**A SHORT TERM PROJECT REPORT**

**ON**

TRAVEL\_AMAZE

**Submitted in the Partial fulfillment of the requirements for The Award of the Degree of**

**BACHELOR OF SCIENCE**

**IN**

**COMPUTER SCIECE**

**BY**

# SHAIK HAFIZA(k2221502)

**Under the Guidance of**

 **G. KRISHNA**

**KAKARAPARTI BHAVANARAYANA COLLEGE (AUTONOMOUS)**

**(Sponsored by S.K.P.V.V. Hindu High School Committee) Kothapeta, Vijayawada**

**2023-24**

**KAKARAPARTI BHAVANARAYANA COLLEGE (AUTONOMOUS)**

**(Sponsored by S.K.P.V.V. Hindu High School Committee)**

**Kothapeta, Vijayawada.**

**Department of Computer Science & Application**



# CERTIFICATE

This is to certify that this work entitled "**TRAVEL\_AMAZE**" being submitted by **G. KRISHNA** bearing **REGD NO -K2221502** in the partial fulfillment for the award of the degree in **BACHELOR OF COMPUTER SCIENCE** of **KRISHNA UNIVERSITY, MACHILIPATNAM** during the Academic year 2022-2023.It is certify that all corrections/ suggestions indicated for internal assessment have been incorporated in the report. The project work has been approved as it satisfies the academic requirements in respect of project work prescribed for the above degree.

**PROJECT GUIDE HEAD OF The DEPARTMENT**

**G.KRISHNA P. Ravindra**

30-05-2024

Page No:

Sk.Hafiza

Sk.Hafiza

**PROGRAM**

**BOOK**

**FOR**

**SHORT**

**-**

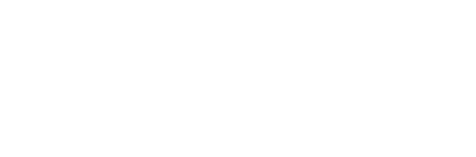
**TERM**

**INTERNSHIP**

**Onsite /**

**(**

**Virtual)**



**KRISHNA**

**University**



05-07-2024

Fresher Bot Technology

**K2221502**

**Kakaraparti Bhavanarayana College**

**Sk.Hafiza**

**DECLARATION**

The Project work entitled **“TRAVEL\_AMAZE** “is a record of bonafide work carried out by me, submitted in partial fulfillment for the award of **BACHELOR OF DEGREE** in **COMPUTER SCIENCE** to the **KRISHNA UNIVERSITY, MACHILIPATNAM.** The result embodied in this project work has not been submitted to any other University or Institute for the award of any degree or diploma.

**NAME OF THE STUDENT SIGNATURE THE STUDENT**

**SHAIK HAFIZA (K22215202)**

**ACKNOWLEDGEMENT**

This project work is a golden opportunity for learning and self-development. We consider our self very lucky to have so many people lead us through in completion of this project.

We express our sincere thanks to the management and our beloved **principal DR. V. NARAYANA RAO** for providing such wonderful facilities required encouragement in completion of this entire project work.

We place an record, our sincere gratitude to sri**. P. RAVINDRA**, **HOD OF COMPUTER APPLICATIONS** for his constant encouragement. He monitored the progress and arranged all facilitates to make this project easier. We owe our profound gratitude to our project guide **G. KRISHNA** who took keen interest on this project work and guided us all along and whose patience we have probably used to the limit. He was always so involved in the entire process, shared his knowledge, and Encouraged to think.

We would like to thank all the other faculty members, technical staff and supporting staff who have provided their contribution in the completion this project work.

# An Internship Report on

**Prompt-engineering**

*Submitted in accordance with the requirement for the degree of*

**B.Sc DATA SCIENCE**

*Under the Faculty Guideship of*

**G. KRISHNA**

*Department of*

COMUTER SCIENE AND APPLICATIONS

Kakaraparti Bhavanarayana College (Autonomous)

Submitted by:

***SHAIK HAFIZA***

Reg.No:

**K2221502**

*Department of*

**COMUTER SCIENE AND APPLICATIONS**

**Kakaraparti Bhavanarayana College (Autonomous)**

**Instructions to Students**

Please read the detailed Guidelines on Internship hosted on the website of AP State Council of Higher Education **https://apsche.ap.gov.in**

1. It is mandatory for all the students to complete 2 months (180 hours) of short- term internship either physically or virtually.
2. Every student should identify the organization for internship in consultation with the College Principal/the authorized person nominated by the Principal.
3. Report to the intern organization as per the schedule given by the College. You must make your own arrangements for transportation to reach the organization.
4. You should maintain punctuality in attending the internship. Daily attendance is compulsory.
5. You are expected to learn about the organization, policies, procedures, and processes by interacting with the people working in the organization and by consulting the supervisor attached to the interns.
6. While you are attending the internship, follow the rules and regulations of the intern organization.
7. While in the intern organization, always wear your College Identity Card.
8. If your College has a prescribed dress as uniform, wear the uniform daily, as you attend to your assigned duties.
9. You will be assigned a Faculty Guide from your College. He/She will be creating a WhatsApp group with your fellow interns. Post your daily activity done and/or any difficulty you encounter during the internship.
10. Identify five or more learning objectives in consultation with your Faculty Guide. These learning objectives can address:
    1. Data and Information you are expected to collect about the organization and/or industry.
    2. Job Skills you are expected to acquire.
    3. Development of professional competencies that lead to future career success.
11. Practice professional communication skills with team members, co-interns, and your supervisor. This includes expressing thoughts and ideas effectively through oral, written, and non-verbal communication, and utilizing listening skills.
12. Be aware of the communication culture in your work environment. Follow up and communicate regularly with your supervisor to provide updates on your progress with work assignments.
13. Never be hesitant to ask questions to make sure you fully understand what you need to do your work and to contribute to the organization.
14. Be regular in filling up your Program Book. It shall be filled up in your own handwriting. Add additional sheets wherever necessary.
15. At the end of internship, you shall be evaluated by your Supervisor of the intern organization.
16. There shall also be evaluation at the end of the internship by the Faculty Guide and the Principal.
17. Do not meddle with the instruments/equipment you work with.
18. Ensure that you do not cause any disturbance to the regular activities of the intern organization.
19. Be cordial but not too intimate with the employees of the intern organization and your fellow interns.
20. You should understand that during the internship programme, you are the ambassador of your College, and your behavior during the internship programme is of utmost importance.
21. If you are involved in any discipline related issues, you will be withdrawn from the internship programme immediately and disciplinary action shall be initiated.
22. Do not forget to keep up your family pride and prestige of your College.

--------<<@>>-------

# Student’s Declaration

I, Sk. Hafiza a student of Prompt Engineering Program, Reg. No. k2221502

of the Department of Computer Science, KBN College do hereby declare that

I have completed the mandatory internship from 30-05-2024 to 05-07-2024 in

**Fresher Bot Technology** under the Faculty Guideship of G. Krishna, Department of Computer Science, K.B.N College

*(Signature and Date)*

## Official Certification

This is to certify that Sk. Hafiza Reg. No. K2221502 has completed his/her Internship in **Fresher Bot Technology***(Name of the Intern Organization)* on Prompt Engineeringunder my supervision as a part of partial fulfillment of the requirement for the Degree of BSc(Data Science) in the Department of

Computer Science, KBN College*.*

This is accepted for evaluation.

*(Signatory with Date and Seal)*

### Endorsements

*Faculty Guide*

*Head of the Department*

*Principal*

## Certificate from Intern Organization

This is to certify that Sk. Hafiza *(Name of the intern)* Reg. No k2221502 of

K. B.N. College*(Name of the College)* underwent internship in **Fresher Bot Technology** *(Name of the Intern Organization)* from **30-05-2024** to **5-07-2024**

The overall performance of the intern during his/her internship is found to be \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Satisfactory/Not Satisfactory).

*Authorized Signatory with Date and Seal*

## CHAPTER 1: EXECUTIVE SUMMARY

#### The internship report shall have only a one-page executive summary. It shall include five or more Learning Objectives and Outcomes achieved, a brief description of the sector of business and intern organization and summary of all the activities done by the intern during the period

**Learning Objectives and Outcomes Achieved**

**Learning Objectives:**

1. **Master Prompt Engineering:** Develop expertise in creating effective and efficient prompts for AI models.
2. **Understand AI Model Behaviour:** Gain insights into how AI models interpret and respond to various prompts.
3. **Enhance Problem-Solving Skills:** Improve the ability to troubleshoot and refine prompts to achieve desired outcomes.
4. **Develop Communication Skills:** Learn to articulate prompt strategies and results to diverse stakeholders.
5. **Apply AI in Real-World Scenarios:** Utilize prompt engineering to solve practical problems in various business sectors.

**Outcomes Achieved:**

1. **Proficiency in Prompt Engineering:** Acquired advanced skills in designing and testing prompts for AI models.
2. **Improved AI Model Performance:** Enhanced the performance and accuracy of AI models through optimized prompts.
3. **Problem-Solving Expertise:** Successfully identified and resolved issues related to prompt formulation and model responses.
4. **Effective Communication:** Demonstrated the ability to explain complex AI concepts and prompt strategies clearly.
5. **Real-World Applications:** Applied prompt engineering techniques to address real-world challenges in the business sector.

## CHAPTER 2: OVERVIEW OF THE ORGANISATION

###### Suggestive contents

1. Introduction of the Organization
2. Vision, Mission, and Values of the Organization
3. Policy of the Organization, in relation to the intern role
4. Organizational Structure
5. Roles and responsibilities of the employees in which the intern is placed.
6. Performance of the Organization in terms of turnover, profits, market reach and market value.
7. Future Plans of the Organization.

**A. Introduction of the Organization**

TRAVEL\_AMAZE is an innovative tourism organization dedicated to revolutionizing the travel experience through cutting-edge chatbot technology. Our mission is to provide personalized, engaging, and seamless travel services that cater to the diverse needs of modern travelers. By leveraging artificial intelligence and advanced machine learning, we aim to make every journey memorable and hassle-free.

**B. Vision, Mission, and Values of the Organization**

**Vision:**

We envision a world where travel is accessible, enjoyable, and sustainable for everyone. By integrating technology with a deep understanding of travel trends and customer preferences, we aim to create a global community of satisfied and inspired traveller.

**Mission:**

Our mission is to enhance the travel experience by offering innovative, reliable, and user-friendly services that meet the evolving needs of our customers. We strive to be a leader in the tourism industry by promoting sustainable travel practices and supporting local communities**.**

**C. Policy of the Organization in Relation to the Intern Role**

**Internship Policy:**

* **Innovation:** Continuously seek creative solutions and embrace new technologies.
* **Integrity:** Maintain transparency, ethical practices, and commitment to high standards.
* **Excellence:** Strive for excellence in every project, focusing on quality and efficiency.
* **Collaboration:** Foster a collaborative environment that encourages teamwork and diverse perspectives.
* **Customer-Centricity:** Prioritize customer needs and aim to exceed their expectations.

**Intern Responsibilities:**

Interns are expected to actively participate in assigned tasks, contribute to team projects, and adhere to company policies and deadlines.

**D. Organizational Structure**

**Organizational Chart:**

1. CTO (Chief Technology Officer)

* Leads technology strategy and oversees chatbot development.
* Lead Chatbot Developer.
* Heads the development and maintenance of the chatbot.

2. AI/ML Engineers

* Develop and enhance AI and machine learning models for the chatbot.

3. Backend Developers

* Handle server-side logic, database management, and system integration.

4. Frontend Developers

* Design and implement the user interface, integrating the chatbot on web and mobile platforms.

5. Sales and Partnerships

a)\_ Sales Director

o Leads sales strategy and manages the sales team.

b) Sales Representatives

* + Sell tourism packages and build client relationships.
  + Customer Support

c) Customer Support Manager

o Manages the customer support team and ensures high-quality service.

d) Support Agents

o Assist users, handle inquiries, and resolve issues through various channels.

**CHAPTER 3: INTERNSHIP PART**

*Description of the Activities/Responsibilities in the Intern Organization during Internship, which shall include - details of working conditions, weekly work schedule, equipment used, and tasks performed. This part could end by reflecting on what kind of skills the intern acquired.*

Description of Activities and Responsibilities During the Internship

During the internship atAI Innovators Inc, the working conditions were designed to provide a professional and supportive environment conducive to learning and growth. The conditions included:

Work Environment: The internship took place in a modern office space equipped with comfortable workstations, high-speed internet, and collaborative spaces for team discussions and meetings.

Remote Work Option: Given the flexibility of the internship program, there was an option for remote work for certain tasks, allowing interns to balance their schedule and work from home as needed.

Support and Guidance: Interns received regular support and guidance from experienced professionals, including one-on-one mentoring sessions and team meetings to discuss progress and challenges.

**ACTIVITY LOG FOR THE FIRST WEEK**

|  |  |  |  |
| --- | --- | --- | --- |
| **Day & Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person In- Charge Signature** |
| Day – 1  (30-05-2024) | * Discussing about the prompt engineering * Basics of prompt engineering | * How to give prompts |  |
| Day-2  (31-05-2024) | * Components of good prompt * Application of prompt engineering | * Various applications on prompts |  |
| Day-3  (01-06-2024) | * Introduction to generative AI | * LLM’s and its importance |  |
| Day-4  (03-06-2024) | * Prompting errors and limitations * Advanced prompts and it’s hands on | Limitations of prompt |  |
| Day-5  (04-06-2024) | * What is Tailwind CSS? | * Information about tailwind.css |  |
| Day-6  (05-06-2024) | * How does tailwind CSS differ from other CSS framework like bootstrap? | * Usage of tailwind.css |  |

**WEEKLY REPORT**

**WEEK – 1 (From Dt : 30-05-2024 to Dt: 05-06-2024)**

**Objective of the Activity Done:**

**Detailed Report:**

**Prompt Engineering:**

Discussing the prompt of engineering involves exploring various aspects related to the field of engineering itself, or specific prompts related to engineering topics. Here are some potential directions to discuss:

1. **Definition and Scope**: Define what engineering encompasses — the application of scientific principles to design and create structures, machines, systems, and processes to solve real-world problems.
2. **Branches of Engineering**: Discuss the different branches such as civil, mechanical, electrical, chemical, etc., each specializing in different areas of application and technology.
3. **Importance**: Highlight the crucial role engineering plays in shaping modern society, from infrastructure to technology advancements, healthcare innovations, environmental sustainability, and beyond.
4. **Challenges**: Explore the challenges faced by engineers, including technological, ethical, environmental, and societal challenges, and how engineers strive to overcome them.
5. **Innovation and Future Trends**: Discuss recent innovations in engineering and emerging trends such as AI, renewable energy, smart cities, and biotechnology, and their potential impacts on society.
6. **Education and Career**: Talk about the educational paths to becoming an engineer, career prospects, and the skills required in the modern engineering landscape.
7. **Ethics and Responsibility**: Address the ethical considerations in engineering, including issues like sustainability, safety, privacy, and the ethical use of technology.
8. **Global Perspective**: Consider engineering from a global standpoint, discussing how different regions approach engineering challenges and collaborate on global projects.
9. **Interdisciplinary Nature**: Highlight how engineering often intersects with other disciplines like medicine, business, and the arts, fostering innovation through interdisciplinary collaboration.

**Public Perception**: Discuss public perceptions of engineering and how these perceptions influence the field's development and societal impact.

**Basics of prompt engineering:**

Prompt engineering refers to the process of crafting and refining prompts for artificial intelligence (AI) models, particularly language models like GPT (Generative Pre-trained Transformer). Here are the basics of prompt engineering explained:

### 1. **Understanding Prompts in AI:**

* **Definition**: A prompt is a textual input or query provided to an AI model to generate a response or perform a task.
* **Role**: It shapes the context and direction of the AI's output, influencing what the model generates.

### 2. **Components of Prompt Engineering:**

* **Language Choice**: The language used in the prompt should be clear and unambiguous to convey the desired task or query.
* **Context Setting**: Provide relevant context that helps the model understand the intended meaning and scope of the task.
* **Examples and Formats**: Using examples or templates can guide the model towards producing desired types of responses.

### 3. **Techniques for Effective Prompt Engineering:**

* **Precision**: Craft prompts that are specific and focused on the desired outcome, avoiding ambiguity.
* **Clarity**: Use clear language and avoid complex structures or unnecessary details that could confuse the model.
* **Iterative Refinement**: Test and refine prompts based on the model's responses, adjusting for better performance.

### 4. **Types of Prompts:**

* **Question-Answering**: Structuring prompts as questions to receive specific answers.
* **Text Generation**: Guiding the model to generate coherent and relevant text based on provided input.
* **Conditional Outputs**: Using prompts to condition the model's responses on specific attributes or constraints.

### 5. **Considerations in Prompt Engineering:**

* **Bias and Fairness**: Ensure prompts are designed to mitigate bias and promote fairness in AI outputs.
* **Evaluation**: Regularly evaluate the effectiveness of prompts in achieving desired outcomes.
* **Domain Specificity**: Tailor prompts to specific domains or tasks to improve model performance in those areas.

### 6. **Tools and Resources:**

* **Prompt Libraries**: Utilize existing prompt libraries or templates to expedite prompt engineering.
* **Feedback Mechanisms**: Incorporate feedback loops to refine prompts based on model performance and user input.

### 7. **Applications and Impact:**

* **Enhanced User Experience**: Well-engineered prompts can improve user interaction with AI systems.
* **Performance Optimization**: Optimal prompts can boost AI model accuracy and efficiency in generating outputs.

### 8. **Future Directions:**

* **Automation**: Develop automated tools or algorithms to assist in prompt generation and refinement.
* **Ethical Considerations**: Address ethical implications of prompt engineering in AI, including privacy and bias concerns.

**Components of good prompt:**

Creating a good prompt is essential for effectively guiding an AI model to produce accurate and relevant outputs. Here are the key components of a good prompt:

### 1. **Clarity and Precision:**

* **Clear Language:** Use straightforward and unambiguous language that clearly communicates the task or query to the AI model.
* **Specificity:** Be precise about what information or output you expect from the model, avoiding vague or overly general prompts.

### 2. **Contextual Relevance:**

* **Relevant Context:** Provide sufficient context within the prompt to help the AI model understand the intent and scope of the task.
* **Background Information:** Include relevant background details that may assist the model in generating a more accurate response.

### 3. **Formatting and Structure:**

* **Consistent Format:** Use a consistent format or structure for prompts, especially in repetitive tasks, to maintain clarity and ease of understanding.
* **Logical Flow:** Organize the information in a logical sequence that guides the model towards the desired output.

### 4. **Examples and Templates:**

* **Concrete Examples:** Include examples within the prompt to illustrate the expected type of response or output from the AI model.
* **Templates:** Utilize predefined templates or formats that align with the specific task or application to streamline prompt creation.

### 5. **Specific Instructions or Constraints:**

* **Instructional Clauses:** Incorporate clear instructions or constraints within the prompt to guide the AI model's behavior or output.
* **Conditional Statements:** Use conditional statements to define specific conditions or criteria that the model should consider in generating its response.

### 6. **Bias Mitigation and Fairness:**

* **Avoiding Bias:** Ensure that prompts are designed to mitigate biases and promote fairness in AI outputs, especially in sensitive or ethical contexts.
* **Diverse Perspectives:** Consider diverse perspectives and potential interpretations of the prompt to minimize unintended biases.

### 7. **Evaluation and Iteration:**

* **Performance Metrics:** Establish criteria or metrics for evaluating the effectiveness of prompts in achieving desired outcomes.
* **Iterative Refinement:** Continuously refine prompts based on feedback, model performance, and evolving requirements to optimize results.

### 8. **User-Centric Design:**

* **User Experience:** Consider the perspective of end-users who interact with the AI model, ensuring that prompts are user-friendly and intuitive.
* **Feedback Mechanisms:** Incorporate mechanisms for collecting user feedback on prompt effectiveness to inform future iterations.

### 9. **Domain-Specific Considerations:**

* **Domain Expertise:** Tailor prompts to specific domains or industries, incorporating domain-specific terminology and knowledge to enhance model understanding and performance.
* **Task Complexity:** Adjust the complexity of prompts based on the complexity of the task or problem being addressed by the AI model.

### 10. **Ethical and Legal Compliance:**

* **Ethical Guidelines:** Adhere to ethical guidelines and legal regulations when designing prompts, particularly in sensitive or regulated industries.

**Application of prompt engineering:**

Prompt engineering finds application across various domains where AI models are used for tasks such as natural language processing, content generation, decision support, and more. Here are some specific applications of prompt engineering:

### 1. **Natural Language Understanding (NLU):**

* **Question Answering:** Designing prompts as questions to extract specific information from a large corpus of text or data.
* **Information Retrieval:** Crafting prompts to retrieve relevant information from databases or knowledge bases.
* **Semantic Parsing:** Creating prompts that parse and interpret complex queries to generate structured outputs.

### 2. **Text Generation and Summarization:**

* **Content Creation:** Developing prompts to guide AI models in generating coherent and informative text for articles, reports, or creative writing.
* **Summarization:** Constructing prompts to instruct models in summarizing lengthy documents or articles into concise and meaningful summaries.

### 3. **Conversational AI:**

* **Chatbots and Virtual Assistants:** Designing prompts to facilitate natural and engaging conversations with users, addressing queries and providing relevant responses.
* **Dialogue Systems:** Crafting prompts that enable AI models to maintain context and coherence in multi-turn conversations.

### 4. **Decision Support Systems:**

* **Predictive Analytics:** Designing prompts to guide AI models in analyzing historical data and predicting future trends or outcomes.
* **Recommendation Systems:** Creating prompts to personalize recommendations based on user preferences and historical behavior.

### 5. **Ethical and Legal Compliance:**

* **Bias Mitigation:** Designing prompts to minimize biases in AI outputs, ensuring fairness and equity in decision-making processes.
* **Privacy Protection:** Crafting prompts to adhere to legal and ethical guidelines regarding data privacy and security.

### 6. **Scientific Research and Analysis:**

* **Data Interpretation:** Developing prompts to guide AI models in interpreting complex scientific data and generating insights.
* **Hypothesis Testing:** Crafting prompts to formulate and test hypotheses based on experimental data or research findings.

### 7. **Healthcare and Medical Applications:**

* **Diagnosis Assistance:** Designing prompts to assist AI models in diagnosing medical conditions based on symptoms and patient history.
* **Patient Monitoring:** Creating prompts to monitor patient data and provide timely alerts or recommendations for healthcare professionals.

### 8. **Financial Services:**

* **Risk Assessment:** Crafting prompts to guide AI models in assessing financial risks and making data-driven decisions.
* **Fraud Detection:** Designing prompts to detect anomalies and fraudulent activities in financial transactions.

### 9. **Education and Training:**

* **Personalized Learning:** Developing prompts to personalize educational content and adapt learning materials based on student progress and preferences.
* **Skill Development:** Crafting prompts to guide AI models in recommending learning resources and activities to enhance specific skills.

### 10. **Automated Customer Support:**

* **Issue Resolution:** Designing prompts to facilitate AI models in troubleshooting customer issues and providing relevant solutions.
* **Feedback Analysis:** Crafting prompts to analyze customer feedback and sentiment to improve service offerings.
* Introduction to generative AI
* Introduction to generative AI

**Introduction to generative AI:**

Generative AI refers to a subset of artificial intelligence (AI) that focuses on creating new content or outputs, such as images, text, music, and even videos, that mimic human-like creativity. Unlike traditional AI systems that primarily analyze data or perform specific tasks based on predefined rules, generative AI models have the ability to generate original content autonomously.

**Introduction to LLMs and it’s working:**

Large Language Models (LLMs) are a type of artificial intelligence model designed to understand and generate human-like text based on large datasets. They are built using deep learning techniques, particularly neural networks, and are trained on vast amounts of text data. Examples of LLMs include GPT (Generative Pre-trained Transformer) models by OpenAI, BERT (Bidirectional Encoder Representations from Transformers) by Google, and many others.

### Benefits and Challenges

#### Benefits:

* **Versatility:** LLMs can be applied to a wide range of natural language processing tasks with minimal adjustments.
* **Human-like Understanding:** They can generate coherent and contextually relevant text, making interactions more natural.

#### Challenges:

* **Resource Intensive:** Training and deploying LLMs require significant computational resources.
* **Bias and Fairness:** LLMs can inherit and even amplify biases present in the training data, raising ethical concerns.
* **Interpretability:** Understanding why a model made a particular decision or generated a specific text can be difficult due to the complexity of the model.

**Prompting errors and limitations:**

### Prompting Errors in LLMs

1. **Ambiguous Prompts:**
   * **Description:** Lack of clarity or multiple interpretations can lead to unexpected responses.
   * **Example:** "Tell me about Paris."
   * **Solution:** Be specific, e.g., "Tell me about the tourist attractions in Paris, France."
2. **Overly Complex Prompts:**
   * **Description:** Asking for multiple things in one prompt can confuse the model.
   * **Example:** "Explain photosynthesis and tell me a joke about plants."
   * **Solution:** Break it down: "Explain photosynthesis." then "Tell me a joke about plants."
3. **Leading Prompts:**
   * **Description:** Prompts implying a specific answer can bias the response.
   * **Example:** "Isn't the sky blue because of the ocean?"
   * **Solution:** Ask neutral questions: "Why is the sky blue?"
4. **Contextual Misalignment:**
   * **Description:** Lack of context or misaligned context leads to irrelevant responses.
   * **Example:** Asking a follow-up without context.
   * **Solution:** Provide clear context, e.g., "Earlier, we discussed the weather. Can you tell me more about climate change?"

### Limitations of LLMs

1. **Knowledge Cutoff:**
   * **Description:** Models don't have information beyond their training data.
   * **Example:** Asking about recent events post-training.
   * **Solution:** Use current sources for up-to-date information.
2. **Understanding and Reasoning:**
   * **Description:** LLMs don't truly understand or reason but predict based on patterns.
   * **Example:** Poor performance on complex logical tasks.
   * **Solution:** Verify complex information from reliable sources.
3. **Bias and Ethical Concerns:**
   * **Description:** Models can reflect biases in training data.
   * **Example:** Generating biased or offensive content.
   * **Solution:** Implement filtering and monitor outputs for bias.
4. **Consistency:**
   * **Description:** Responses can be inconsistent for similar questions.
   * **Example:** Different answers for slight variations of the same question.
   * **Solution:** Standardize prompt phrasing.
5. **Length and Complexity of Responses:**
   * **Description:** Long or complex prompts can lead to incomplete responses.
   * **Example:** Detailed prompts resulting in truncated responses.
   * **Solution:** Keep prompts concise and break down complex queries.
6. **Sensitivity to Prompt Variations:**
   * **Description:** Small changes in wording can lead to different responses.
   * **Example:** "What are the benefits of exercise?" vs. "Why is exercise good for you?"
   * **Solution:** Test and refine prompt formulations.

**What is Tailwind CSS:**

**Tailwind CSS** is a utility-first CSS framework that provides low-level utility classes for building custom designs directly in your HTML. Unlike traditional CSS frameworks that provide pre-designed components, Tailwind focuses on offering a comprehensive set of utility classes to help you style your elements without writing custom CSS.

### Key Features of Tailwind CSS

1. **Utility-First Approach:**
   * Tailwind CSS offers utility classes that apply specific styles directly to elements in your HTML.
   * Example: <div class="text-center text-blue-500">Hello, Tailwind!</div>
2. **Customization:**
   * Tailwind is highly customizable via a configuration file (tailwind.config.js), allowing you to tailor the design system to your project's needs.
   * Example: Customizing colors, spacing, and other design tokens.
3. **Responsive Design:**
   * Tailwind provides responsive utility variants, making it easy to apply different styles at different screen sizes.
   * Example: <div class="text-center md:text-left">Responsive Text</div>
4. **State Variants:**
   * Tailwind includes variants for different states like hover, focus, and active.
   * Example: <button class="bg-blue-500 hover:bg-blue-700">Hover Me</button>
5. **Flexibility:**
   * Tailwind's utility classes allow for flexible design by composing styles directly in your markup.
   * Example: <div class="flex items-center justify-center h-screen">Centered Content</div>
6. **No Unused CSS:**
   * Tailwind includes a purge option to remove unused CSS, optimizing the final CSS bundle size.
   * Example: Configure purge in tailwind.config.js to scan your files and remove unused styles.

### Benefits of Using Tailwind CSS

1. **Productivity:**
   * Faster development with utility classes, reducing the need to write custom CSS.
   * Consistent design patterns through reusable classes.
2. **Maintainability:**
   * Easier to manage and update styles directly in HTML.
   * Clear and concise class names that describe the styles applied.
3. **Consistency:**
   * Ensures a consistent design system across the project by using predefined utility classes.
   * Centralized configuration for design tokens like colors, spacing, and typography.

**How Tailwind is Differ from Others css like Bootstrap:**

Tailwind CSS and Bootstrap are two popular CSS frameworks, but they have different philosophies and approaches to styling web applications. Here are some key differences:

### 1. ****Design Philosophy:****

* **Bootstrap:**
  + Provides a set of pre-designed components like buttons, navbars, modals, etc., that follow a consistent design language.
  + Emphasizes ease of use and getting started quickly with a uniform look.
  + Uses a grid system and utility classes but focuses more on components.
* **Tailwind CSS:**
  + Provides low-level utility classes that can be combined to build custom designs without any opinionated styling.
  + Emphasizes flexibility and customizability, allowing developers to create unique designs without overriding pre-styled components.

### 2. ****Customization:****

* **Bootstrap:**
  + Customization is achieved through overriding styles or using SASS variables to change the look of components.
  + Has a more rigid structure, which can be limiting if a custom design is needed.
* **Tailwind CSS:**
  + Built with customization in mind; you can create custom themes by modifying the Tailwind configuration file.
  + Provides extensive utility classes for spacing, typography, colors, and more, allowing for fine-grained control over styling.

### 3. ****Component-Based vs. Utility-First:****

* **Bootstrap:**
  + Component-based framework where you use pre-built components like buttons, forms, cards, etc.
  + Faster to implement common UI patterns but can lead to a more uniform look across different projects.
* **Tailwind CSS:**
  + Utility-first framework where you use utility classes directly in your HTML to style elements.
  + Requires writing more classes in HTML but offers greater flexibility and avoids the "one-size-fits-all" issue.

### 4. ****Learning Curve:****

* **Bootstrap:**
  + Easier for beginners to pick up because of its pre-designed components and simpler class structure.
  + Quick to implement but can become complex when heavy customization is needed.
* **Tailwind CSS:**
  + Steeper learning curve initially because it requires understanding and using utility classes.
  + More efficient and flexible once you get used to it, as it allows for rapid and consistent styling.

### 5. ****File Size:****

* **Bootstrap:**
  + Includes styles for all components, which can lead to a larger file size.
  + Requires additional steps to tree-shake unused CSS.
* **Tailwind CSS:**
  + Uses a "purge" feature to remove unused styles in production, resulting in a smaller file size.
  + Utility classes mean only the styles you use are included, which can be more efficient.

### 6. ****Performance:****

* **Bootstrap:**
  + Can become bulky if you don't remove unused components.
  + May require additional optimization for performance.
* **Tailwind CSS:**
  + Designed to be performant by default with its utility-first approach.
  + Easier to maintain a lean CSS file with the purge feature.

### Summary:

* **Bootstrap** is great for quickly getting a consistent look and feel with pre-designed components, making it ideal for projects where time-to-market is critical and design consistency is a priority.
* **Tailwind CSS** is perfect for projects that require a high level of customization and flexibility, allowing developers to craft unique designs without being constrained by pre-styled components.

### ACTIVITY LOG FOR THE SECOND WEEK

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **Day**  **&**  **Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person InCharge**  **Signature** |
| Day – 1  (06-06-2024) | Installing the Tailwind css & Knowing the use of Tailwind css | * Installation and setup. * Generated Input.css and output.css. * Creating a simple index.html file. |  |
| Day - 2  (07-06-2024) | Creating a Contact form using Eclipse | * Know about creating the contact form * Learn about the Eclipse |  |
| Day – 3  (08-06-2024) | Prompt Structure  How to give a prompt in Chatgpt | * Problem solving with Prompts * Understanding the prompt Engineering * Advanced prompting Techniques |  |
| Day – 4  (10-06-2024) | Creating the GitHub account | * Github account creation * Having the gitbash |  |
| Day – 5  (11-06-2024) | Working on the git using Commands | * Understanding the Gihub fundamentals. * Basic Git commands. * Project management. |  |
| Day –6  (12-06-2024) | Cloning a repository using Git | Learn how to learn Cloning the repository |  |

**WEEKLY REPORT**

**WEEK – 2 (From Dt: 06-06-2024 to Dt: 12-06-2024)**

**Objective of the Activity Done:**

**Detailed Report:**

### Overview of Tailwind CSS

### Tailwind CSS differs from traditional CSS frameworks like Bootstrap or Foundation in that it doesn't come with pre-designed components. Instead, it provides a set of utility classes that you can use directly in your HTML markup to style elements.

### Uses of Tailwind CSS

Tailwind CSS is a versatile and powerful CSS framework that can be used for a wide range of applications and scenarios. Here are some common uses of Tailwind CSS:

#### 1. **Rapid Prototyping**

* **Quick Development:** Tailwind CSS allows developers to quickly create prototypes and mockups with utility classes, speeding up the design and iteration process.
* **No Custom CSS:** By using predefined utility classes, you can avoid writing custom CSS during the initial stages of development.

#### 2. **Customizable Design System**

* **Consistent Styling:** Tailwind CSS provides a consistent set of design tokens and utility classes that can be easily customized to match your brand's design system.
* **Theme Customization:** You can define custom colors, spacing, typography, and more in the tailwind.config.js file to create a unique design system.

#### 3. **Responsive Design**

* **Built-In Breakpoints:** Tailwind CSS includes responsive design capabilities with breakpoint prefixes (sm:, md:, lg:, xl:) that allow you to apply different styles at different screen sizes.
* **Mobile-First Approach:** Tailwind CSS encourages a mobile-first design approach, making it easier to build responsive layouts.

#### 4. **Utility-First CSS**

* **Utility Classes:** Tailwind CSS provides a comprehensive set of utility classes for common CSS properties such as margin, padding, colors, fonts, and more, allowing you to build designs directly in your HTML.
* **No CSS Overhead:** By using utility classes, you can avoid the overhead of writing custom CSS and keep your styles organized and maintainable.

#### 5. **Component-Based Design**

* **Reusable Components:** Tailwind CSS works well with component-based frameworks like React, Vue, and Angular, enabling you to create reusable UI components with consistent styling.
* **Inline Styling:** You can use Tailwind's utility classes directly within your component templates, making it easy to style components without external stylesheets.

#### 6. **Production-Ready**

* **PurgeCSS Integration:** Tailwind CSS can be configured to remove unused CSS classes in production builds, resulting in smaller CSS file sizes and improved performance.
* **Optimized Performance:** Tailwind's approach to CSS minimizes the amount of custom CSS needed, reducing the risk of bloated stylesheets.

#### 7. **Theming and Customization**

* **Custom Themes:** Tailwind CSS allows you to create custom themes by extending its configuration file, making it easy to switch themes or create variations.
* **Dark Mode:** Tailwind CSS supports dark mode out of the box, allowing you to create dark mode designs effortlessly.

#### 8. **Consistent UI**

* **Design Consistency:** Tailwind CSS helps maintain design consistency across your application by using a standardized set of utility classes.
* **Shared Design Language:** Teams can share a common design language with Tailwind CSS, reducing the chance of inconsistent styles and improving collaboration.

Tailwind CSS:

STEP-1: NEW FOLDER AND ADDING TO WORKSPACE(VS CODE-RDE)

STEP-2: SETTINGS IN VS CODE(EXTENSION SETTINGS->NULL->CHOROME DESTIMATION) NOTE: CLOSE THE UNNECCESARY FILES ON THE WORKSPACE

STEP3: TO INCLUDE THE AI MODEL WE REQ A BACKEND TECH(NODE JS)(MAKE IT INSTALL IN THE PLATFORM/SYSTEM)

STEP-4: INSTALLTION PROCESS OF TAILWIND WITH OFFICIAL DOCUMENTATION

https://tailwindcss.com/docs/installation

STEP-5: SINGLE COMMAND ACCES HAS TO BE THERE TO THE TERIMAL(TO THE WORKSPACE)

STEP-6: CREATING THE FOLDER SRC(SOURCE)-->INPUT.CSS-->OUTPUT.CSS (GENERATED)

STEP-7: INDEX HTML FILE

Contact Form Using Eclipse

1)create a folder and name it as html\_prj

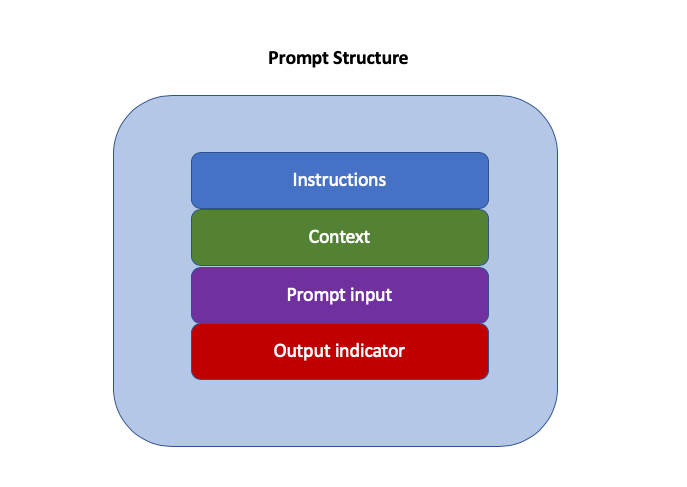
2)add the same folder to vs code--> index.html-->contact form

-->give me a contact with tailwind css with 4 main basic fields on enquirey

3) eclipse--> file--> new-> other-->dynamic web project-->name (first letter Uppercase)-->next\*2-->generate xml(tick)

4) https://tomcat.apache.org/download-10.cgi download apache tomcat(server)

5)add the files to the src folder in eclipse from vs code/folder that you have created in step-1

HOW TO GIVE PROMPT IN CHATGPT:-

* 1. **Be Clear and Specific**
  2. Ask Direct Questions
  3. **Provide Context When Necessary**
  4. **Use Examples**
  5. Ask for Recommendations or Best Practices

***Creating a Account in GitHub:-***

Creating a GitHub account is straightforward. Follow these steps to get started:

### Step-by-Step Guide to Creating a GitHub Account

1. **Visit the GitHub Website:**
   * Open your web browser and go to [GitHub](https://github.com/).
2. **Sign Up:**
   * Click on the **"Sign up"** button located at the top-right corner of the homepage.
3. **Enter Your Details:**
   * **Username:** Choose a unique username for your account. This will be your public identifier on GitHub.
   * **Email Address:** Enter a valid email address. You'll need this for account verification and communication.
   * **Password:** Choose a strong password. GitHub will show a strength indicator to help you choose a secure password.
4. **Verify You're Not a Robot:**
   * Complete the CAPTCHA challenge to verify that you are a human.
5. **Choose Your Plan:**
   * GitHub offers both free and paid plans. Select the plan that suits your needs. Most users start with the free plan.
6. **Verify Your Email:**
   * GitHub will send a verification email to the address you provided. Go to your email inbox, find the email from GitHub, and click on the verification link.
7. **Complete the Setup:**
   * After verifying your email, you may be asked a few questions to customize your experience. This can include selecting topics of interest or setting up a GitHub organization if you're working as part of a team.

**Basic Git Commands:-**

* git init:-
  + Initializes a new Git repository in the current directory.

|  |
| --- |
| git init |

* git clone:-
  + Creates a copy of an existing repository.

|  |
| --- |
| git clone <repository-url> |

* **git status:-**
  + Displays the state of the working directory and the staging area.

|  |
| --- |
| git status |

* **git add**
  + Adds changes in the working directory to the staging area.

|  |
| --- |
| git add <file>  git add . |

* **git commit**
  + Records changes to the repository with a message describing the changes.

|  |
| --- |
| git commit -m “Commit message” |

* **git push**
* Uploads local repository content to a remote repository.

|  |
| --- |
| git push origin <branch> |

* **git pull**
* Fetches and merges changes from a remote repository to the local branch.

|  |
| --- |
| git pull origin <branch> |

**Clone a Repository**:

* Find the repository you want to clone on GitHub.
* Click on the Code button and copy the URL of the repository (HTTPS or SSH).
* Open a terminal or command prompt.
* Navigate to the directory where you want to clone the repository:

|  |
| --- |
| cd /path/to/your/directory |

Clone the repository:

|  |
| --- |
| git clone https://github.com/username/repository.git |

### 

### ACTIVITY LOG FOR THE THIRD WEEK

|  |  |  |  |
| --- | --- | --- | --- |
| **Day**  **&**  **Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person InCharge**  **Signature** |
| Day – 1  (13-06-2024) | Briefly Introduction about Java | * Define Java * Components of java * Structure of Java |  |
| Day - 2  (14-06-2024) | What is NLP? | * Introduction to NLP * Key concepts of NLP |  |
| Day – 3  (15-06-2024) | Debugging in Ai model | * What is Debuging * Steps to Debug |  |
| Day – 4  (18-06-2024) | Generating different otp’s using java code Java oops concepts | * Encapsulation * Abstraction * Inheritance * polymorphism |  |
| Day – 5  (19-06-2024) | What is No SQLl & My SQL | * What is SQL? * What is NoSQL * Difference of No SQL & My SQL |  |
| Day –6  (20-06-2024) | Test Activities & Uses of Java  Programming | * Uses of Java |  |

**WEEKLY REPORT**

**WEEK – 3 (From Dt: 06-06-2024 to Dt: 12-06-2024)**

**Objective of the Activity Done:**

**Detailed Report:**

Object-Oriented Programming (OOP) :- in Java is a programming paradigm that uses "objects" to design applications and programs. It provides a clear modular structure for programs which makes it good for defining abstract data types where implementation details are hidden, promoting reuse, and making software easier to maintain. Here are the main OOP concepts in Java:

### Classes and Objects

* **Class**: A blueprint for creating objects. A class defines a datatype by bundling data and methods that work on the data into one single unit.
* **Object**: An instance of a class. Objects are created from classes.

### Inheritance

Inheritance is a mechanism where a new class inherits the properties and methods of an existing class. The class that inherits is called the subclass (or derived class), and the class being inherited from is called the superclass (or base class).

### Polymorphism

Polymorphism allows methods to do different things based on the object it is acting upon. It can be achieved through method overloading and method overriding.

* **Method Overloading**: Multiple methods in the same class with the same name but different parameters.

**Method Overriding**: A subclass provides a specific implementation for a method that is already defined in its superclass.

### Encapsulation

Encapsulation is the wrapping up of data under a single unit. It is the mechanism that binds together the code and the data it manipulates, and keeps both safe from outside interference and misuse.

### Abstraction

Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. Abstract classes and interfaces are used to achieve abstraction.

|  |
| --- |
| **@abstractmethod**  **def sound(self):**  **pass**  **class Dog(Animal):**  **def sound(self):**  **return "Woof!"**  **class Cat(Animal):**  **def sound(self):**  **return "Meow!"**  **# Instantiate the objects**  **dog = Dog()**  **cat = Cat()**  **print(dog.sound()) # Output: Woof!**  **print(cat.sound()) # Output: Meow!** |

* **Abstract Class**: A class that cannot be instantiated and is used to define a template for subclasses.

**Interface**: A reference type in Java, it is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Structure of java

Java programs are structured in a specific way to adhere to the conventions and requirements of the Java language. Understanding the structure of a Java program is essential for writing, compiling, and running Java code efficiently. Below is an overview of the typical structure of a Java program:



NLP (natural language processing)

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human languages. It involves the use of algorithms to process and analyze large amounts of natural language data. Here’s a brief overview of key NLP concepts and a small example using Python's NLTK library.

### Key Concepts in NLP

1. **Tokenization**: Splitting text into individual words or sentences.
2. **Part-of-Speech Tagging**: Identifying the grammatical parts of speech for each word in a sentence.
3. **Named Entity Recognition (NER)**: Identifying entities such as names, dates, and organizations in text.
4. **Sentiment Analysis**: Determining the sentiment expressed in a piece of text.
5. **Text Classification**: Assigning categories or labels to text.
6. **Machine Translation**: Translating text from one language to another.
7. **Topic Modeling**: Identifying topics present in a set of documents.

**Debugging of an AI Model:-**

Debugging a malfunction in an AI model involves a series of steps to identify, diagnose, and fix the issue. Here’s a structured approach to debugging AI models:

### Steps to Debug an AI Model

1. **Identify the Issue**
   * **Symptoms**: What are the symptoms of the malfunction? (e.g., poor accuracy, high error rates, unexpected outputs)
   * **Context**: When does the issue occur? (e.g., specific inputs, during training, during inference)
2. **Collect Data and Logs**
   * **Data**: Gather the input data that causes the malfunction.
   * **Logs**: Collect logs from the model training and inference phases.
3. **Check Data Preprocessing**
   * **Data Quality**: Ensure the input data is clean and correctly preprocessed.
   * **Normalization/Scaling**: Verify that data normalization or scaling is correctly applied.
4. **Inspect Model Architecture**
   * **Layers and Parameters**: Check if the model layers and parameters are correctly defined.
   * **Compatibility**: Ensure that the architecture is compatible with the input data.
5. **Evaluate Model Training**
   * **Training Loss**: Analyze the training loss over epochs to detect issues like overfitting or underfitting.
   * **Validation Metrics**: Compare training and validation metrics to identify generalization problems.
6. **Examine Hyperparameters**
   * **Learning Rate**: Ensure the learning rate is appropriate.
   * **Batch Size**: Check if the batch size is suitable for the data and model.
   * **Other Hyperparameters**: Evaluate other hyperparameters that might affect model performance.
7. **Run Controlled Experiments**
   * **Ablation Studies**: Remove or modify parts of the model to see how it affects performance.
   * **Simplify Model**: Start with a simpler version of the model to isolate the problem.
8. **Check for Overfitting/Underfitting**
   * **Regularization**: Ensure regularization techniques (e.g., dropout, L2 regularization) are correctly implemented.
   * **Data Augmentation**: Verify that data augmentation is appropriately applied to increase generalization.
9. **Validate Inference Pipeline**
   * **Model Deployment**: Check if the model is correctly deployed.
   * **Inference Data**: Ensure the data used during inference is processed similarly to the training data.
10. **Utilize Debugging Tools**
    * **TensorBoard**: Use TensorBoard to visualize training progress and identify anomalies.
    * **Debugging Libraries**: Use libraries like pdb in Python or specialized debugging tools for deep learning frameworks

### SQL (Structured Query Language)

SQL is a standardized programming language used for managing and manipulating relational databases. It is designed for querying, updating, and managing data within a relational database management system (RDBMS). SQL databases are table-based, where data is stored in rows and columns. Some key features of SQL databases include:

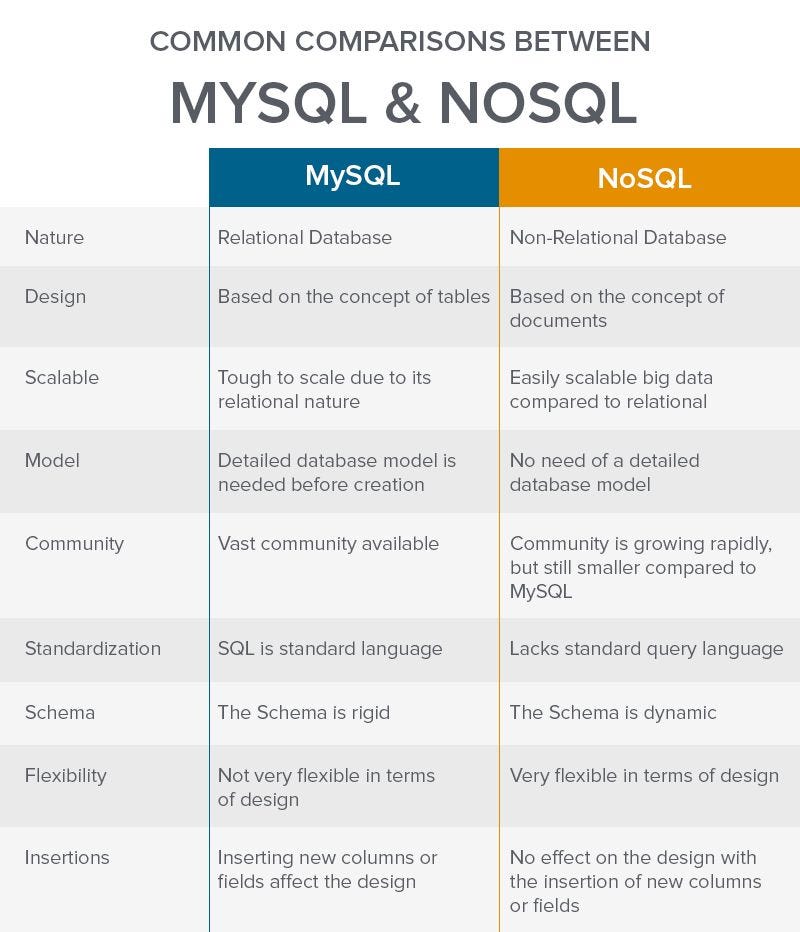
* **Schema-based**: SQL databases have a fixed schema, meaning the structure of data (tables, columns, data types) must be defined before inserting data.
* **ACID Compliance**: SQL databases follow ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring reliable transactions.
* **Relationships**: SQL databases use foreign keys to establish relationships between tables, allowing for complex queries and data integrity.
* **Examples**: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

### NoSQL (Not Only SQL)

NoSQL is an umbrella term for a diverse set of database management systems that are designed to handle large volumes of unstructured, semi-structured, or structured data. NoSQL databases provide flexible schemas and are optimized for performance and scalability. They are often used in big data and real-time web applications. Some key features of NoSQL databases include:

* **Schema-less**: NoSQL databases are schema-less or have dynamic schemas, allowing for the storage of varied data formats without predefined structures.
* **Horizontal Scalability**: NoSQL databases are designed to scale out by distributing data across multiple servers.
* **Flexible Data Models**: NoSQL databases support various data models, including document-based, key-value pairs, wide-column stores, and graph databases.
* **Examples**: MongoDB (document-based), Redis (key-value store), Cassandra (wide-column store), Neo4j (graph database).

In summary, SQL databases are ideal for structured data with complex relationships and require ACID compliance, while NoSQL databases are suitable for handling large volumes of diverse data types with high performance and scalability needs



***Uses of Java Programming:-***

**Common uses of java**

* Web Application
* Mobile Application
* Desktop Application
* Scientific Application
* Enterprise Application

Java is a powerful, versatile, and widely-used programming language that supports a wide range of application development, from mobile apps to large-scale enterprise systems. Its platform independence, robust features, and extensive standard library make it a popular choice among developers.

1. **Web Development**:
   * Java is used to build dynamic web applications using frameworks like Spring, Hibernate, and Struts.
   * Java Servlets and JavaServer Pages (JSP) are used to create server-side applications.
2. **Mobile Development**:
   * Java is the primary language for Android app development. The Android SDK (Software Development Kit) uses Java as the main programming language.
3. **Enterprise Applications**:
   * Java is extensively used in enterprise environments for building large-scale, robust, and secure applications. Java Enterprise Edition (Java EE) provides APIs and runtime environments for scripting enterprise software, including network applications and web services.
4. **Desktop GUI Applications**:
   * Java provides tools like Swing, AWT (Abstract Window Toolkit), and JavaFX for building graphical user interfaces (GUIs) for desktop applications.
5. **Scientific and Research Applications**:
   * Java is used in scientific computing and research for its portability, reliability, and vast array of available libraries. It is often used for simulations and mathematical computations.
6. **Embedded Systems**:
   * Java is used in developing applications for embedded systems such as smart TVs, set-top boxes, and Blu-ray players due to its portability and robustness.
7. **Big Data Technologies**:
   * Java plays a significant role in big data technologies. Apache Hadoop, a popular framework for processing large data sets, is written in Java.
8. **Cloud-Based Applications**:
   * Java is used for building cloud-based applications due to its ability to scale and handle distributed computing effectively. Many cloud providers offer robust support for Java applications.
9. **Games Development**:
   * Java is used for game development, especially for Android-based games and some desktop games. The Java 2D and 3D APIs help in creating rich graphical content.
10. **Financial Services**:
    * Java is widely used in financial services for creating server-side applications, electronic trading systems, and other financial applications due to its performance and reliability.
11. **Internet of Things (IoT)**:
    * Java is used in IoT for building applications that can run on a wide range of devices, from small sensors to large servers.
12. **Distributed Applications**:
    * Java's Remote Method Invocation (RMI) and Enterprise JavaBeans (EJB) are used to build distributed applications that can run on multiple servers across a network.

The widespread use of Java is attributed to its platform independence, strong community support, extensive libraries, and its performance, security, and scalability features.

### ACTIVITY LOG FOR THE FOURTH WEEK

|  |  |  |  |
| --- | --- | --- | --- |
| **Day**  **&**  **Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person InCharge**  **Signature** |
| Day – 1  (21-06-2024) | Creating a WebPage Using Flowbite  NLU & LLM | * What is a webpage? * What is Nlu? * What is Llm? |  |
| Day – 2  (22-06-2024) | Briefly Description of Rasa | * What is Rasa? * Why we use Rasa? |  |
| Day – 3  (24-06-2024) | Structure of Rasa | * Structure of Rasa * What are Rasa file * Components of Rasa |  |
| Day – 4  (25-06-2024) | Briefly Discuss about the nlu, domain.yml, Stories.yml and actions.py file | * Use of Nlu in Rasa &its structure * Use of Domain.yml & its Structure * Use of Stories in Rasa &its structure |  |
| Day – 5  (26-06-2024) | Creating the Rasa framework in Vs code | * How to Create a Rasa Framework |  |
| Day –6  (27-06-2024) | Test on weekly reports & Discuss about commands in Rasa Framework | * Learn Commands on rasa * Uses of rasa Commands |  |

**WEEKLY REPORT**

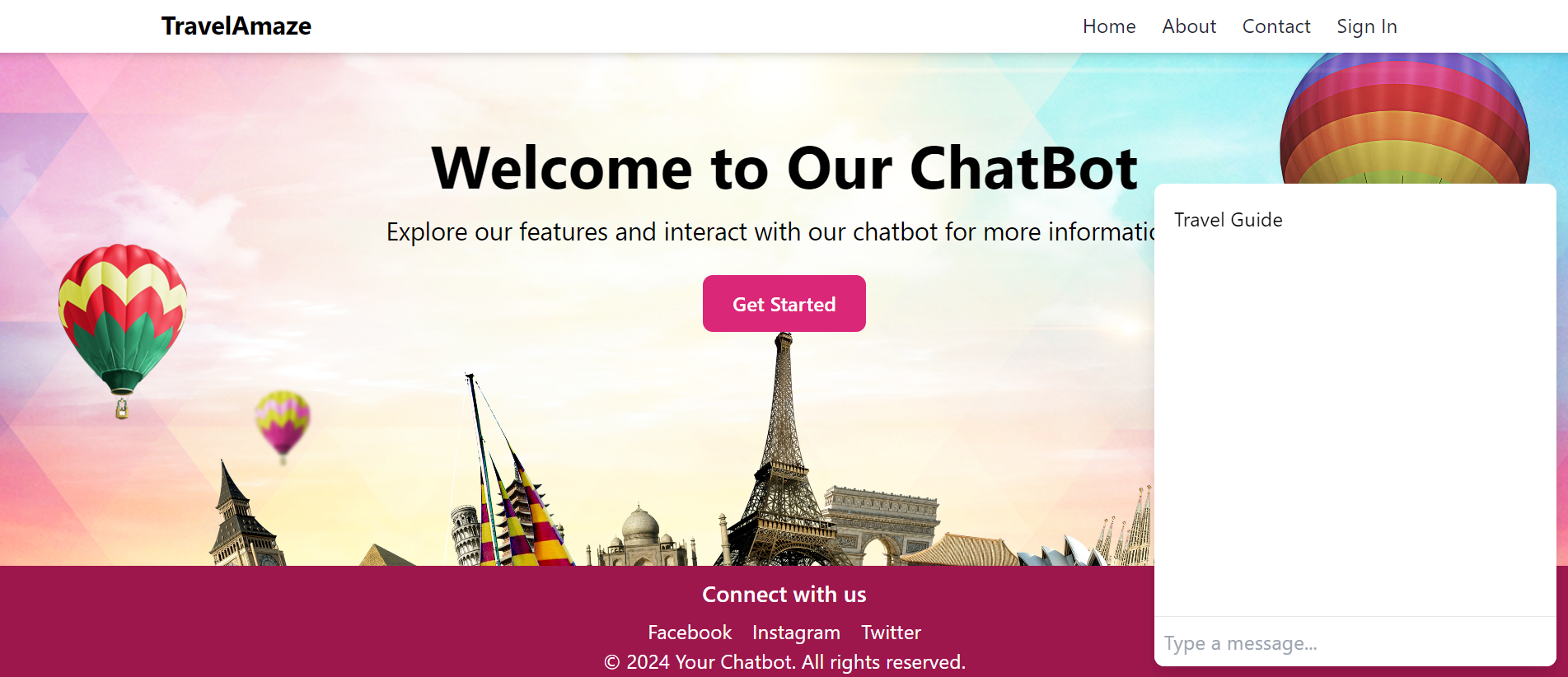
**WEEK – 4 (From Dt: 20-06-2024 to Dt: 27-06-2024)**

**Objective of the Activity Done:**

**Detailed Report:**

**WEBPAGE:-**

A webpage is a single document or resource of content on the World Wide Web that is typically accessed through a web browser. It can contain text, images, multimedia elements, and hyperlinks to other webpages or resources, providing information, functionality, or entertainment to users.



NLU:-

NLU stands for Natural Language Understanding, which is a branch of artificial intelligence (AI) that focuses on enabling machines to comprehend and interpret human language in a meaningful way. NLU allows computers to analyze and derive meaning from text or speech inputs, facilitating tasks such as sentiment analysis, entity recognition, language translation, and intent detection in chatbots and virtual assistants. It involves complex algorithms and techniques to process and understand the nuances of human language, bridging the gap between human communication and machine understanding.

LLM:-

LLM stands for Large Language Model. LLMs have demonstrated capabilities in natural language understanding, translation, summarization, and even creative writing tasks. They represent a significant advancement in AI technology, enabling applications across various fields, including chatbots, content generation, and language translation.

**What is Rasa**

Rasa is an open-source framework for building conversational AI chatbots and assistants. It provides tools and libraries to develop and deploy AI-powered chatbots that can understand natural language and engage in meaningful conversations with users. Here's an overview of Rasa and its key features:

### Overview of Rasa

1. **Open-Source Framework:**
   * Rasa is entirely open-source, allowing developers to customize and extend its capabilities to suit specific project requirements without vendor lock-in.
2. **Natural Language Understanding (NLU):**
   * Rasa includes tools for NLU, enabling chatbots to comprehend user intents, extract entities (such as dates, locations, and names), and handle dialogue contextually.
3. **Dialogue Management:**
   * Rasa manages dialogue flows, enabling chatbots to maintain context across conversations, handle multi-turn interactions, and provide relevant responses based on user inputs.
4. **Integration with Machine Learning:**
   * Rasa integrates machine learning techniques, including supervised learning for intent classification and entity recognition, and reinforcement learning for optimizing dialogue policies.
5. **Scalability and Customization:**
   * Rasa supports scalability for large-scale deployments and offers flexibility in customization through its modular architecture and extensive libraries.
6. **Deployment Options:**
   * Chatbots built with Rasa can be deployed on various platforms, including cloud services, on-premise servers, and messaging channels like Slack, Facebook Messenger, and WhatsApp.

### Components of Rasa

1. **Rasa NLU:**
   * Handles natural language understanding tasks, including intent classification and entity extraction.
2. **Rasa Core:**
   * Manages dialogue flow and conversation management, ensuring seamless interactions with users.
3. **Rasa SDK:**
   * Allows developers to extend Rasa's functionality with custom actions and logic.
4. **Rasa X:**
   * An additional toolset for managing, improving, and testing Rasa-powered chatbots through a user-friendly interface.

### Use Cases

* **Customer Support:** Automate customer queries and provide instant assistance.
* **Virtual Assistants:** Build intelligent assistants for tasks like scheduling, reminders, and information retrieval.
* **Enterprise Applications:** Integrate with business systems to streamline operations and enhance user interactions.

### Conclusion

Rasa provides developers with powerful tools to create sophisticated conversational AI applications. Its flexibility, scalability, and focus on open-source principles make it a popular choice for building chatbots and virtual assistants that can engage users effectively across various platforms and use cases.

Structure Of Rasa:-

|  |
| --- |
|  |

Components of Rasa:-

Rasa is comprised of several key components that work together to enable the creation, training, and deployment of conversational AI applications. Here are the main components of Rasa:

 **Rasa NLU (Natural Language Understanding):**

* Responsible for understanding and extracting intents (what the user wants) and entities (specific details) from natural language inputs (text or speech). It uses machine learning models to classify intents and extract entities from user messages.

 **Rasa Core (Dialogue Management):**

* Manages the dialogue flow and decides how the assistant should respond based on the current conversation context, previous interactions, and the predicted user intent. Rasa Core uses policies and machine learning models to predict the next best action (response) for the assistant.

Nlu.yml***:-***

In Rasa, NLU (Natural Language Understanding) is used to interpret and understand the intent behind user messages. Here's why NLU is crucial and how it fits into the overall functionality of Rasa:

### Importance of NLU in Rasa:

1. **Intent Classification:**
   * NLU in Rasa is responsible for classifying the intent of user messages. An intent represents the goal or purpose behind what the user wants to achieve through their interaction with the chatbot or assistant. For example, if a user says "Book a flight to New York," the intent might be classified as book\_flight.
2. **Entity Extraction:**
   * Alongside intents, NLU also extracts entities from user messages. Entities are specific pieces of information within the user input that are relevant to fulfilling the intent. In the example above, "New York" could be extracted as an entity representing the destination.

**DOMAIN.**YML

The domain.yml file is a crucial component in developing chatbots using the Rasa framework. It essentially serves as the brain of your chatbot, defining its domain or universe of knowledge and actions. Here’s a brief overview of what it typically includes:

1. **Intents:** Defines the user’s intentions or goals when interacting with the chatbot. For example, intents could include "greeting", "booking", "information\_request", etc.
2. **Entities:** Specifies specific pieces of information that the chatbot needs to extract from user messages. Entities might include dates, locations, names, etc.
3. **Slots:** Represents the information the chatbot needs to keep track of during the conversation. Slots can be filled by entities extracted from user inputs or through other actions.
4. **Actions:** Describes what actions the chatbot can take, such as uttering responses, calling an API, or executing custom code.
5. **Responses:** Contains the text responses that the chatbot can use to reply to users based on different intents and scenarios.
6. **Forms:** Defines forms that guide the chatbot through collecting information from the user to fulfill a specific task, like booking a hotel or flight.
7. **Templates:** Predefined templates for responses that can be used across different parts of the conversation.

Overall, the domain.yml file structures how your chatbot understands user inputs, processes information, and generates appropriate responses, making it a fundamental piece of any Rasa-based chatbot project.

STORIES.YML:-

Certainly! The stories.yml file in the context of Rasa framework projects serves as a collection of conversational paths or stories that the chatbot can follow during interactions with users. Here’s a brief explanation of what it typically includes:

1. **Conversational Paths:** Stories define sequences of user inputs (intents) and the corresponding actions (bot responses or actions) that the chatbot should follow in various scenarios. Each story represents a potential dialogue flow between the user and the bot.
2. **Intent Sequences:** Stories describe the sequence of intents that a user might send to the chatbot during a conversation. For instance, a story could start with a "greeting" intent, followed by an "information\_request" intent, and then end with a "thank\_you" intent.
3. **Actions:** Alongside intents, stories also specify the actions the chatbot should take in response to each user input. These actions can include sending a message, asking for information, calling an API, or any other task defined in the Rasa actions.
4. **Conditions and Branching:** Stories can include conditions to handle different paths based on slot values, previous actions, or any other context. This allows the chatbot to adapt its responses based on the user’s current state or request.
5. **Training Data:** The stories file serves as crucial training data for the chatbot’s machine learning models. By presenting different dialogues and scenarios, it helps the model learn how to predict the next action or response based on user inputs.

ACTIONS.PY

In essence, the stories.yml file plays a pivotal role in defining the flow of conversations and training the chatbot to handle a variety of user interactions effectively within the Rasa framework.

In the Rasa framework, the actions.py file is where you define custom actions that your chatbot can perform in response to user inputs or as part of a conversation flow. Here’s a brief introduction to its purpose and functionality:

1. **Custom Actions:** The actions.py file allows you to define Python classes that implement specific actions your chatbot should execute. These actions can range from retrieving data from an external API, querying a database, performing calculations, or any other programmable task.
2. **Interaction with External Systems:** Many chatbots need to interact with external systems to fetch information or perform tasks beyond simple responses. For example, a weather bot might use an action to fetch real-time weather data from an API.
3. **Event Handling:** Actions in actions.py can handle events triggered during the conversation flow, such as slot filling (gathering specific information from the user), confirmation prompts, or any other custom logic you want to integrate into the chatbot’s behavior.
4. **Integration with NLU and Dialogue Management:** Actions defined in actions.py are typically called by the Rasa framework based on the predicted intents and entities from the NLU (Natural Language Understanding) model and the dialogue state managed by the dialogue management system.
5. **Response Generation:** Actions can generate responses that are then sent back to the user via the chat interface. These responses are often defined in the domain.yml file or generated dynamically based on the action’s logic.
6. **Deployment:** When deploying a Rasa chatbot, the actions.py file needs to be running as a server that listens for requests from the Rasa framework. This setup allows the chatbot to execute actions in real-time during conversations with users.

Overall, the actions.py file is crucial for extending the functionality of your Rasa chatbot beyond simple text responses, enabling it to interact with external resources and perform complex tasks based on the user’s needs and inputs.

Creating a Rasa Framework:-

Creating a project in the Rasa framework involves several key steps to set up your chatbot development environment and begin building conversational AI applications. Here’s a structured approach to get started:

### Step 1: Install Rasa

Make sure you have Python installed (preferably Python 3.6+), and then install Rasa Open Source using pip:

**pip install rasa**

### Step 2: Initialize a New Project

Create a new directory for your Rasa project and initialize it using the rasa init command. This command sets up a basic project structure with essential files:

**rasa init**

### Step 3: Understand Project Structure

The rasa init command sets up a basic project structure which typically includes:

* **config.yml:** Configuration file for training your NLU and dialogue management models.
* **domain.yml:** File defining your chatbot’s domain including intents, entities, actions, responses, and more.
* **data/nlu.yml:** Training data for NLU model (intents and examples).
* **data/stories.yml:** Training data for dialogue management (story paths).
* **actions/actions.py:** File to define custom actions your chatbot can perform.
* **credentials.yml:** File to store API keys and other credentials for external services.

### Step 4: Define Your Domain

Edit domain.yml to define your chatbot’s domain including intents (user goals), entities (data to extract), slots (memory for conversation), actions (bot responses or tasks), templates (predefined responses), and forms (data collection flows).

### Step 5: Create Training Data

* **NLU Training Data:** Populate data/nlu.yml with examples of user messages categorized by intents and annotated with entities.
* **Dialogue Training Data:** Create conversational paths in data/stories.yml to outline different dialogues your chatbot should be able to handle.

### Step 6: Define Actions

Implement custom actions in actions/actions.py to perform tasks such as calling APIs, querying databases, or any other external interaction your chatbot requires.

### Step 7: Configure and Train Your Models

Edit config.yml to configure your NLU and dialogue management pipelines (e.g., language model, intent classification, entity extraction). Train your models using:

**rasa train**

### Step 8: Run Your Chatbot

Start your Rasa server to interact with your chatbot via a command line interface:

**rasa shell**

Or, integrate your chatbot into a custom application or platform by running the server:

**rasa run -m models --endpoints endpoints.yml --port 5005 --credentials credentials.yml**

### Step 9: Iterate and Improve

Continuously improve your chatbot by refining training data, adding new intents and responses, enhancing actions, and iterating based on user feedback and testing.

**ACTIVITY LOG FOR THE FIFTH WEEK**

|  |  |  |  |
| --- | --- | --- | --- |
| **Day**  **&**  **Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person InCharge**  **Signature** |
| Day – 1  (28-06-2024) | Activating rasa environment | * Creating environment   Activating environment |  |
| Day - 2  (29-06-2024) | Implementation of chatbot widget through our rasa model | * Chat bot linking   And testing |  |
| Day – 3  (1-07-2024) | Creating our own intents  Entities with rasa | * Intents creation   Entities creation on nlu and stories |  |
| Day – 4  (02-07-2024) | Api fetching through weather  With our data | Creation of weather api |  |
| Day – 5  (03-07-2024) | Creating our own example api | * Api own creating * Intents * Stories * domain |  |
| Day –6  (04-07-2024) | Test on weekly reports & Decision making in python | Test conduction on weekly activities |  |

**WEEKLY REPORT**

**WEEK – 5 (From Dt: 28-05-2024 to Dt )**

**Objective of the Activity Done:**

**Detailed Report**

**Environment Setup and Activation**

**Objectives:**

* Set up the Rasa environment.
* Install necessary dependencies.
* Create and activate a virtual environment.

**Activities:**

1. **Install Rasa:**
   * Install Python (preferably 3.7 or 3.8).
   * Install Rasa using pip:

pip install rasa

1. **Set Up Virtual Environment:**
   * Create a virtual environment:

python -m venv rasa\_env

* + Activate the virtual environment:
    - Windows:

rasa\_env\Scripts\activate

* + - macOS/Linux:

rasa\_env/bin/activate

1. **Initialize Rasa Project:**
   * Create a new Rasa project:

rasa init

* + Follow the prompts to train an initial model and test it.

**Day 2: Running Rasa and Basic Model Training**

**Objectives:**

* Run the Rasa server.
* Train a basic model with default intents.

**Activities:**

1. **Start Rasa Action Server:**
   * In one terminal, start the Rasa action server:

rasa run actions

1. **Start Rasa Server:**
   * In another terminal, start the Rasa server:

rasa run

1. **Interactive Learning:**
   * Use Rasa's interactive learning to train the model further:

rasa interactive

**Implementing a Chatbot Widget**

**Objectives:**

* Integrate a chatbot widget into a web page.
* Test the integration with the Rasa model.

**Activities:**

1. **Install Chatbot Widget:**
   * Use a third-party library like Botfront or Chatroom to integrate a chatbot widget into your web application.
2. **Configure Webhooks:**
   * Set up the Rasa server to handle incoming messages from the widget.
3. **Embed Widget:**
   * Add the widget's HTML/JavaScript code to your web page.
4. **Test the Integration:**
   * Open the web page and interact with the chatbot to ensure it communicates with the Rasa server.

**Creating Custom Intents and Entities**

**Objectives:**

* Define custom intents and entities.
* Update the training data.

**Activities:**

1. **Define Custom Intents:**
   * Open data/nlu.yml and add new intents with training examples.
   * Example:

yaml

Copy code

- intent: ask\_weather

examples: |

- What's the weather like today?

- Tell me the weather

- How's the weather?

1. **Define Entities:**
   * Update data/nlu.yml to include entities.
   * Example:

yaml

Copy code

- intent: ask\_weather

examples: |

- What's the weather like in [New York](location)?

- Tell me the weather in [London](location)

- How's the weather in [San Francisco](location)?

1. **Update Domain File:**
   * Open domain.yml and add intents, entities, and responses.
   * Example:

yaml

Copy code

intents:

- greet

- ask\_weather

entities:

- location

responses:

utter\_weather:

- text: "The weather in {location} is sunny."

1. **Train the Model:**
   * Train the Rasa model with the updated training data:

bash

Copy code

rasa train

**Day 5: Fetching Weather Data via API**

**Objectives:**

* Implement a custom action to fetch weather data from an external API.
* Update the chatbot to use the custom action.

**Activities:**

1. **Set Up Custom Action:**
   * Open actions/actions.py and create a custom action.
   * Example:

python

Copy code

from rasa\_sdk import Action, Tracker

from rasa\_sdk.executor import CollectingDispatcher

import requests

class ActionFetchWeather(Action):

def name(self) -> str:

return "action\_fetch\_weather"

def run(self, dispatcher: CollectingDispatcher,

tracker: Tracker,

domain: Dict[str, Any]) -> List[Dict[str, Any]]:

location = tracker.get\_slot('location')

api\_key = 'YOUR\_API\_KEY'

weather\_api\_url = f'http://api.openweathermap.org/data/2.5/weather?q={location}&appid={api\_key}'

response = requests.get(weather\_api\_url).json()

if response.get("weather"):

weather\_description = response["weather"][0]["description"]

dispatcher.utter\_message(text=f"The weather in {location} is {weather\_description}.")

else:

dispatcher.utter\_message(text=f"Could not fetch weather for {location}.")

return []

1. **Register Custom Action:**
   * Update domain.yml to include the new action.
   * Example:

yaml

Copy code

actions:

- action\_fetch\_weather

1. **Update Stories:**
   * Update data/stories.yml to include the new action.
   * Example:

yaml

Copy code

stories:

- story: fetch weather

steps:

- intent: ask\_weather

- action: action\_fetch\_weather

1. **Restart Servers:**
   * Restart the Rasa action server and Rasa server.

**Day 6: Example API Project**

**Objectives:**

* Create an example project demonstrating API integration with Rasa.
* Implement additional functionalities.

**Activities:**

1. **Project Overview:**
   * Create a project that fetches weather, news, and exchange rates from different APIs.
2. **Custom Actions:**
   * Implement custom actions for fetching data from multiple APIs.
   * Example:

python

Copy code

class ActionFetchNews(Action):

def name(self) -> str:

return "action\_fetch\_news"

def run(self, dispatcher: CollectingDispatcher,

tracker: Tracker,

domain: Dict[str, Any]) -> List[Dict[str, Any]]:

news\_api\_url = 'YOUR\_NEWS\_API\_URL'

response = requests.get(news\_api\_url).json()

if response.get("articles"):

news\_titles = [article["title"] for article in response["articles"][:3]]

dispatcher.utter\_message(text=f"Top news: {', '.join(news\_titles)}")

else:

dispatcher.utter\_message(text="Could not fetch news.")

return []

class ActionFetchExchangeRate(Action):

def name(self) -> str:

return "action\_fetch\_exchange\_rate"

def run(self, dispatcher: CollectingDispatcher,

tracker: Tracker,

domain: Dict[str, Any]) -> List[Dict[str, Any]]:

exchange\_rate\_api\_url = 'YOUR\_EXCHANGE\_RATE\_API\_URL'

response = requests.get(exchange\_rate\_api\_url).json()

if response.get("rates"):

rate\_usd\_to\_eur = response["rates"].get("EUR")

dispatcher.utter\_message(text=f"Exchange rate from USD to EUR is {rate\_usd\_to\_eur}.")

else:

dispatcher.utter\_message(text="Could not fetch exchange rate.")

return []

1. **Update Domain File:**
   * Add new intents, entities, and actions to domain.yml.
   * Example:

yaml

Copy code

intents:

- ask\_news

- ask\_exchange\_rate

actions:

- action\_fetch\_news

- action\_fetch\_exchange\_rate

1. **Update Stories:**
   * Add new stories for handling user queries related to news and exchange rates.
   * Example:

yaml

Copy code

stories:

- story: fetch news

steps:

- intent: ask\_news

- action: action\_fetch\_news

- story: fetch exchange rate

steps:

- intent: ask\_exchange\_rate

- action: action\_fetch\_exchange\_rate

1. **Test the Project:**
   * Train the model and test the chatbot for all functionalities.

**Day 7: Review and Optimize**

**Objectives:**

* Conduct a review of the project.
* Optimize and document the project.

**Activities:**

1. **Review Code:**
   * Conduct a code review to ensure best practices and code quality.
2. **Optimize Performance:**
   * Refine training data and optimize custom action code for better performance.
3. **Documentation:**
   * Document the project setup, implementation details, and usage instructions.
4. **Presentation:**
   * Prepare a presentation to showcase the project to peers or mentors, gathering feedback for improvement.

Decision Making Statements

Decision-making statements in Python allow the program to execute certain pieces of code based on specific conditions. The main decision-making statements in Python are:

1. **if statement**
2. **if-else statement**
3. **if-elif-else statement**
4. **nested if statement**

### 1. if Statement

The if statement evaluates a condition and executes the block of code within it if the condition is True.

|  |
| --- |
| X = 10  If x>5:  Print(“x is greater than 5”) |

### 2. if-else Statement

The if-else statement provides an alternative block of code that executes if the condition is False.

|  |
| --- |
| X = 10  If x>5:  Print(“x is greater than 5”)  Else  Print(“ x is less than 5”) |

### 3. if-elif-else Statement

The if-elif-else statement allows for multiple conditions to be checked sequentially. If a condition is True, the corresponding block of code executes, and the rest are skipped.

|  |
| --- |
| number = 10# You can change this value to test different casesif number > 0:print("The number is positive.")elif number < 0:print("The number is negative.") |

### 4. Nested if Statement

An if or else statement can contain another if or else statement. This is known as nesting.

|  |
| --- |
| if number >= 0:  if number == 0:  print("The number is zero.")  else: print("The number is positive.")  if number % 2 == 0:  print("The number is even.")  else: print("The number is odd.")  else: print("The number is negative.") |

**ACTIVITY LOG FOR THE SIXTH WEEK**

|  |  |  |  |
| --- | --- | --- | --- |
| **Day**  **&**  **Date** | **Brief description of the daily activity** | **Learning Outcome** | **Person InCharge**  **Signature** |
| Day – 1 |  |  |  |
| Day - 2 |  |  |  |
| Day – 3 |  |  |  |
| Day – 4 |  |  |  |
| Day – 5 |  |  |  |
| Day –6 |  |  |  |

**WEEKLY REPORT**

**WEEK – 6 (From Dt………..….. to Dt )**

**Objective of the Activity Done:**

**Detailed Report**

### CHAPTER 5: OUTCOMES DESCRIPTION

**Describe the work environment you have experienced** *(in terms of people interactions, facilities available and maintenance, clarity of job roles, protocols, procedures, processes, discipline, time management, harmonious relationships, socialization, mutual support and teamwork, motivation, space and ventilation, etc.)*

### DESCRIPTION OF WORK ENVIRONMENT:-

### People Interactions

* **Communication**: Open and transparent communication channels where employees feel comfortable sharing ideas and feedback.
* **Respect**: Mutual respect among colleagues, regardless of their position or role in the organization.
* **Team Spirit**: A strong sense of camaraderie and teamwork, with colleagues supporting one another to achieve common goals.

### Facilities Available and Maintenance

* **Workspace**: Well-maintained, ergonomic workspaces that cater to the needs of employees, including desks, chairs, and necessary equipment.
* **Amenities**: Availability of essential amenities such as clean restrooms, break rooms, and kitchen areas.
* **Technology**: Up-to-date technology and tools to facilitate smooth and efficient work processes.

### Clarity of Job Roles

* **Defined Roles**: Clear job descriptions and expectations for each role to avoid confusion and overlap.
* **Responsibilities**: Well-communicated responsibilities and accountability for tasks and projects.
* **Career Path**: Clear pathways for career progression and development within the organization.

### Discipline and Time Management

* **Punctuality**: Emphasis on punctuality and adherence to work schedules.
* **Accountability**: Systems in place to monitor and enforce adherence to deadlines and commitments.
* **Balance**: Encouragement of a healthy work-life balance to prevent burnout and maintain productivity.

### Mutual Support and Teamwork

* **Collaboration**: Encouragement of collaborative work environments where teamwork is prioritized.
* **Support Systems**: Availability of support systems such as mentorship programs and peer support networks.
* **Recognition**: Regular recognition and appreciation of individual and team contributions.

### PROJECT DOCUMENTATION

**ABSTRACT**

**(Title:** TravelAmaze: AI-Powered Personalized Travel Assistant)

This internship project explores the application of the Rasa framework in enhancing visitor engagement and customer service within the tourism industry. The project involved developing a chatbot prototype using Rasa to address common inquiries and provide personalized recommendations to tourists visiting a specific region.

The objectives included understanding the capabilities of Rasa for creating conversational interfaces, integrating with existing tourism databases to retrieve relevant information, and evaluating user interactions to improve conversational flows and user experience.

Methodologies included initial setup and configuration of Rasa, iterative development based on user feedback and testing. Evaluation criteria focused on accuracy of information provided, natural language understanding capabilities, and overall user satisfaction.

Key findings highlighted the effectiveness of Rasa in handling complex queries related to tourist attractions, accommodations, and local events, thereby enhancing the accessibility of information for visitors. Recommendations for future development include expanding the chatbot’s knowledge base, integrating additional APIs for real-time updates, and further refining natural language processing models to better understand colloquial tourist queries.

This internship project not only provided hands-on experience with Rasa framework implementation but also deepened insights into the potential of conversational AI in improving tourism services and visitor experiences. This abstract summarizes the project’s objectives, methodologies, findings, and recommendations derived from the internship.

**CHAPTER -1**

**Introduction to Rasa Framework**

**Introduction:-**

Rasa is an open-source framework for building conversational AI, including chatbots and virtual assistants. It provides tools for natural language understanding (NLU) and dialogue management, enabling developers to create intelligent, context-aware conversational agents

Rasa is particularly known for its flexibility and ability to be deployed on-premises, providing developers with control over data privacy and security.

**Why we choose Rasa:-**

1. **Open Source and Free**: Rasa is open source, allowing developers to use and modify it without licensing costs. This makes it an attractive option for startups and individual developers.
2. **Customizability**: Rasa provides a high level of customizability, enabling developers to tailor the NLU models, dialogue management, and actions to fit specific project requirements. This flexibility is crucial for creating unique and sophisticated conversational experiences.
3. **Control Over Data**: Since Rasa can be deployed on-premises, organizations have full control over their data. This is essential for projects where data
4. **Advanced NLU and Dialogue Management**: Rasa offers robust natural language understanding and dialogue management capabilities, allowing for the creation of intelligent, context-aware chatbots. It supports complex conversation flows and can handle multiple intents and entities.
5. **Interactive Learning**: Rasa X provides a user-friendly interface for reviewing and improving the chatbot's performance. It supports interactive learning, allowing developers to refine the bot's responses based on real conversations.

**Usage of Rasa Framework:-**

The Rasa framework offers several benefits that can significantly enhance a project or business operation. Here’s how Rasa can be beneficial in various ways:

1. **Improved Customer Experience**

* **24/7 Availability**: Chatbots built with Rasa can provide round-the-clock support, ensuring that customers can get help whenever they need it.
* **Instant Responses**: Users receive immediate responses to their inquiries, reducing wait times and improving satisfaction.
* **Personalized Interaction**: Rasa can tailor responses based on user history and preferences, creating a more personalized and engaging experience.

### 2. **Cost Efficiency**

* **Reduced Operational Costs**: By automating repetitive tasks and handling common queries, businesses can reduce the need for large support teams.
* **Scalable Solutions**: Chatbots can handle multiple conversations simultaneously, allowing businesses to scale their support without proportional increases in cost.

### 3. **Enhanced Productivity**

* **Automating Routine Tasks**: Rasa can handle mundane tasks such as appointment scheduling, order tracking, and FAQs, freeing up human agents for more complex issues.
* **Efficient Workflows**: Integrations with existing systems (like CRM, ticketing, and inventory management) streamline operations and improve workflow efficiency.

### 4. **Data-Driven Insights**

* **User Behavior Analysis**: By analyzing conversation data, businesses can gain insights into customer needs, preferences, and pain points.
* **Performance Monitoring**: Rasa X provides tools for monitoring and improving bot performance, helping businesses continuously refine their conversational agents.

### 5. **Customization and Flexibility**

* **Tailored Solutions**: Rasa allows for extensive customization, enabling businesses to create chatbots that align closely with their specific requirements and brand voice.

**Rasa components:-**

The Rasa framework is a powerful tool for building conversational AI, particularly chatbots, and includes several key components:

1. **Rasa NLU (Natural Language Understanding)**:
   * **Intent Recognition**: Identifies what the user wants to achieve (e.g., booking a hotel, finding a restaurant).
   * **Entity Extraction**: Extracts specific information from the user’s message (e.g., dates, names, locations).
2. **Rasa Core**:
   * **Dialogue Management**: Manages the flow of the conversation by predicting the next action based on the current state of the conversation.
   * **Policies**: Define the behavior of the bot. Examples include MemoizationPolicy, KerasPolicy, and RulePolicy.
   * **Actions**: Custom actions the bot can perform, such as querying a database or sending a message.
3. **Rasa SDK**:
   * Provides a framework for writing custom actions that interact with external systems, such as APIs, databases, or custom business logic.
4. **Training Data**:
   * **NLU Training Data**: Includes examples of user inputs with labeled intents and entities.
   * **Stories**: Annotated conversations that show the bot how to respond in different situations.

**Rasa framework Architecture:-**

The Rasa architecture is designed to build intelligent conversational agents using natural language understanding (NLU) and dialogue management (Core). Here's an overview of its architecture:

### 1. **User Interaction Layer**

* **Channels**: This layer includes various messaging platforms and interfaces where users interact with the chatbot (e.g., Facebook Messenger, Slack, custom web chat).

### 2. **Message Processing Layer**

* **Rasa NLU**:
  + **Intent Recognition**: Identifies the intent of the user's message.
  + **Entity Extraction**: Extracts key pieces of information from the user's message.

### 3. **Dialogue Management Layer**

* **Rasa Core**:
  + **Policies**: Rules and machine learning models that determine the next action the bot should take based on the current state of the conversation.
    - **Rule Policy**: Handles simple, rule-based actions.
    - **Memoization Policy**: Remembers previously seen conversations to predict the next action.
    - **TED Policy (Transformer Embedding Dialogue)**: Uses machine learning to predict actions based on dialogue history.
  + **Actions**: Specific operations the bot can perform, which can be predefined or custom.
    - **Custom Actions**: Executed using the Rasa SDK to perform complex operations such as querying a database or calling an external API.

### 4. **Conversation State and Data Management Layer**

* **Tracker Store**: Stores the conversation state, including user messages, bot actions, and slot values (key pieces of information collected during the conversation).
* **Domain**: Defines the intents, entities, slots, responses, and actions that the bot understands and can perform.
* **Training Data**:
  + **NLU Training Data**: Examples of user inputs with labeled intents and entities.
  + **Stories**: Annotated examples of conversations that show how the bot should respond in different scenarios.

### 5. **Integration and Deployment Layer**

* **Endpoints**: Configuration for connecting to external services, such as custom action servers, databases, and APIs.
* **Event Broker**: Facilitates real-time monitoring and logging of events within the conversation, enabling integration with external analytics or logging services.

### 6. **Development and Collaboration Tools**

* **Rasa X**: A tool for improving the assistant by reviewing conversations, managing training data, and testing models. It provides a user-friendly interface for non-technical team members to collaborate on the chatbot's development.

### CHAPTER – 2

### Definition of the TravelAmaze Project

TravelAmaze is an AI-tourism, chatbot-based travel assistant designed to revolutionize the way users plan and experience their journeys. Built on the Rasa framework and utilizing Tailwind CSS for a responsive and visually appealing interface, TravelAmaze provides personalized travel recommendations, real-time updates, and seamless itinerary planning. This innovative platform aims to enhance user satisfaction by delivering a comprehensive and interactive travel planning experience.

#### Who We Are

At TravelAmaze, we believe in the power of technology to transform the way people explore the world. Our AI-driven chatbot is at the core of our service, offering 24/7 assistance, personalized recommendations, and real-time support to travelers. Whether you are planning a solo adventure, a family vacation, or a business trip, our chatbot is designed to provide you with tailored solutions and expert guidance.

#### Our Mission

Our mission is to enhance the travel experience by offering innovative, reliable, and user-friendly services that meet the evolving needs of our customers. We strive to be a leader in the tourism industry by promoting sustainable travel practices and supporting local communities.

#### Our Vision

We envision a world where travel is accessible, enjoyable, and sustainable for everyone. By integrating technology with a deep understanding of travel trends and customer preferences, we aim to create a global community of satisfied and inspired travelers.

#### What We Offer

* **Personalized Travel Assistance**: Our chatbot provides instant answers to your travel queries, helping you plan and book your trips effortlessly.
* **Tailored Recommendations**: Based on your interests and preferences, our chatbot suggests destinations, activities, and accommodations that suit your needs.
* **24/7 Support**: No matter where you are or what time it is, our chatbot is always available to assist you with any travel-related issues.
* **Sustainable Travel Options**: We promote eco-friendly travel choices and partner with local businesses to support sustainable tourism practices.

#### Future Enhancements

* **Multilingual Support:** Expand language options to cater to a global audience.
* **Voice Interaction:** Implement voice recognition and response features for hands-free user experience.
* **Social Media Integration:** Allow users to share their itineraries and travel experiences directly on social media platforms.

#### What We Offer

* **Personalized Travel Assistance**: Our chatbot provides instant answers to your travel queries, helping you plan and book your trips effortlessly.
* **Tailored Recommendations**: Based on your interests and preferences, our chatbot suggests destinations, activities, and accommodations that suit your needs.
* **24/7 Support**: No matter where you are or what time it is, our chatbot is always available to assist you with any travel-related issues.
* **Sustainable Travel Options**: We promote eco-friendly travel choices and partner with local businesses to support sustainable tourism practices.

Organizational Structure for TravelAmaze

 **CTO (Chief Technology Officer)**

* + Leads technology strategy and oversees chatbot development.
* **Lead Chatbot Developer**
  + Heads the development and maintenance of the chatbot.
* **AI/ML Engineers**
  + Develop and enhance AI and machine learning models for the chatbot.
* **Backend Developers**
  + Handle server-side logic, database management, and system integration.
* **Frontend Developers**
  + Design and implement the user interface, integrating the chatbot on web and mobile platforms.

### Process of Creating Travel\_Amaze Chatbot:

### TravelAmaze is an AI-Tourism, chatbot-based travel assistant designed to revolutionize the way users plan and experience their journeys. Built on the Rasa framework and utilizing Tailwind CSS for a responsive and visually appealing interface, TravelAmaze provides personalized travel recommendations, real-time updates, and seamless itinerary planning.

### Step by step process of creating chatbot

### Step-1: Install Rasa

### First, ensure you have Python installed. Then, install Rasa.

|  |
| --- |
| pip install rasa |

### Step-2: Create a new rasa project

### Initialize a new Rasa project.

|  |
| --- |
| rasa init |

### Step-3: Define intents and entities

### Edit the “nlu.yml” file to define the intents and entities related to your tourism chatbot.

|  |
| --- |
| **nlu.yml:**  version: "3.1"  nlu:  - intent: greet  examples: |  - Hi  - Hello  - Hey  - Hi there  - Hey there  - Hello there  - hi  - intent: goodbye  examples: |  - Goodbye  - Bye  - See you later  - Bye bye  - Talk to you later  - intent: ask\_info  examples: |  - Tell me about [Eiffel Tower](tourist\_attraction)  - I want to know about [Statue of Liberty](tourist\_attraction)  - Give me information on [Colosseum](tourist\_attraction)  - What can you tell me about [Big Ben](tourist\_attraction)  - I need details about [Louvre Museum](tourist\_attraction)  - Can you provide info on [Taj Mahal](tourist\_attraction)  - [Goa](tourist\_attraction)  - [Manali](tourist\_attraction)  - [Ooty](tourist\_attraction)  - [Red fort](tourist\_attraction)  - [Tokyo](tourist\_attraction)  - [Osaka](tourist\_attraction)  - [Los vegas]  - [Las Angeles]  - I want to know about  - [Statue of Liberty](tourist\_attraction)  - intent: search\_place  examples: |  - Find a [restaurant](type) near [Central Park](location)  - Show me a [hotel](type) around [Times Square](location)  - I need a [cafe](type) close to [Union Square](location)  - Can you find a [gym](type) near [Wall Street](location)  - can you find a [restaurant](type) near [Central Prak](location)?  - i need a [cafe](type) close to [union square]{"entity": "location", "value": "Union Square"}  - show me a [hotel](type) around one town  - i need a [cafe](type) close to bsent road  - [Hotel]{"entity": "type", "value": "hotel"}  - intent: ask\_tourist\_spots  examples: |  - What are the tourist spots in [New York](location)?  - Tell me about tourist attractions in [Paris](location).  - Show me tourist spots in [Tokyo](location).  - What can I see in [Berlin](location)?  - What are the tourist spots in [New York](location)  - [New York](location)  - intent: ask\_question  examples: |  - [What are the top tourist attractions in New York?](question)  - [Can you recommend some good restaurants near Eiffel Tower?](question)  - [What's the best time to visit Eiffel Tower?](question)  - [How do I get to Statue of Liberty from New York Harber?](question)  - [Tell me about the weather forecast for Paris next week.](question)  - [What are some popular activities to do in Tokyo?](question)  - [Are there any special events or festivals happening in India soon?](question)  - [Could you suggest a hotel or accommodation option in Seoul?](question)  - [Do you have any information about Ajantha Caves?](question)  - [Can you provide tips for traveling to Louvre Museum?](question)  - What are the top tourist attractions in [New York](location)?  - Can you provide simple tips for traveling to Louvre Museum?  - Tell me about the weather forecast for Paris next week.  - What are some popular activities to do in Tokyo?  - Could you suggest a [hotel](type) or accommodation option in Seoul?  - synonym: Union Square  examples: |  - union square  - synonym: hotel  examples: |  - Hotel |

### Step-4: Create Stories

### Edit the stories.yml file to define the conversation flow.

|  |
| --- |
| **Stories.yml:**  version: "3.0"  stories:  - story: greet user  steps:  - intent: greet  - action: utter\_greet  - action: utter\_ask\_tourist\_attraction  - story: goodbye user  steps:  - intent: goodbye  - action: utter\_goodbye  - story: provide information via form  steps:  - intent: ask\_info  - action: tourist\_info\_form  - active\_loop: tourist\_info\_form  - active\_loop: null  - action: action\_tourist\_info  - story: ask for assistance  steps:  - intent: search\_place  - action: action\_search\_place  - story: fetch tourist spots  steps:  - intent: ask\_tourist\_spots  entities:  - location: "New York"  - action: action\_fetch\_tourist\_spots  - story: ask AI question  steps:  - intent: ask\_question  - action: action\_get\_ai\_response |

### Step-5: Define Responses

### Edit the domain.yml file to define responses and custom actions.

|  |
| --- |
| domain.yml:  version: '3.0'  intents:  - ask\_info  - ask\_question  - ask\_tourist\_spots  - goodbye  - greet  - search\_place  entities:  - type  - location  - tourist\_attraction  slots:  tourist\_attraction:  type: text  influence\_conversation: true  mappings:  - type: from\_entity  entity: tourist\_attraction  location:  type: text  influence\_conversation: true  mappings:  - type: from\_entity  entity: location  type:  type: text  influence\_conversation: true  mappings:  - type: from\_entity  entity: type  responses:  utter\_greet:  - text: Hello! How can I help you today?  utter\_goodbye:  - text: Goodbye! Have a great day!  utter\_ask\_tourist\_attraction:  - text: Which place would you like to know about?  actions:  - action\_get\_ai\_response  - utter\_greet  - action\_fetch\_tourist\_spots  - action\_search\_place  - utter\_ask\_tourist\_attraction  - action\_tourist\_info  forms:  tourist\_info\_form:  required\_slots:  - tourist\_attraction  session\_config:  session\_expiration\_time: 60  carry\_over\_slots\_to\_new\_session: true |

### Step-6: Implement Custom actions

### Create a actions.py file to define custom actions. In this file, you'll use the Wikimedia API to fetch tourist information.

|  |
| --- |
| actions.py:  from typing import Any, Text, Dict, List  from rasa\_sdk import Action, Tracker  from rasa\_sdk.executor import CollectingDispatcher  from rasa\_sdk.events import SlotSet  from rasa\_sdk.forms import FormValidationAction  import http.client  import json  import requests  class ActionTouristInfo(Action):  def name(self) -> Text:  return "action\_tourist\_info"  def run(self, dispatcher: CollectingDispatcher,  tracker: Tracker,  domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:  tourist\_attraction = tracker.get\_slot('tourist\_attraction')  if tourist\_attraction:  response = requests.get(f"https://en.wikipedia.org/api/rest\_v1/page/summary/{tourist\_attraction}")  data = response.json()  if response.status\_code == 200:  description = data.get('extract', 'Sorry, I could not find any information on that.')  dispatcher.utter\_message(text=description)  else:  dispatcher.utter\_message(text="Sorry, I couldn't fetch information at the moment.")  else:  dispatcher.utter\_message(text="Could you please specify the tourist attraction you want to know about?")  return []  class ValidateTouristInfoForm(FormValidationAction):  def name(self) -> Text:  return "validate\_tourist\_info\_form"  def validate\_tourist\_attraction(self,  slot\_value: Any,  dispatcher: CollectingDispatcher,  tracker: Tracker,  domain: Dict[Text, Any]) -> Dict[Text, Any]:  if slot\_value:  return {"tourist\_attraction": slot\_value}  else:  dispatcher.utter\_message(text="I didn't understand that. Can you please specify the tourist attraction again?")  return {"tourist\_attraction": None}  class ActionSearchPlace(Action):  def name(self) -> str:  return "action\_search\_place"  def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:  place\_type = tracker.get\_slot("type")  location = tracker.get\_slot("location")  api\_key = "AIzaSyAGCdA9HLhndXlo6rLkOmpr3n9aMw8X4AQ"  url = f"https://maps.googleapis.com/maps/api/place/textsearch/json?query={place\_type}+in+{location}&key={api\_key}"  response = requests.get(url)  results = response.json().get("results", [])  if results:  names = [result["name"] for result in results[:5]]  response\_text = "Here are the top results: \n" + "\n".join([f"{i+1}. {name}" for i, name in enumerate(names)])  else:  response\_text = "I couldn't find any results for your query."  dispatcher.utter\_message(text=response\_text)  return []  class ActionFetchTouristSpots(Action):  def name(self) -> Text:  return "action\_fetch\_tourist\_spots"  def run(self, dispatcher: CollectingDispatcher,  tracker: Tracker,  domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:  # Extract location from the latest user message  entities = tracker.latest\_message['entities']  location = None  for entity in entities:  if entity['entity'] == 'location':  location = entity['value']  break  if not location:  dispatcher.utter\_message(text="Sorry, I couldn't understand the location.")  return []  # Use a geocoding API to convert location to coordinates (latitude and longitude)  geocoding\_api\_key = "c45c2b93c8b84c5da58a9d9197da92ff" # Replace with your OpenCage API key  geocoding\_response = requests.get(f"https://api.opencagedata.com/geocode/v1/json?q={location}&key={geocoding\_api\_key}")  if geocoding\_response.status\_code == 200:  geocoding\_data = geocoding\_response.json()  if geocoding\_data['results']:  coordinates = geocoding\_data['results'][0]['geometry']  latitude = coordinates['lat']  longitude = coordinates['lng']  radius = 10000 # 10 km radius  limit = 10 # Limit to 10 results  # Call Wikimedia API to fetch tourist spots  response = requests.get(f"https://en.wikipedia.org/w/api.php?action=query&list=geosearch&gscoord={latitude}|{longitude}&gsradius={radius}&gslimit={limit}&format=json")  if response.status\_code == 200:  data = response.json()  places = data.get("query", {}).get("geosearch", [])  if places:  spots = [place["title"] for place in places]  spots\_list = "\n".join(spots)  dispatcher.utter\_message(text=f"Here are some tourist spots in {location}:\n{spots\_list}")  else:  dispatcher.utter\_message(text=f"Sorry, I couldn't find any tourist spots in {location}.")  else:  dispatcher.utter\_message(text="Sorry, I couldn't fetch tourist spots at the moment. Please try again later.")  else:  dispatcher.utter\_message(text=f"Sorry, I couldn't find the coordinates for {location}.")  else:  dispatcher.utter\_message(text="Sorry, I couldn't fetch the coordinates at the moment. Please try again later.")  return []  class ActionGetAiResponse(Action):  def name(self) -> Text:  return "action\_get\_ai\_response"  def run(self, dispatcher: CollectingDispatcher,  tracker: Tracker,  domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:  # Get the user's question from the latest message  user\_question = tracker.latest\_message['text']  # Set up the connection and headers for the API call  conn = http.client.HTTPSConnection("chatgpt-42.p.rapidapi.com")  payload = json.dumps({  "messages": [{"role": "user", "content": user\_question}],  "system\_prompt": "",  "temperature": 0.9,  "top\_k": 5,  "top\_p": 0.9,  "image": "",  "max\_tokens": 256  })  headers = {  'x-rapidapi-key': "6c163ad2camsh8b065d6c196f570p19c333jsn61739cf3fb36",  'x-rapidapi-host': "chatgpt-42.p.rapidapi.com",  'Content-Type': "application/json"  }  # Make the API request  conn.request("POST", "/matag2", payload, headers)  res = conn.getresponse()  data = res.read()  # Print the raw response for debugging  print("Raw API response:", data.decode("utf-8"))  # Parse the JSON response  response\_data = json.loads(data.decode("utf-8"))  # Print the parsed response for debugging  print("Parsed API response:", response\_data)  # Extract the AI response  ai\_response = response\_data.get("result", "I'm sorry, I couldn't find an answer to your question.")  # Send the AI response back to the user  dispatcher.utter\_message(text=ai\_response)  return [] |

### Step-7: Set up the pipeline

### Update the config.yml by uncommenting the data

### Step-8: Set up the localhost by uncommenting the endpoint.yml

### Step-9: Train the model

### Train your Rasa model with the following command:

|  |
| --- |
| rasa train |

### Step-10: Run the action server:

### Start the actions server to handle custom actions

### .

|  |
| --- |
| Rasa run actions |

### Step-11: Run the Rasa Server

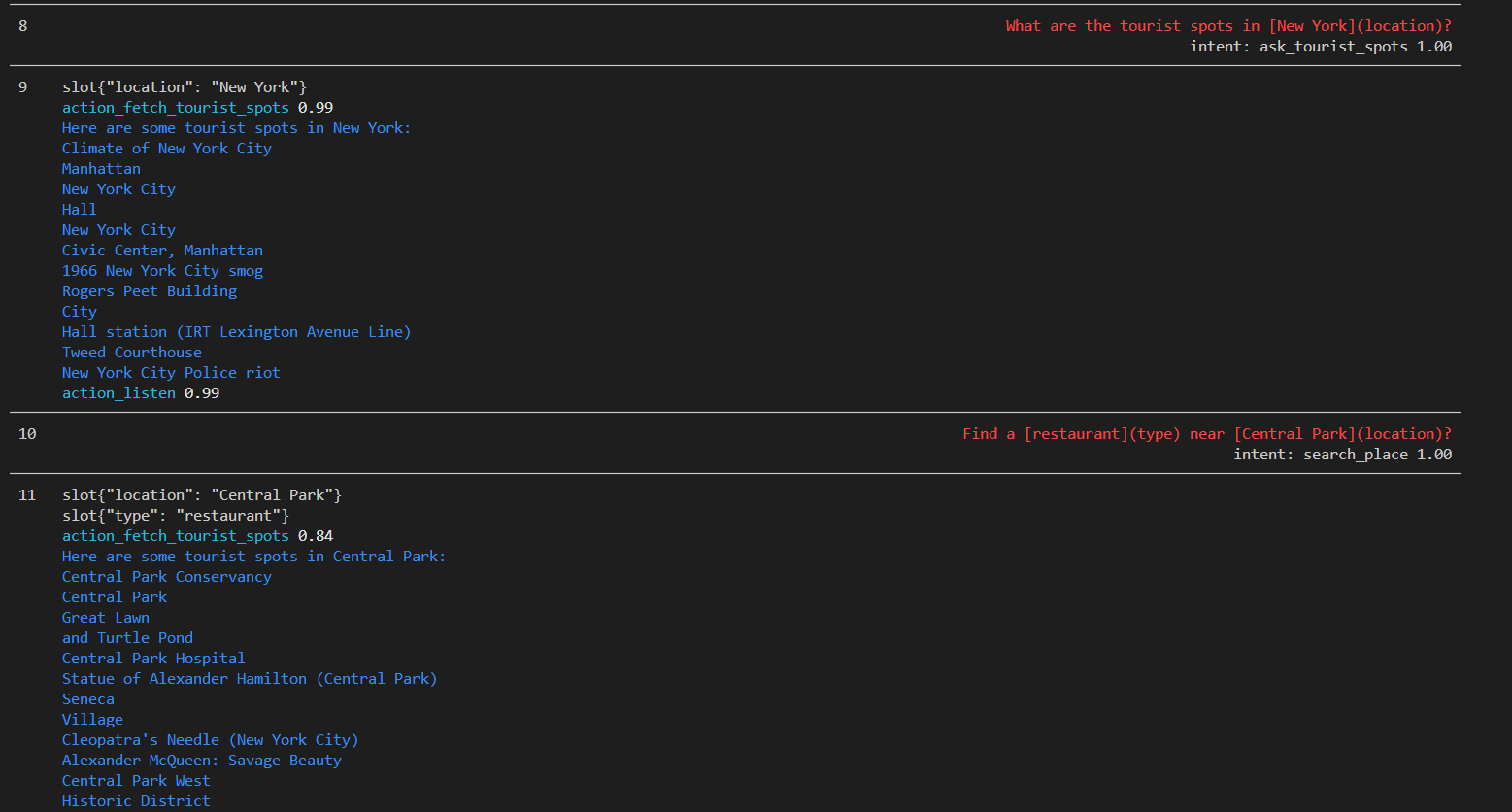
### Start the rasa server

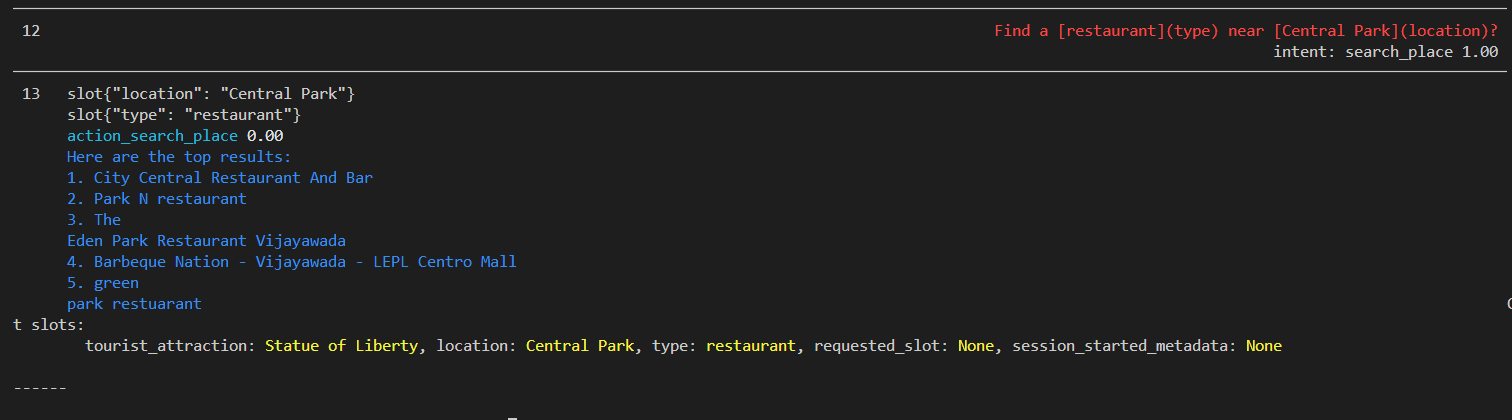
|  |
| --- |
| rasa shell |

### Step-12: Test your Bot

### Interact with your bot through the Rasa shell or integrate it with a messaging platform to test its responses.

### Outputs of the Travel\_Amaze Chatbot project:





### *Student Self Evaluation of the Short-Term Internship*

**Student Name: Sk. Hafiza**

**Registration No: K2221202**

**Term of Internship:**

**From: 30-05-2024 To : 05-06-2024**

**Date of Evaluation: 05-06-2024**

**Organization Name & Address: Fresher Bot Technology**

**Please rate your performance in the following areas:**

**Rating Scale: Letter grade of CGPA calculation to be provided**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 Oral communication | 1 | 2 | 3 | 4 | 5 |
| 2 Written communication | 1 | 2 | 3 | 4 | 5 |
| 3 Proactiveness | 1 | 2 | 3 | 4 | 5 |
| 4 Interaction ability with community | 1 | 2 | 3 | 4 | 5 |
| 5 Positive Attitude | 1 | 2 | 3 | 4 | 5 |
| 6 Self-confidence | 1 | 2 | 3 | 4 | 5 |
| 7 Ability to learn | 1 | 2 | 3 | 4 | 5 |
| 8 Work Plan and organization | 1 | 2 | 3 | 4 | 5 |
| 9 Professionalism | 1 | 2 | 3 | 4 | 5 |
| 10 Creativity | 1 | 2 | 3 | 4 | 5 |
| 11 Quality of work done | 1 | 2 | 3 | 4 | 5 |
| 12 Time Management | 1 | 2 | 3 | 4 | 5 |
| 13 Understanding the Community | 1 | 2 | 3 | 4 | 5 |
| 14 Achievement of Desired Outcomes | 1 | 2 | 3 | 4 | 5 |
| **15 OVERALL PERFORMANCE** | **1** | **2** | **3** | **4** | **5** |

**Date: Signature of the Student**

**Evaluation by the Supervisor of the Intern Organization**

|  |
| --- |
| **Student Name: Sk. Hafiza Registration No: k2221502**    **Term of Internship:**  **From: 30-05-2024 To : 05-06-2024**    **Date of Evaluation: 05-07-2024**    **Organization Name & Address: Fresher Bot Technology**    **Name & Address of the Supervisor with Mobile Number** |

Please rate the student’s performance in the following areas:

Please note that your evaluation shall be done independent of the Student’s

selfevaluation

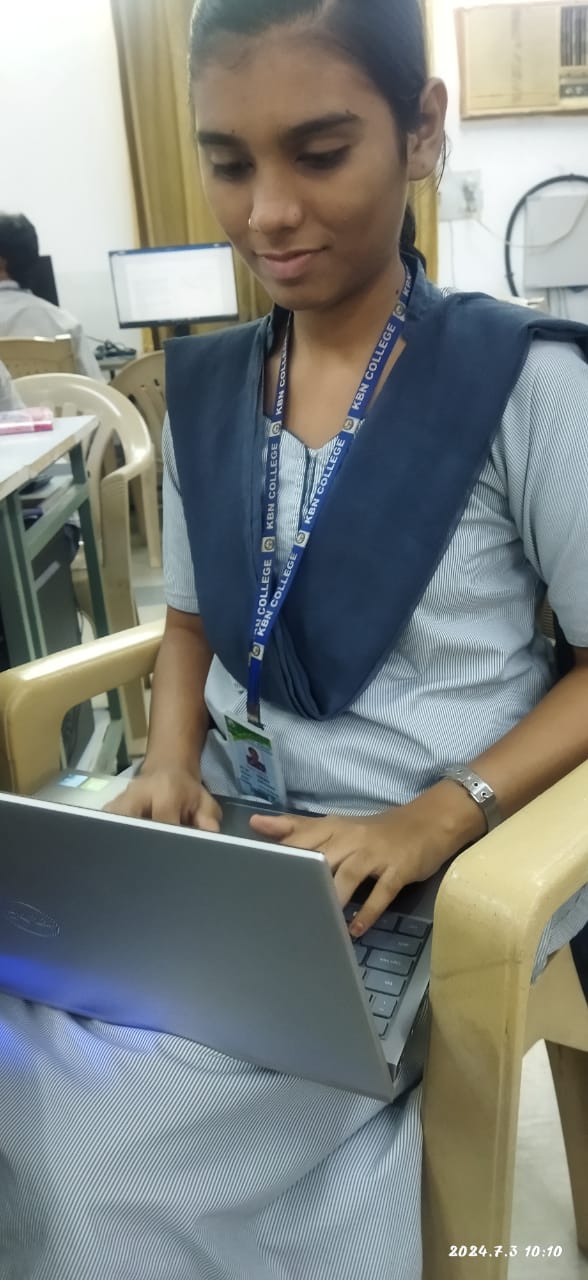
Rating Scale: 1 is lowest and 5 is highest rank

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 Oral communication | 1 | 2 | 3 | 4 | 5 |
| 2 Written communication | 1 | 2 | 3 | 4 | 5 |
| 3 Proactiveness | 1 | 2 | 3 | 4 | 5 |
| 4 Interaction ability with community | 1 | 2 | 3 | 4 | 5 |
| 5 Positive Attitude | 1 | 2 | 3 | 4 | 5 |
| 6 Self-confidence | 1 | 2 | 3 | 4 | 5 |
| 7 Ability to learn | 1 | 2 | 3 | 4 | 5 |
| 8 Work Plan and organization | 1 | 2 | 3 | 4 | 5 |
| 9 Professionalism | 1 | 2 | 3 | 4 | 5 |
| 10 Creativity | 1 | 2 | 3 | 4 | 5 |
| 11 Quality of work done | 1 | 2 | 3 | 4 | 5 |
| 12 Time Management | 1 | 2 | 3 | 4 | 5 |
| 13 Understanding the Community | 1 | 2 | 3 | 4 | 5 |
| 14 Achievement of Desired Outcomes | 1 | 2 | 3 | 4 | 5 |
| **15 OVERALL PERFORMANCE** | **1** | **2** | **3** | **4** | **5** |

**Date: Signature of the Supervisor**

**PHOTOS & VIDEO LINKS**







VIDEO LINK:

EVALUATION

**Internal Evaluation for Short Term Internship**

## (On-site/Virtual)

**Objectives:**

* To integrate theory and practice.
* To learn to appreciate work and its function towards the future.
* To develop work habits and attitudes necessary for job success.
* To develop communication, interpersonal and other critical skills in the future job.
* To acquire additional skills required for the world of work.

**Assessment Model:**

* There shall only be internal evaluation.
* The Faculty Guide assigned is in-charge of the learning activities of the students and for the comprehensive and continuous assessment of the students.
* The assessment is to be conducted for 100 marks.
* The number of credits assigned is 4. Later the marks shall be converted into grades and grade points to include finally in the SGPA and CGPA.
* The weightings shall be:

o Activity Log 25 marks o Internship Evaluation 50marks o Oral Presentation 25 marks

* Activity Log is the record of the day-to-day activities. The Activity Log is assessed on an individual basis, thus allowing for individual members within groups to be assessed this way. The assessment will take into consideration the individual student’s involvement in the assigned work.
* While evaluating the student’s Activity Log, the following shall be considered –
  1. The individual student’s effort and commitment.
  2. The originality and quality of the work produced by the individual student.
  3. The student’s integration and co-operation with the work assigned.
  4. The completeness of the Activity Log.
* The Internship Evaluation shall include the following components and based on Weekly Reports and Outcomes Description a. Description of the Work Environment.
  1. Real Time Technical Skills acquired.
  2. Managerial Skills acquired.
  3. Improvement of Communication Skills.
  4. Team Dynamics
  5. Technological Developments recorded.

**MARKS STATEMENT**

**(To be used by the Examiners)**

**INTERNAL ASSESSMENT STATEMENT**

**Name Of the Student:** Sk. Hafiza

**Programme of Study:** Prompt Engineering

**Year of Study:** 2022-2025

**Group:** Bsc Datacience

**Register No/H.T. No: Name of the College:** K.B.N College

**University:** Krishna University

|  |  |  |  |
| --- | --- | --- | --- |
| ***Sl.No*** | ***Evaluation Criterion*** | ***Maximum Marks*** | ***Marks Awarded*** |
| 1. | Activity Log | 25 |  |
| 2. | Internship Evaluation | 50 |  |
| 3. | Oral Presentation | 25 |  |
|  | GRAND TOTAL | 100 |  |

Date: **Signature of the Faculty Guide**

### Certified by

Date:  **Signature of the Head of the Department/Principal**

Seal:

