Here's the **1–13 section document** updated with your project title and team members included:

---

1. Introduction

• **Project Title**: Citizen AI: Intelligent Citizen Engagement Platform
• **Team Members**:

Arthi M.

Kamali K.

Arputha maha sridevi R.

Kavishree K.

Durgadevi N.

---

2. Project Overview

Purpose

The purpose of a Citizen AI: Intelligent Citizen Engagement Platform is to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging AI and real-time data, the assistant helps optimize essential resources like energy, water, and waste, while also guiding sustainable behaviors among citizens through personalized tips and services. For city officials, it serves as a decision-making partner—offering clear insights, forecasting tools, and summarizations of complex policies to support strategic planning. Ultimately, this assistant bridges technology, governance, and community engagement to foster greener cities that are more efficient, inclusive, and resilient.

Features

Conversational Interface – Natural language interaction for questions, updates, and guidance.

Policy Summarization – Converts lengthy government documents into concise, actionable summaries.

Resource Forecasting – Predicts energy, water, and waste usage using historical and real-time data.

Eco-Tip Generator – Personalized sustainability advice based on user behavior.

Citizen Feedback Loop – Collects and analyzes public input for city planning and services.

KPI Forecasting – Projects key performance indicators for strategic planning.

Anomaly Detection – Early warning for unusual sensor or usage data.

Multimodal Input Support – Accepts text, PDFs, and CSVs for analysis and forecasting.

Streamlit or Gradio UI – Intuitive dashboard for both citizens and city officials.

---

3. Architecture

Frontend (Streamlit) – Interactive web UI with dashboards, chat, uploads, feedback forms, and report viewers.

Backend (FastAPI) – REST framework powering document processing, chat, eco-tip generation, and vector embedding.

LLM Integration (IBM Watsonx Granite) – Natural-language understanding and generation for summaries, tips, and reports.

Vector Search (Pinecone) – Stores document embeddings for semantic search.

ML Modules – Forecasting and anomaly detection using scikit-learn and pandas/matplotlib.

---

4. Setup Instructions

Prerequisites

Python 3.9+

pip and virtual-environment tools

IBM Watsonx & Pinecone API keys

Internet access

Installation

Clone the repository.

Install dependencies from `requirements.txt`.

Create a `.env` file and configure credentials.

Run the FastAPI backend server.

Launch the Streamlit frontend.

Upload data and interact with the modules.

---

5. Folder Structure

`app/` – FastAPI backend logic (routers, models, integration modules)

`app/api/` – Modular API routes (chat, feedback, report, document vectorization)

`ui/` – Frontend components for Streamlit pages and forms

`smart_dashboard.py` – Entry script for the Streamlit dashboard

`granite_llm.py` – Handles communication with IBM Watsonx Granite model

`document_embedder.py` – Embeds documents and stores in Pinecone

`kpi_file_forecaster.py` – Forecasts future trends

`anomaly_file_checker.py` – Flags unusual KPI data

`report_generator.py` – Builds AI-generated sustainability reports

---

6. Running the Application

Launch the FastAPI server to expose backend endpoints.

Run the Streamlit dashboard to access the web interface.

Navigate via the sidebar.

Upload documents or CSVs, interact with the chat assistant, and view reports, summaries, and predictions in real time.

---

7. API Documentation

`POST /chat/ask` – AI-generated responses

`POST /upload-doc` – Upload & embed documents

`GET /search-docs` – Semantic document search

`GET /get-eco-tips` – Sustainability tips

`POST /submit-feedback` – Store citizen feedback

---

8. Authentication

Demo runs open, but secure deployments can include:

Token-based authentication (JWT/API keys)

OAuth2 with IBM Cloud credentials

Role-based access (admin, citizen, researcher)

Planned enhancements: user sessions & history tracking

---

9. User Interface

Minimalist, accessible design with:

Sidebar navigation

KPI visualizations & summary cards

Tabs for chat, eco tips, and forecasting

Real-time form handling

PDF report download capability

---

10. Testing

Unit Testing – Prompt functions & utilities

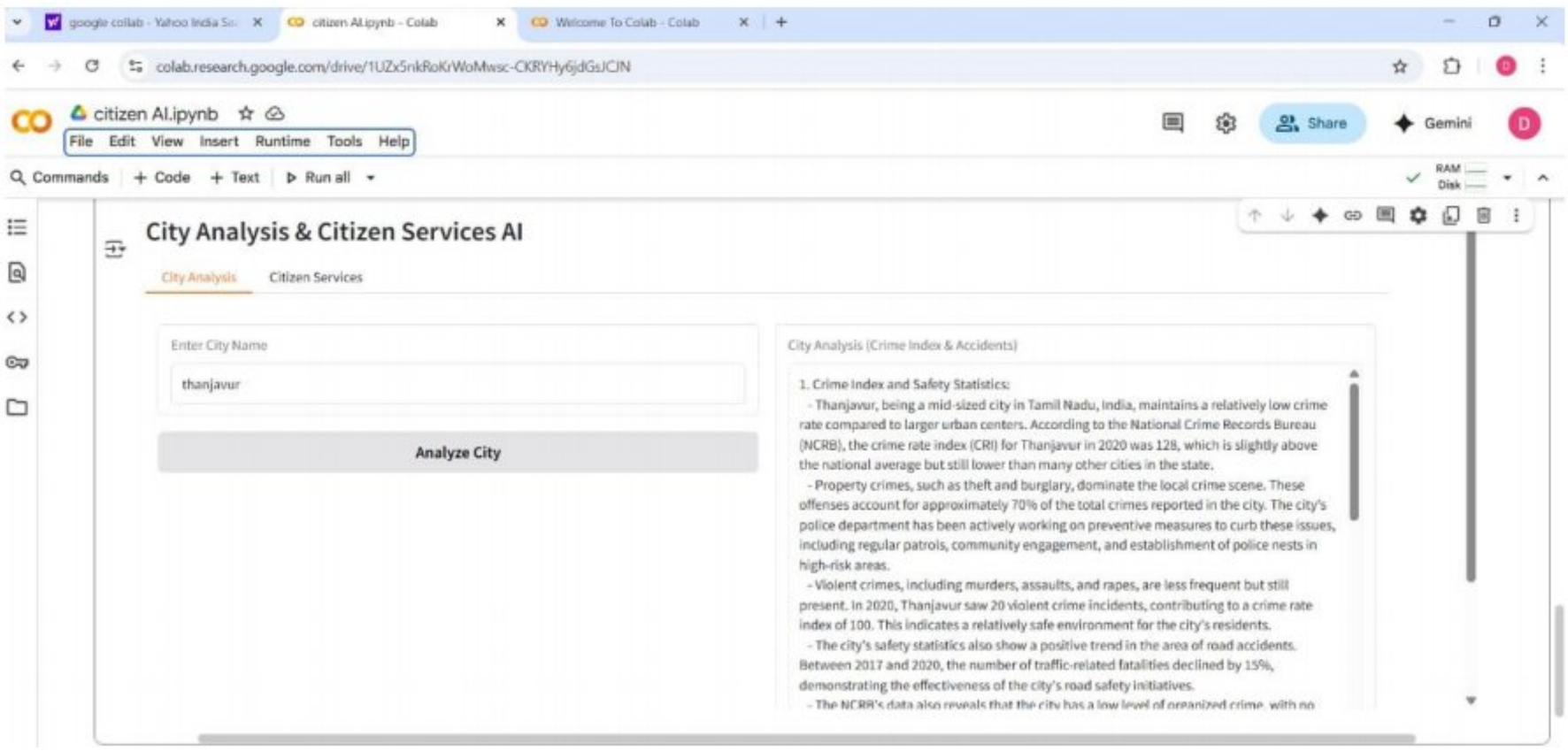
API Testing – Swagger UI, Postman, test scripts

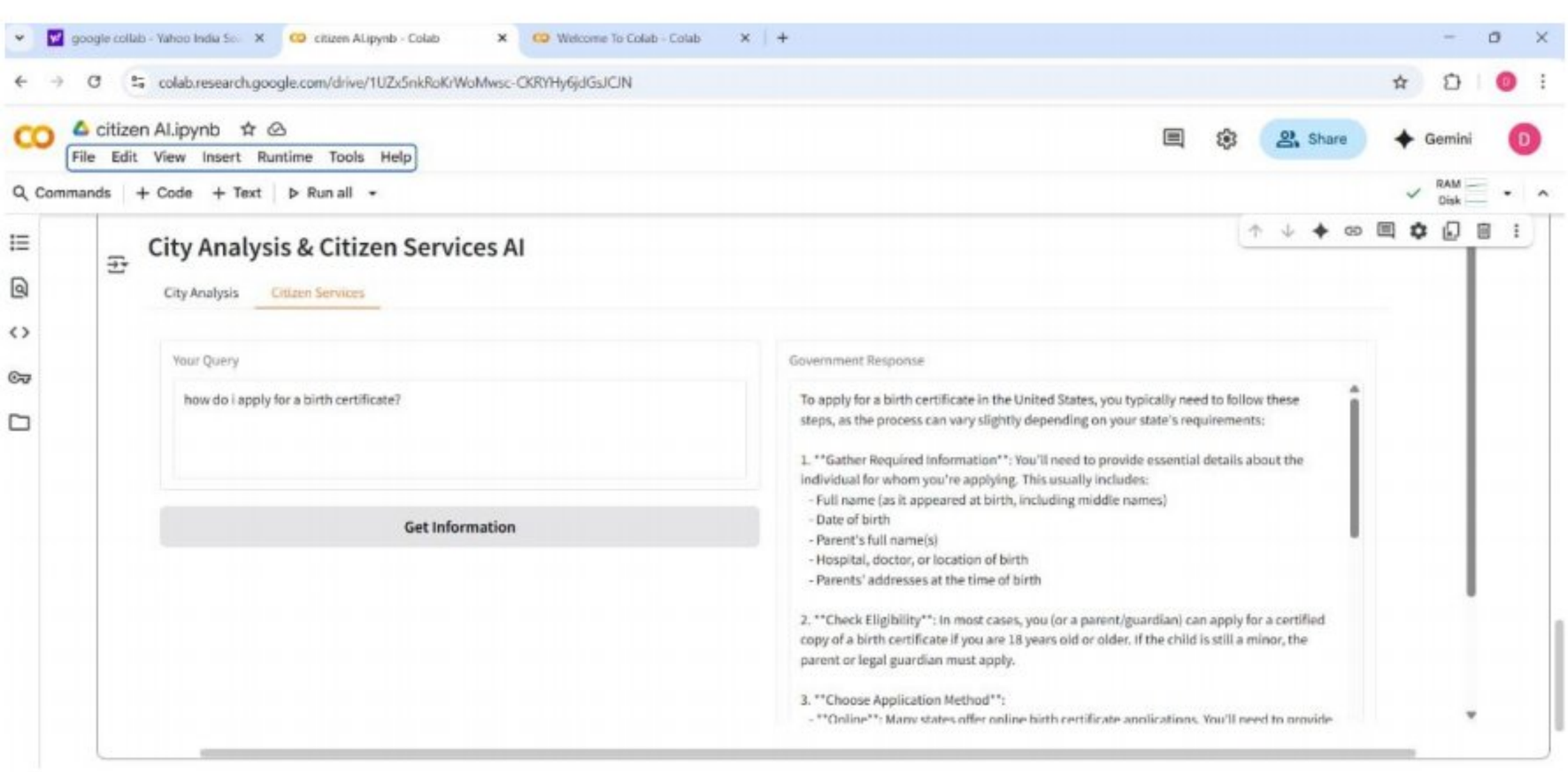
Manual Testing – File uploads, chat responses, output checks

Edge Case Handling – Malformed inputs, large files, invalid API keys

---

11. Screen Shots





---

12. Known Issues

Requires stable internet; limited offline use.

API rate limits may throttle heavy traffic.

Large file uploads can slow or exceed memory.

Demo lacks full authentication/security.

Forecasting accuracy depends on data quality.

Minor UI issues on mobile/older browsers.

Real-time IoT sensor integration not fully tested.

---

13. Future Enhancement

Advanced Security: Add full JWT/OAuth2 authentication and role-based access.

Mobile App: Develop responsive mobile/iOS/Android versions.

Real-Time IoT Integration: Direct streaming from city sensors for live data updates.

Multilingual Support: Expand to regional languages for wider citizen engagement.

Enhanced Analytics: Use deep learning for more accurate forecasting and anomaly detection.

Offline Mode: Limited functionality when internet is unavailable.

Automated Reporting: Scheduled AI-generated sustainability reports for officials.