# Project 1: Intelligent ChatBOT Application

## Tech Stacks:

- Programming: Python, Flask  
- Cloud Platform: Google Cloud Platform (GCP)  
- APIs/Services: Dialogflow CX, Cloud Run, Cloud Storage, Cloud Functions  
- Databases: Firestore, BigQuery  
- AI/ML Models: Gemini AI Integration, Word Embedding Models

## Roles and Responsibilities:

1. Dialogflow CX Setup:  
- Designed and configured a conversational flow for the chatbot, creating intents and entities for efficient user interaction.  
- Implemented context management to enable dynamic and personalized responses.  
  
2. Integration with AI Models:  
- Integrated the chatbot with Gemini AI for advanced query resolution using pretrained transformer models.  
- Utilized word embeddings for semantic similarity in user queries and responses.  
  
3. Document Upload and Parsing:  
- Enabled the chatbot to handle uploaded PDF documents, extracting key information and indexing it for user queries.  
- Implemented word embeddings for document search and retrieval.  
  
4. Backend Development:  
- Deployed the application using Docker and Cloud Run for serverless scalability.  
- Connected the chatbot with Firestore for storing user conversations and session data.  
  
5. Cloud Integrations:  
- Utilized Cloud Storage for secure file uploads.  
- Triggered Cloud Functions to process files and update Firestore with metadata.  
  
6. Testing and Monitoring:  
- Implemented logs and performance monitoring using Cloud Logging and Cloud Monitoring to track application usage.

## Outcomes:

- Successfully created a scalable and intelligent chatbot application capable of handling dynamic user queries.  
- Explored and implemented GCP services, showcasing their integration capabilities.  
- Demonstrated prototype functionality for financial use cases, document handling, and user interaction.

# Project 2: AI-Powered Document Retrieval System

## Tech Stacks:

- Programming: Python  
- Cloud Platform: Google Cloud Platform (GCP)  
- APIs/Services: Cloud Storage, Cloud Functions, Firestore  
- Libraries: TensorFlow, Gensim, SpaCy

## Roles and Responsibilities:

1. Document Upload and Processing:  
- Designed a pipeline for uploading PDF and Word documents to Cloud Storage.  
- Triggered Cloud Functions to parse documents and extract textual data.  
  
2. Embedding and Semantic Search:  
- Used pre-trained word embedding models (e.g., Word2Vec, GloVe) to create vector representations of document content.  
- Enabled semantic search capabilities by matching user queries with document embeddings.  
  
3. Database Integration:  
- Stored processed document embeddings and metadata in Firestore for efficient retrieval.  
  
4. Chatbot Integration:  
- Integrated with the chatbot to allow users to query document content dynamically.

## Outcomes:

- Automated the processing and retrieval of document data with high accuracy.  
- Provided users with quick and contextually relevant information from uploaded documents.

# Project 3: GCP Exploration and Automation

## Tech Stacks:

- Programming: Python  
- Cloud Platform: Google Cloud Platform (GCP)  
- Services Used: Cloud Storage, Cloud Logging, BigQuery, Cloud Scheduler

## Roles and Responsibilities:

1. Cloud Automation:  
- Scheduled jobs using Cloud Scheduler to automate data processing tasks.  
- Integrated with BigQuery for analytics on processed data.  
  
2. Performance Monitoring:  
- Set up Cloud Logging and Monitoring for real-time insights into application performance.  
  
3. Prototype Development:  
- Developed multiple prototypes to test the capabilities of GCP APIs.

## Outcomes:

- Gained hands-on experience with GCP's automation capabilities.  
- Showcased the use of serverless architecture for handling dynamic workloads.