“Initially, I didn’t have any prior knowledge of Microsoft Azure, so I started by exploring and learning the basics of the platform — like how to create resource groups, understand Azure AI services, and manage deployments. My teammate helped me get started with the fundamentals, and from there, I began implementing smaller components to gain practical experience.

As part of my learning, I explored the Azure AI Foundry and conducted a detailed case study. I learned how to create, fine-tune, and deploy models there. I also experimented with NLP-based models to understand how natural language queries are processed and how semantic search works. Later, I used these deployed models and their endpoints in our project implementation.

I also explored Azure Cognitive Search in depth — learning how to create data sources, indexes, and indexers, and how to connect them with Azure Blob Storage. I understood how search indexes help in improving information retrieval and how cognitive search can be combined with AI to deliver semantic results.

Once I got a clear understanding of both the AI and search components, I started focusing on the frontend. My key responsibility in the project was developing the user interface using Angular. I designed and built the prompt window that interacts with the backend APIs. I made sure that user inputs are processed correctly, and the results from the Azure AI models are displayed seamlessly on the UI.

I also learned how to integrate the Azure services using the provided API endpoints and keys, ensuring secure communication between the frontend and backend. After integrating, I worked with my teammates to test the complete setup — connecting the frontend UI, backend services, and Azure Cognitive Search.

Finally, I verified that our deployed UI on Azure Web App was functional and connected correctly with the cognitive search and OpenAI services. So overall, my contribution involved exploring Azure from scratch, performing a case study in AI Foundry, deploying and using AI models, understanding and implementing cognitive search, and developing and integrating the frontend UI with the backend services to make the project fully functional.”

**Week 1 – Exploration and Understanding**

* **Objective:** Understand the Azure platform and project requirements from scratch.
* **What I personally did:**
  + Started with **zero prior experience in Azure** and spent significant time exploring the portal, AI Studio, AI Foundry, and Cognitive Search.
  + Learned **how Azure services are structured** — resource groups, AI services, web app hosting, and endpoints.
  + Understood **the workflow of AI models** and how they can be integrated with a frontend application.
  + Discussed and clarified with teammates about **project goals, AI services to use, and backend requirements**.
  + Began experimenting with **small hands-on examples**, like calling Azure OpenAI models from sample scripts and trying semantic search queries.
* **Outcome:** Built a strong foundation, which enabled me to take ownership of **integration and frontend tasks later**.

**Week 2 – Case Study, AI Model Deployment, and Backend Integration**

* **Objective:** Deploy AI models, understand endpoints, and explore Azure Cognitive Search.
* **What I personally did:**
  + Conducted a **detailed case study in AI Foundry**, exploring **NLP, semantic search, and AI model capabilities**.
  + **Deployed AI models** in Azure AI Foundry for real use. I generated and **shared endpoints and keys** with my teammates for integration.
  + Explored **Azure Cognitive Search**:
    - Created **data sources** from Azure Blob Storage.
    - Defined **indexes and indexers** to process data and enable search functionality.
    - Tested **search queries**, ensuring semantic results were returned accurately.
  + Built a **small demo application** to showcase how **user queries flow through cognitive search and AI models**, validating end-to-end functionality.
  + Shared **all deployment details, endpoints, and keys** with the team and **coordinated integration** with backend services.
* **Outcome:** I became the main contributor in **model deployment, endpoint configuration, and search integration**, laying the groundwork for the frontend to connect seamlessly.

**Week 3 – Frontend Development, Prompt Generation, and Full Integration**

* **Objective:** Build the frontend UI, integrate it with backend AI services, and ensure a professional user experience.
* **What I personally did:**
  + Took full responsibility for **frontend development using Angular**.
  + Designed and implemented the **prompt window UI**, ensuring it is user-friendly, visually aligned, and responsive.
  + Integrated the frontend **with Azure AI services**:
    - Connected the prompt input to the **deployed OpenAI models** using endpoints and keys.
    - Connected the **search feature to Cognitive Search** indexes for semantic query results.
  + Tested **end-to-end functionality**, ensuring that:
    - User inputs trigger correct AI responses.
    - Search results are retrieved accurately.
    - UI displays all results seamlessly.
  + Deployed the **web application on Azure** and validated that the frontend interacts correctly with backend services in the live environment.
  + Coordinated with teammates to integrate **all components**, ensuring that the workflow — from user input to AI model response — works flawlessly.
  + Refined the **prompt flow and frontend UI**, improving look, responsiveness, and professional alignment.
* **Outcome:** I own the **frontend and prompt generation module**, ensuring the interface works perfectly with the deployed AI models and cognitive search, and is ready for deployment and demonstration.

**Overall Contribution**

* Explored and learned **Azure platform from scratch**.
* Conducted **AI Foundry case study**, deployed AI models, and generated **endpoints and keys** for integration.
* Configured **Azure Cognitive Search** — data sources, indexes, and indexers — and validated search results.
* Developed **full frontend UI**, including **prompt generation module**, integrated seamlessly with backend AI services.
* Deployed the **web application on Azure**, tested end-to-end flow, and coordinated with the team for final integration.
* Ensured the **UI is professional, responsive, and fully functional**, ready for demonstration to senior management.

✅ This version clearly shows **all your personal contributions**, emphasizing:

* Learning from scratch.
* Case study and model deployment.
* Cognitive search exploration and integration.
* Frontend UI development, prompt generation, and full integration.
* Azure web app deployment and end-to-end testing.

**Project Flow Explanation (Verbal Version)**

“The flow of our project starts with the user interacting with the frontend. The user enters a query or prompt in the prompt window that I designed. This input is then sent securely to our backend services using the API endpoints and keys of the deployed AI models.

The backend first checks the query and, if necessary, interacts with Azure Cognitive Search, which searches through our indexed data to retrieve relevant information. The results from Cognitive Search are combined with the AI model’s processing. The AI model generates a semantic response or processed output, which is sent back to the frontend.

On the frontend, the prompt window receives the response and displays it to the user in a clear, professional, and responsive way. All of this happens seamlessly, so the user experiences a smooth interaction, as if the AI is understanding and answering their query in real-time.

So in short: user input → frontend prompt → backend processing → Cognitive Search and AI model → combined response → frontend display. I’m responsible for the frontend part and prompt integration, making sure the user experience is smooth and the responses display correctly.”

 **Driver Monitoring (Data Acquisition):**

* A **camera inside the vehicle** continuously records the driver’s face.
* The system focuses on the **eyes** to detect signs of drowsiness.

 **Image Processing (Python + Computer Vision):**

* Using **Python and OpenCV**, the system analyzes the **Eye Aspect Ratio (EAR)**.
* EAR measures **how open or closed the eyes are**.
* If EAR falls below a threshold for a certain time, it indicates the driver is getting drowsy.

 **Decision Trigger:**

* When drowsiness is detected, the **Python program sends a command via Bluetooth** to the Arduino controlling the vehicle.

 **Vehicle Response (Arduino 2 – Robotic Vehicle):**

* The robotic vehicle first makes a **controlled left turn** to grab the driver’s attention.
* Then it **stops completely** to prevent accidents.
* The **red LED** on the vehicle lights up as a **visual alert**, and the **buzzer** sounds as an **audio alert**.

 **Communication & Alerts (Arduino 1 + GSM):**

* Arduino 1 acts as a **bridge between Python and the robotic vehicle**.
* It displays alerts on an **LCD screen** and manages the **GSM module**.
* A **real-time SMS notification** is sent to the system administrator, including the **location of the vehicle**.

 **Parking & Safety:**

* After stopping, the vehicle can **park safely** using **ultrasonic sensors** to avoid collisions.
* Alerts continue until the driver is fully attentive.

 **Summary of Safety Layers:**

* **Visual Alert:** Red LED
* **Audio Alert:** Buzzer
* **Automated Vehicle Stop:** Prevents accident
* **Admin Notification:** Ensures secondary intervention
* *"During my UG, I worked on my project* ***DriveSense****, which focuses on automated driver drowsiness detection and accident prevention. Building on this work, I contributed to a research paper that was published in the* ***International Conference on Cognitive Computing and AI (ICCCAI 2024)****, associated with Taylor University. The paper was also included in the* ***AIP Conference Proceedings****, which are recognized and indexed publications for conference research. This experience helped me gain exposure to* ***research, technical writing, and presenting my work in an international forum****, in addition to the hands-on technical skills I developed while building the project."*