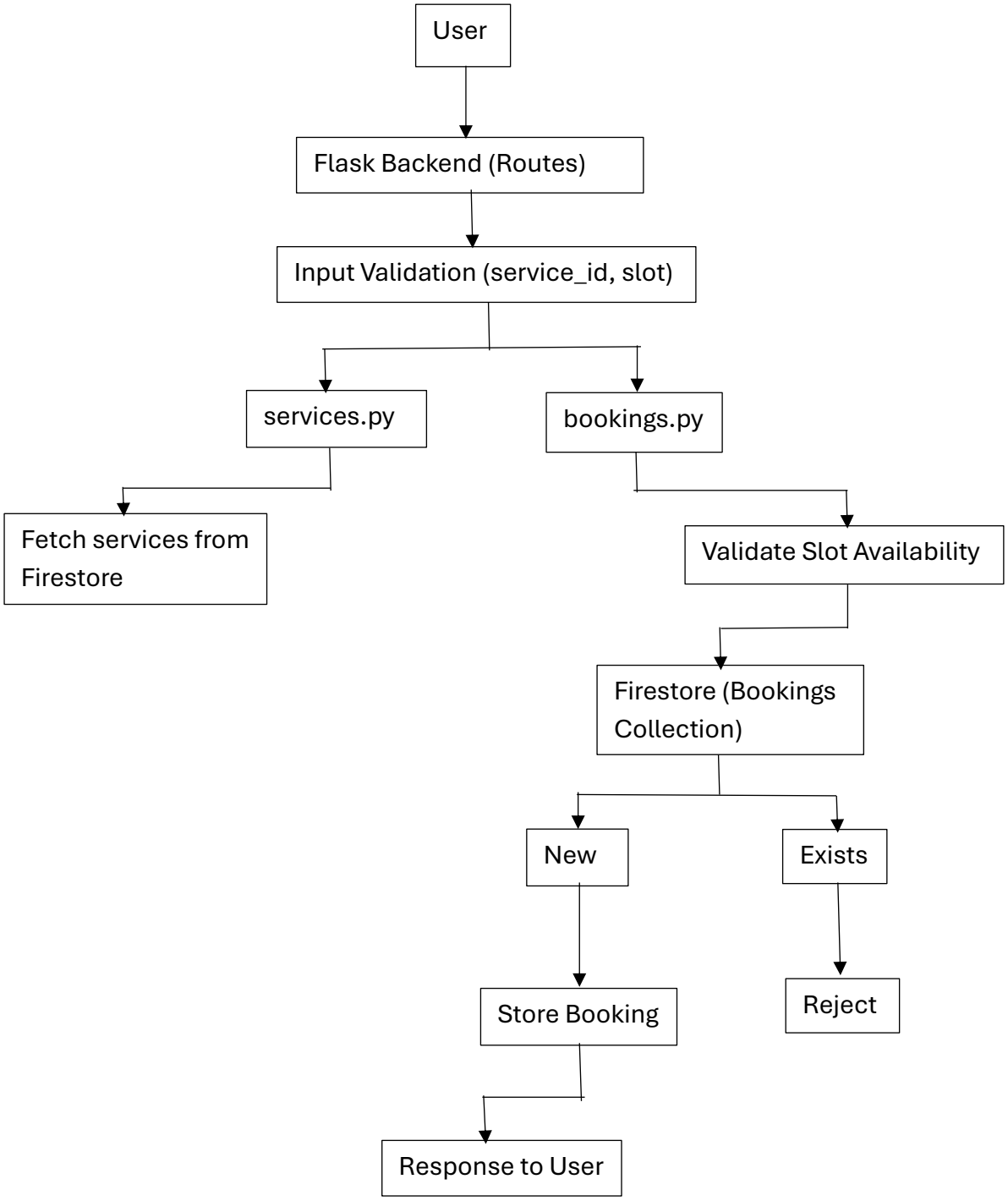
## Tools & Technologies Used

- Python
- Flask – REST API framework
- Google Firestore – NoSQL database
- Firebase Admin SDK – Secure server-side access to Firestore

## Service Booking System – Backend Workflow Flow Diagram

This figure illustrates the end-to-end backend request flow of the Service Booking System implemented using Flask and Google Firestore. The workflow begins with a user request received by the Flask backend, followed by input validation for service and slot details. Service-related requests are handled by the services.py route, which retrieves available services and time slots from Firestore. Booking requests are processed by the bookings.py route, where slot availability is validated against existing booking records in the Firestore bookings collection. If the selected slot is available, the booking is stored; otherwise, the request is rejected to prevent duplicate bookings. The system then returns an appropriate response to the user.

# Backend Workflow Diagram



## API Endpoints

The backend exposes RESTful endpoints to manage services and bookings. All endpoints are implemented using Flask and interact directly with Google Firestore.

- **GET /services**
  Returns the list of available services along with their available time slots stored in Firestore.
- **POST /book**
  Accepts booking details such as service ID and time slot. The backend validates slot availability and prevents duplicate bookings before storing the booking in Firestore.

## Firestore Data Model

The application uses Google Firestore as a NoSQL database with the following collections:

- **services**
  Stores service-related information.
  - service_id
  - service_name
  - available_slots
- **bookings**
  Stores confirmed bookings.
  - booking_id
  - service_id
  - slot
  - user_name
  - timestamp

This data model ensures efficient reads, clean writes, and reliable duplicate booking prevention.