# Insurance Policy Management System (IPMS)

**Objective**: Build an IPMS using Flask to showcase a full-stack development approach, from server-side logic with Flask to database interactions and RESTful API services, focused on insurance operations.

# Core Features:

1. **User Registration & Authentication**:
    - Secure user login/logout and registration processes using Flask-Login.
    - Utilize Flask-WTF for handling form inputs securely.
2. **Policy Management**:
    - Model definitions for various insurance policies (health, vehicle, property, etc.) using Flask-SQLAlchemy.
    - CRUD operations for managing policies.
3. **Customer Management**:
    - Interface for adding, updating, and deleting customer information.
    - Association of customers with their respective policies.
4. **Claim Processing**:
    - Mechanisms to file and track insurance claims.
    - Approval workflow for claims processing.
5. **RESTful API for Mobile/Web App**:
    - Design RESTful services for a mobile or web application interfacing with the IPMS.
    - Implement secure API endpoints for policy management, claim filing, and user authentication.
6. **Reporting and Analytics** (optional):
    - Generate reports on policy subscriptions, claims, and user activity.
    - Implement basic analytics for tracking insurance trends.

# Project Structure:

1. **Introduction to Flask & Flask Framework Overview**:
    - Initial project setup with Flask, understanding the application and request contexts.
2. **Flask Routing and Views**:
    - URL routing for customer and policy management views.
    - Dynamic routes for policy details and claim processing.
3. **Templates and Static Files**:
    - Use Jinja templates for rendering HTML pages.
    - Manage CSS and JavaScript for a better user interface.
4. **Flask Blueprint for Modular Applications**:

    - Organize the application into distinct components, such as authentication, API, and web interface modules.
5. **Flask and Databases**:
    - Database integration with Flask-SQLAlchemy for storing user, policy, and claim data.
    - Model definitions, migrations, and CRUD operations within the Flask app context.
6. **RESTful APIs with Flask**:
    - Develop RESTful services for external interactions with the insurance system.
    - Authenticate and authorize users and applications accessing the API.
7. **Form Handling with Flask-WTF**:
    - Securely manage forms for user and policy management interfaces.
8. **API Authentication and Authorization**:
    - Protect API routes using authentication mechanisms suitable for both web and mobile consumers.

# Deliverables:

- Source code repository for the IPMS Flask application.
- Documentation for the RESTful API and system architecture.
- A Postman collection for testing the API endpoints.
- A project report detailing the design, development process, challenges faced, and solutions implemented.