Inception BD

111 no House, Domdom Sharok, Lahini Bot Tola, Kushtia-7001

Full Stack Data Science with Generative AI Bootcamp 1.0

Assignment 05

Last Submission Date: 28 December 2025, at 11:59 PM

📘 Assignment Submission Instructions:-

Step 1: Create a new repository on your GitHub account.
Step 2: Push all the code in the repo and share the link.
Step 3: Send the link via email to: learn.inceptionbd@gmail.com
Note: Make sure you record your project demo and share the video in Google Drive also attaches the drive link in the Readme.md file

Gmail Format:-

Compose a new email with the following details:

To: learn.inceptionbd@gmail.com
Subject: FSDS with Gen AI Assignment-5 Submission
Body:

Hello, please find my assignment submission at the following GitHub repository link.
Name: Md. Ridoy Hossain
Student ID: 2AE89DFB
Repository Link: **https://github.com/inceptionbd/Computer-Vision**

Best Regards
Md. Ridoy Hossain

🧩 Assignment Title

Personal AI Assistant (Gemini API + OOP + Streamlit)

🎯 Objective

Students will design and develop a **Personal AI assistant** capable of:

- Answering questions
- Helping with learning & productivity
- Acting like a smart digital assistant

The project must follow **Object-Oriented Programming (OOP)** principles and use **Gemini API** as the intelligence engine with a **Streamlit UI**.

🛠️ Tech Stack

- Python 3.10+
- Streamlit
- Google Gemini API
- python-dotenv
- OOP Architecture

🧠 OOP Concepts Required

✔ Classes & Objects
✔ Encapsulation
✔ Inheritance
✔ Modular Coding

🤖 JARVIS Core Capabilities

1. Greet the user like a personal assistant
2. Answer general questions using Gemini
3. Act as:
    - Tutor
    - Coding assistant
    - Career helper
4. Maintain conversation memory using a JSON file
5. Follow system-level instructions
6. Error handling & graceful fallback

📁 Mandatory Project Structure

```
jarvis_assistant/
│
├── app.py                  # Streamlit UI
│
├── jarvis/
│   ├── assistant.py        # JARVIS brain
│   ├── gemini_engine.py    # Gemini API handler
│   ├── prompt_controller.py  # System behavior & role
│   └── memory.py           # Conversation memory
│
├── config/
│   └── settings.py         # Environment & config
│
├── .env                    # API key
├── requirements.txt
└── README.md
```

🧱 Class Design (Optional, You can Change)

1️⃣ GeminiEngine Class (`gemini_engine.py`)

```
class GeminiEngine:
    def __init__(self, api_key):
        pass

    def generate(self, prompt):
        pass
```

Responsibilities

- Connect to Gemini API
- Send prompt
- Return AI response
- Handle API errors

2️⃣ PromptController Class (`prompt_controller.py`)

```
class PromptController:
    def build_prompt(self, user_input, memory):
        pass
```

Responsibilities

- Define Assistant personality
- Inject system instructions
- Format conversation context

3️⃣ Memory Class (`memory.py`)

```
class Memory:
    def add(self, role, message):
        pass

    def get_history(self):
        pass
```

Responsibilities

- Store conversation history in JSON file
- Load the previous conversation from JSON file and send to the model
- Return formatted context

4️⃣ JarvisAssistant Class (`assistant.py`)

```
class JarvisAssistant:
    def __init__(self, engine, prompt_controller, memory):
        pass

    def respond(self, user_input):
        pass
```

Responsibilities

- Control assistant workflow
- Coordinate memory → prompt → Gemini → response

5️⃣ Settings Class (`settings.py`)

```
class Settings:
    def load_api_key(self):
        pass
```

Responsibilities

- Securely load environment variables

🖥️ Streamlit UI Requirements(Optional, You can change) (`app.py`)

Students must implement:

- JARVIS-themed UI
- Chat-style interface
- Sidebar controls:
  - Assistant role (Tutor / Coder / Mentor)
  - Clear memory
- Chat history display

Example:

```
st.title("🧠 JARVIS – Your AI Assistant")
user_input = st.chat_input("Ask JARVIS...")
```

🔐 Security & Best Practices

✔ API key must be stored in `.env`
✔ `.env` must be ignored from GitHub
✔ No Gemini logic inside `app.py`

⭐ Bonus Features (Optional but Powerful)

- Voice input (speech-to-text)
- Streaming responses
- Chat export
