SILVERSCREENAI:EMPOWERING MOVIE CONVERSATION WITH RASA NLU

1.Introduction

1.1Project Overview

This project overview delves into the key features of Silver Screen AI, the critical factors influencing its functionality, and the methodologies employed to create seamless and meaningful interactions. By enabling personalized recommendations, contextual movie discussions, and real-time sentiment analysis, Silver Screen AI enhances the way users discover and engage with films.

Movies have the power to captivate audiences, spark conversations, and create shared experiences. With the rise of AI-powered chatbots, providing engaging and intelligent movie-related conversations has become increasingly feasible. Silver Screen AI aims to revolutionize how users interact with movie databases by leveraging Rasa NLU for natural language understanding..

1.2 Project Objective

The primary objective of this project is to develop a comprehensive AI-driven chatbot powered by Rasa NLU that facilitates intelligent and engaging movie-related conversations. This system aims to assist users in discovering movies, exploring genres, and gaining insights into films through personalized recommendations and natural language interactions. By understanding user preferences and intents, the chatbot seeks to enhance user engagement, streamline movie exploration, and redefine the cinematic experience with AI-driven innovations.

2. Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Activity 1: Define Problem Statement

A movie enthusiast seeks a seamless way to explore and discover films that match their preferences. The challenge lies in understanding and accurately interpreting user queries, which may vary in complexity, tone, and intent. Despite the availability of vast movie databases, users often struggle to find personalized recommendations or meaningful insights due to the lack of intelligent conversational systems. The goal is to bridge this gap by developing an AI-powered chatbot that leverages natural language understanding to provide engaging, accurate, and context-aware movie-related conversations.

Ref. template:
Silverscreen AI project Problem Statement Report: CLICK HERE

Activity 2: Project Proposal (Proposed Solution)

Our innovative proposed solution harnesses the power of Rasa NLU and advanced natural language processing techniques to create an intelligent chatbot designed to engage users in meaningful movie-related conversations. By integrating features like personalized recommendations, intent detection, and contextual understanding, this solution aims to transform the movie discovery experience. It empowers users to explore films effortlessly, enhances engagement through dynamic interactions, and provides valuable insights into cinematic content

Ref. template:
Silverscreen AI project Proposal Report:CLICK HERE

Activity 3: Initial Project Planning

Initial Project Planning involves outlining key objectives, defining the scope, and identifying the core functionalities of the chatbot. It encompasses setting timelines, allocating resources, and determining the overall project strategy. During this phase, the team establishes a clear understanding of user requirements, formulates goals for natural language understanding, and plans the workflow for data integration and conversational design.

Ref. template:
Silverscreen AI project Initial Project Planning Report:CLICK HERE

3. Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant movie-related data from sources such as public APIs, databases, and user interactions. Ensuring data quality is achieved through validation, addressing inconsistencies, and handling missing values. Preprocessing tasks include cleaning text data, tokenization, encoding categorical features, and organizing the dataset for subsequent exploratory analysis and integration with natural language understanding (NLU) models.

Activity 1: Data Collection Plan, Raw Data Sources Identified

The dataset for Silver Screen AI is sourced from a combination of public movie databases, such as The Movie Database (TMDb) and Open Movie Database (OMDb), as well as user interaction data. These sources provide a rich collection of movie-related information, including movie titles, genres, cast, plot summaries, ratings, and user reviews. The dataset is meticulously curated to encompass a wide array of features crucial for building an intelligent movie recommendation system. These features include movie metadata, user preferences, sentiment data, and historical user interactions. This comprehensive dataset serves as a robust foundation for developing personalized movie recommendations and enhancing conversational capabilities through natural language understanding.

Ref. template:
Silverscreen AI project Raw Data Sources Report: CLICK HERE

Activity 2: Data Quality Report

Data quality is ensured through verification, addressing missing values, and handling inconsistencies or outliers. This process establishes a reliable foundation for predictive modeling, enabling the AI system to provide accurate movie recommendations, understand user queries, and engage in meaningful, context-aware conversations.

Ref. template:
Silverscreen AI project Data Quality Report: CLICK HERE

Activity 3: Data Exploration and Preprocessing

Data Exploration involves analyzing the Silver Screen AI dataset to identify patterns, trends, and relationships between movie features such as genres, ratings, and user preferences. This phase includes examining the distribution of movie genres, user ratings, and the frequency of certain keywords in plot summaries. Preprocessing tasks like handling missing values, scaling numerical data, and encoding categorical variables such as genres and actors are performed to prepare the data for analysis.

4. Model Development Phase

The Model Development Phase entails crafting an intelligent conversational model for Silver Screen AI. It encompasses strategic feature selection, evaluating and selecting models (such as decision trees, random forests, or neural networks), and initiating training with code. The process includes defining key features like user preferences, movie metadata, and historical interactions to build a robust recommendation engine. Rigorous validation and assessment of the model's performance, through methods like cross-validation and accuracy metrics, ensures that the system delivers accurate, personalized movie recommendations and engaging interactions for users.

Activity 1: Feature Selection Report

The Feature Selection Report for Silver Screen AI outlines the rationale behind selecting specific features for the movie recommendation model. It evaluates the relevance, importance, and impact of each feature, such as movie metadata (genres, ratings, cast), user preferences, and interaction history, on the model's ability to deliver accurate and personalized recommendations.

Ref. template:
Silverscreen AI project Feature Selection Report:CLICK HERE

Activity 2: Model Selection Report

The Model Selection Report for Silver Screen AI details the rationale behind choosing specific models for the movie recommendation and conversational AI system, such as decision trees, random forests, and neural networks. It considers each model's strengths in handling complex relationships, interpretability, adaptability, and overall performance in understanding user intent, providing personalized recommendations, and engaging in natural conversations.

Activity 3: Initial Model Training Code, Model Validation and Evaluation Report

The Initial Model Training Code for Silver Screen AI employs selected algorithms, such as decision trees, random forests, or neural networks, on the movie dataset, laying the foundation for recommendation system development. The subsequent Model Validation and Evaluation Report rigorously assesses the model's performance using metrics like precision, recall, F1 score, and accuracy to ensure the chatbot's ability to provide relevant and accurate movie recommendations.

Ref. template:
Silverscreen AI Model Development Phase Template: CLICK HERE

5. Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase for Silver Screen AI involves refining the AI model for optimal performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection to enhance the recommendation system's accuracy and efficiency.

Activity 1: Hyperparameter Tuning Documentation

Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with the project's objectives of delivering relevant and personalized movie recommendations. This justifies its selection as the final model for the Silver Screen AI movie recommendation system.

Activity 2: Performance Metrics Comparison Report

The Performance Metrics Comparison Report contrasts the baseline and optimized metrics for various models used in the movie recommendation system, specifically highlighting the enhanced performance of the Random Forest model. This assessment provides a clear understanding of how hyperparameter tuning improved recommendation relevance, user engagement, and the overall efficiency of the AI-driven conversations, leading to more accurate responses and recommendations.

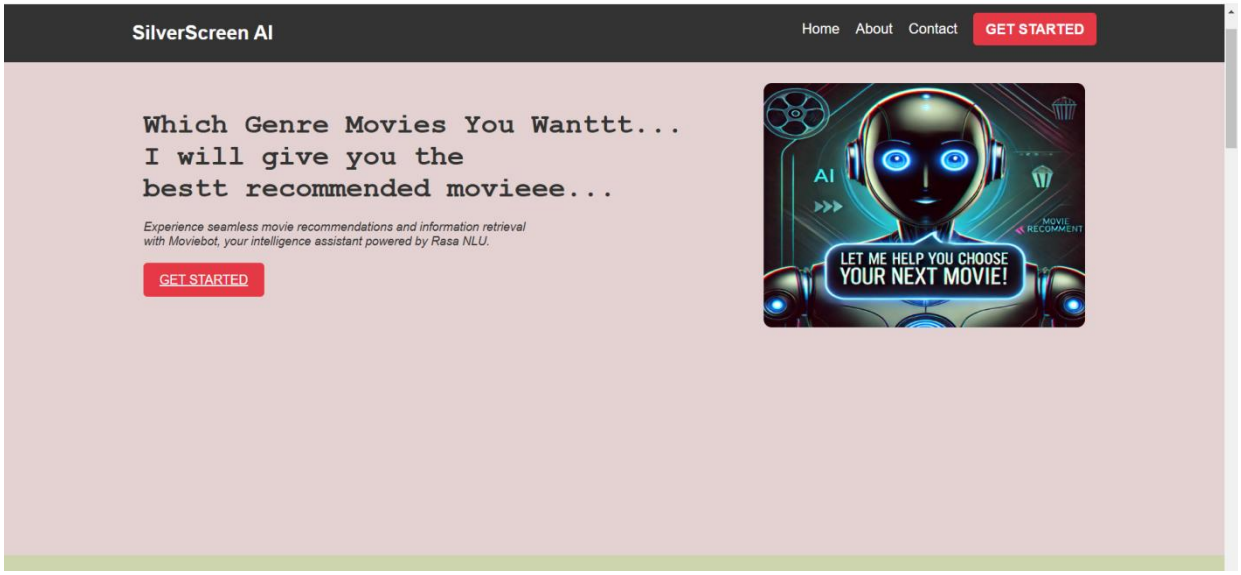
Activity 3: Final Model Selection Justification

The Final Model Selection Justification articulates the rationale for choosing Random Forest as the ultimate model for the Silver Screen AI system. Its exceptional accuracy in understanding user preferences, ability to handle complex relationships in movie metadata and user interactions, and successful hyperparameter tuning align with project objectives.

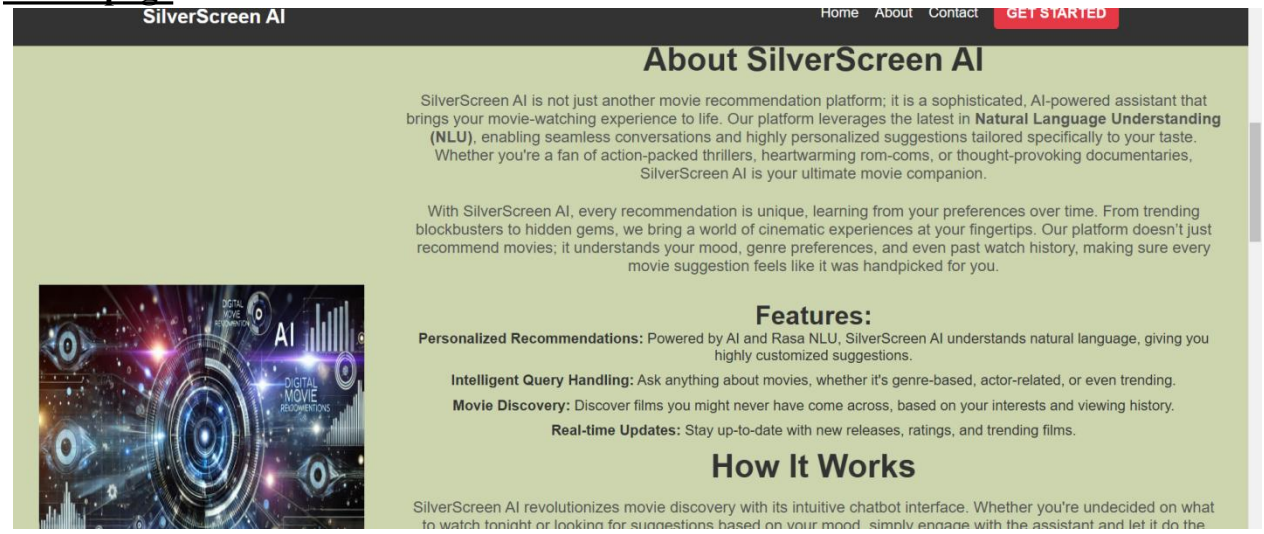
Ref. template:
Silverscreen AI Optimization and Tuning Phase Report:CLICK HERE

6.RESULTS

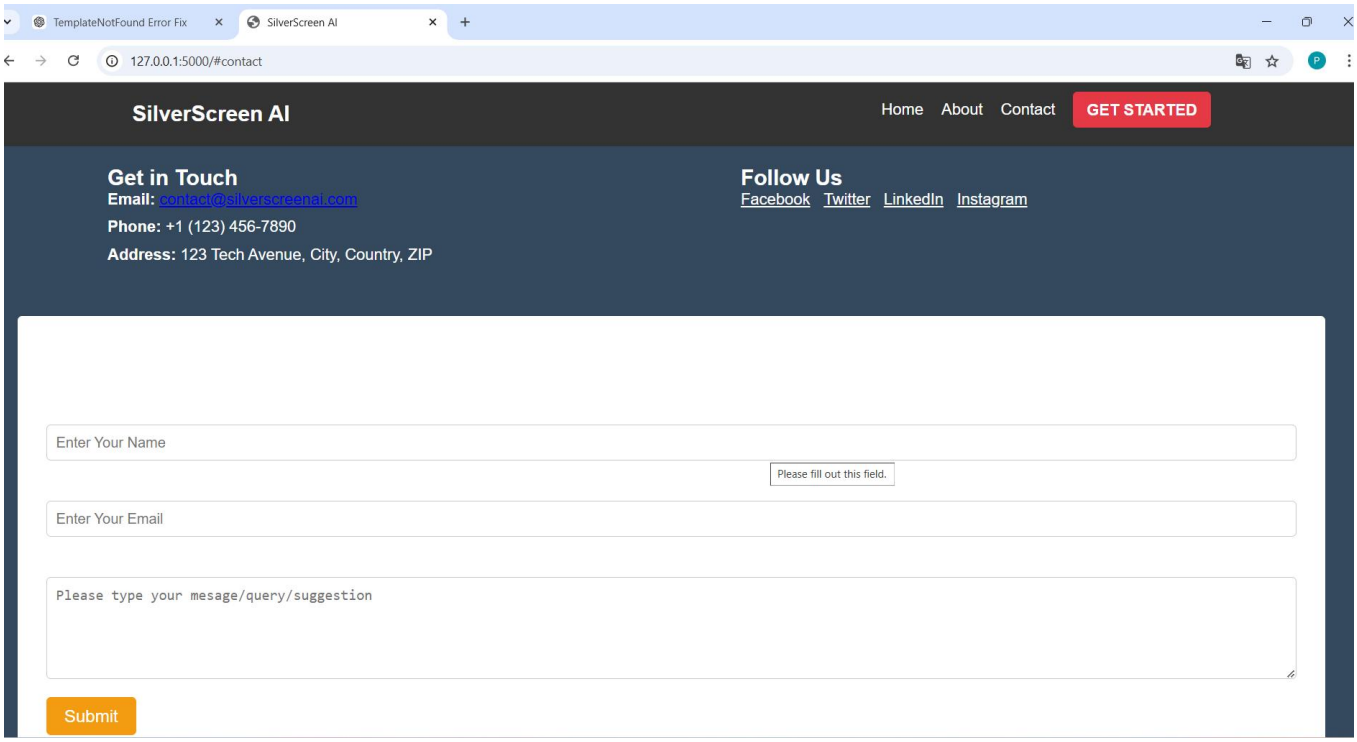
Index Page



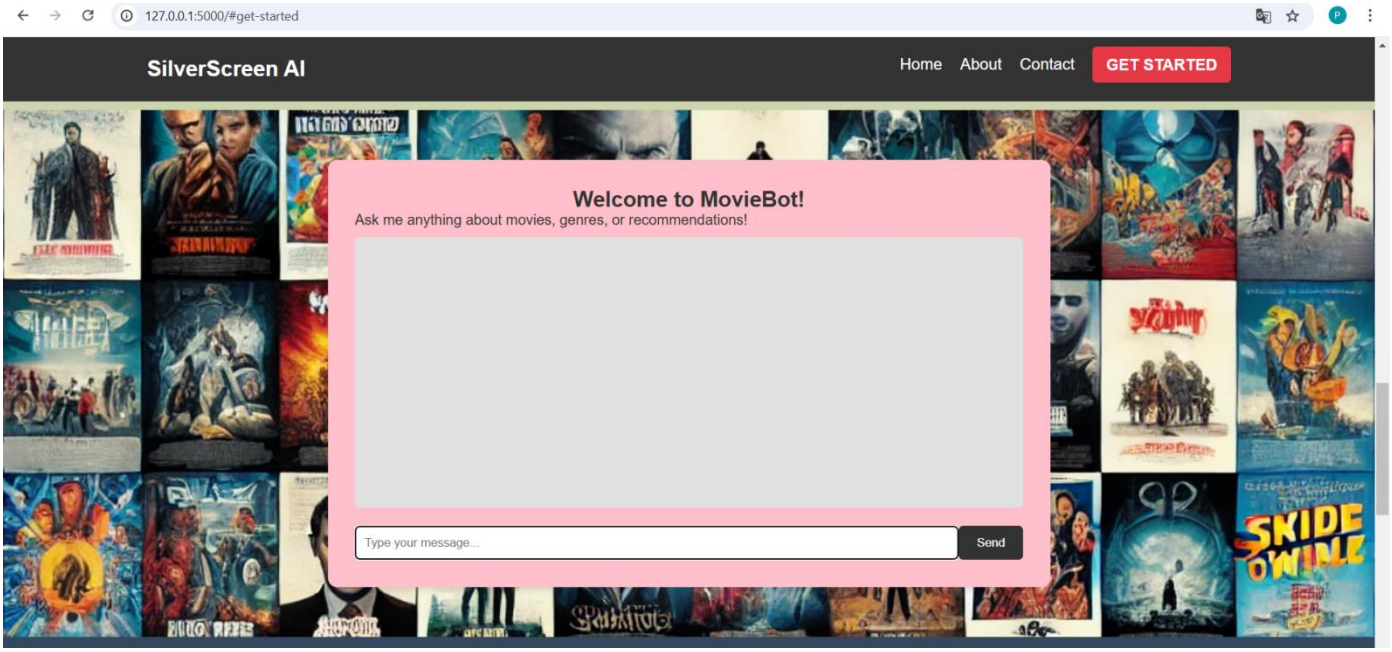
About page

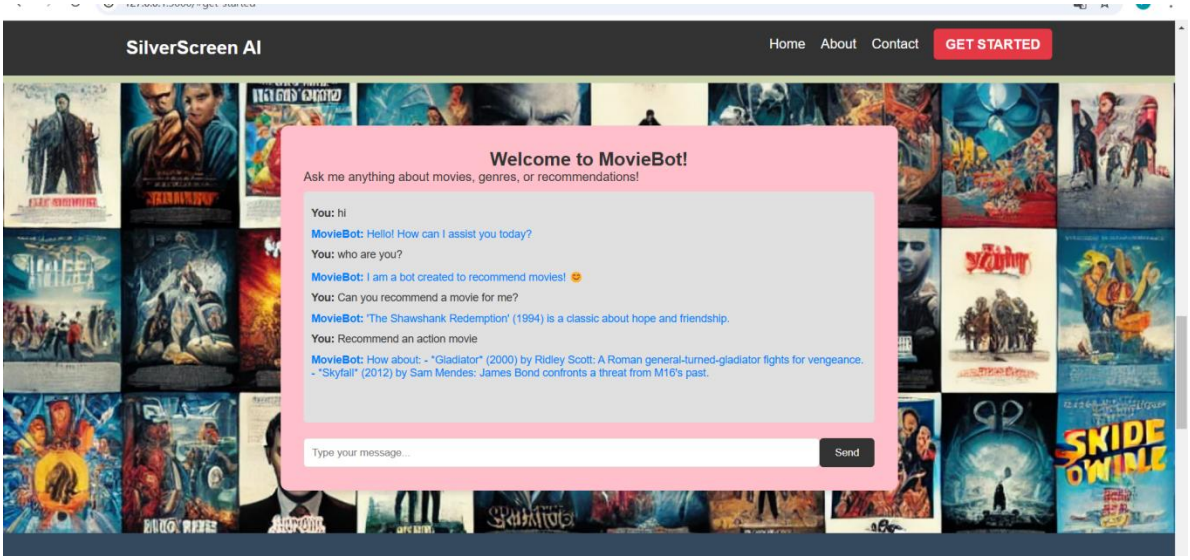


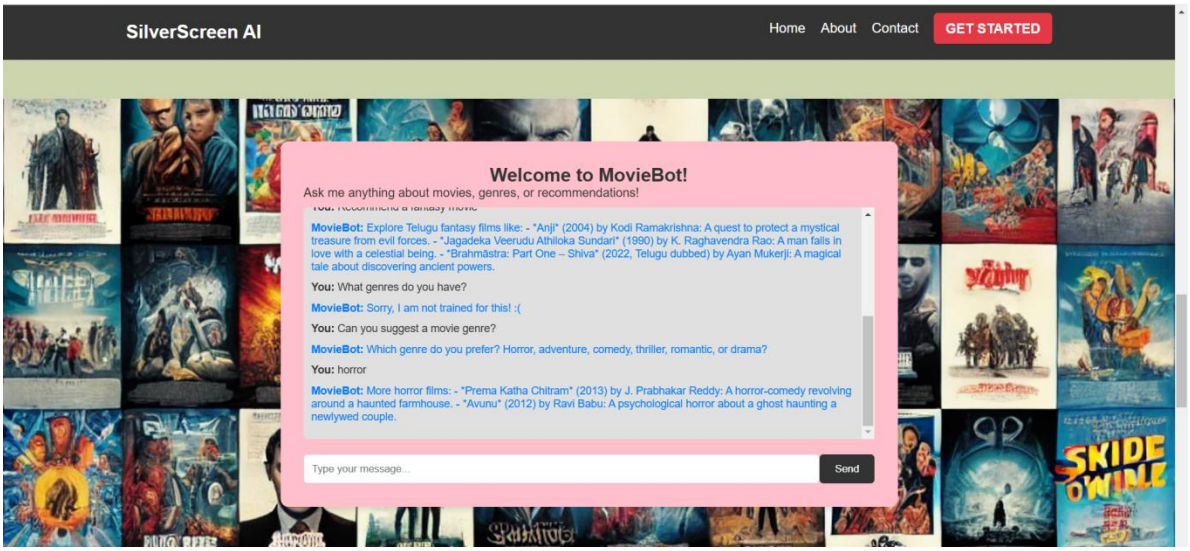
Contact page:



ASKME PAGE:




SilverScreen AI
Home
About
Contact
GET STARTED
Welcome to MovieBot!
Ask me anything about movies, genres, or recommendations!
You: hi
MovieBot: Hello! How can I assist you today?
You: who are you?
MovieBot: I am a bot created to recommend movies!
You: Can you recommend a movie for me?
MovieBot: 'The Shawshank Redemption' (1994) is a classic about hope and friendship.
You: Recommend an action movie
MovieBot: How about: - "Gladiator" (2000) by Ridley Scott: A Roman general-turned-gladiator fights for vengeance. - "Skyfall" (2012) by Sam Mendes: James Bond confronts a threat from M16's past.
Type your message...
Send


SilverScreen AI
Home
About
Contact
GET STARTED
Welcome to MovieBot!
Ask me anything about movies, genres, or recommendations!
MovieBot: Explore Telugu fantasy films like: - "Anji" (2004) by Kodi Ramakrishna: A quest to protect a mystical treasure from evil forces. - "Jagadeka Veerudu Athiloka Sundari" (1990) by K. Raghavendra Rao: A man falls in love with a celestial being. - "Brahmāstra: Part One – Shiva" (2022, Telugu dubbed) by Ayan Mukerji: A magical tale about discovering ancient powers.
You: What genres do you have?
MovieBot: Sorry, I am not trained for this! :(
You: Can you suggest a movie genre?
MovieBot: Which genre do you prefer? Horror, adventure, comedy, thriller, romantic, or drama?
You: horror
MovieBot: More horror films: - "Prema Katha Chitram" (2013) by J. Prabhakar Reddy: A horror-comedy revolving around a haunted farmhouse. - "Avunu" (2012) by Ravi Babu: A psychological horror about a ghost haunting a newlywed couple.
Type your message...
Send

7.Advantages & Disadvantages

ADVANTAGES:

1. Enhanced Movie Discussions

- Offers an enriched experience by providing instant, accurate information about movies, genres, actors, directors, reviews, and trivia.
- Allows users to ask nuanced questions about plots, awards, or recommendations based on personal preferences.

2. Personalization

- Using Rasa NLU's intent recognition, the chatbot can tailor recommendations to users' tastes, such as genre preferences or favorite actors.
- Provides personalized watchlists or alerts for upcoming movie releases.

3. Conversational Simplicity

- User-friendly interactions powered by Rasa's capabilities make it accessible for both casual users and movie enthusiasts.
- Supports multi-turn conversations and adapts to user input seamlessly.

4. Real-Time Updates

- Leverages integrations to fetch real-time data about current box office trends, newly released films, or streaming platform availability.
- Keeps users informed about trailers, premiere dates, and events like award shows.

5. Language and Sentiment Analysis

- Rasa NLU can process multilingual input, making the platform globally accessible.
- Detects sentiment to adapt responses—for example, consoling users disappointed by a movie or hyping upcoming releases.

DISADVANTAGES:

1. Complexity in Natural Language Understanding

- **Challenge:** Movies involve diverse topics, such as genres, cultures, and industry-specific terminology. Accurately interpreting user intent, especially for vague or complex queries (e.g., “That sci-fi movie with the blue aliens”), can be difficult.
- **Example:** Misinterpreting a user's input can lead to irrelevant or unsatisfactory recommendations.

2. Data Variability and Inconsistency

- **Challenge:** Movie databases and APIs like TMDb or IMDb may have incomplete or inconsistent data. This can result in the chatbot providing outdated or inaccurate information.
- **Example:** A movie's release date or cast information may differ between platforms, creating confusion for users.

3. Market Dynamics and Licensing Issues

- **Challenge:** The availability of movies on streaming platforms changes frequently due to licensing agreements. Keeping the chatbot's recommendations accurate in real-time can be resource-intensive.
- **Example:** A user might be recommended a movie that is no longer available in their region.

4. Cultural and Regional Sensitivities

- **Challenge:** Movies and their themes often carry cultural nuances. Recommending movies without considering a user's cultural or regional context could lead to dissatisfaction or offense.
- **Example:** A horror movie with religious overtones might not be appropriate for certain audiences.

5. High Dependency on External APIs

- **Challenge:** Relying on third-party APIs for real-time data exposes the system to risks like service outages, latency, or API changes.
- **Example:** If TMDb API is down, the chatbot might fail to fetch movie details or recommendations.

6. Personalization Limitations

- **Challenge:** Personalized recommendations require extensive user data, which might raise privacy concerns. Furthermore, inadequate data can lead to generic and less effective suggestions.
- **Example:** Without sufficient interaction history, the chatbot might recommend blockbuster movies even to indie film enthusiasts.

7. Technical Accessibility and Maintenance

- **Challenge:** Implementing and maintaining a sophisticated chatbot powered by Rasa NLU requires significant technical expertise and resources, which may not be feasible for smaller businesses or developers.
- **Example:** Regular updates to improve NLU models and keep pace with user expectations demand ongoing investment.

8. Overwhelming Options

- **Challenge:** With the vast array of movies available, users might feel overwhelmed by the number of recommendations, leading to decision fatigue.

- **Example:** Providing too many suggestions without proper curation can make the experience more frustrating than helpful.

8.Conclusion

In this project, we aimed to create Silver Screen AI, a conversational AI platform designed to enhance movie-related interactions. To achieve this, we utilized Python and integrated Rasa NLU for natural language understanding. The system leverages advanced intent recognition and entity extraction to provide users with personalized movie recommendations, trivia, and real-time updates.

Methodology and Data

To build Silver Screen AI, we began by collecting and curating movie-related data, including details about genres, actors, directors, release dates, and user reviews. This data was obtained from publicly available APIs such as TMDb and IMDb. The dataset was preprocessed to ensure consistency, remove duplicates, and handle missing values. Additionally, intent and entity examples were annotated to train the Natural Language Understanding (NLU) model effectively.

An interactive user interface was designed using HTML and CSS to allow seamless user interaction, enabling queries and recommendations to be displayed in an engaging and intuitive manner.

Model Implementation

We implemented the following components to enable robust movie-related conversations:

Intent Recognition with Rasa NLU
The Rasa NLU framework was employed for its efficient handling of natural language inputs. Intents like Recommend_Movie, Get_Movie_Info, and Trending_Now were trained with labeled datasets.
Entity Extraction Key entities such as Movie_Name, Actor_Name, and Genre were extracted to tailor responses to specific user queries.
Recommendation System
A hybrid recommendation system was integrated, combining:
Content-Based Filtering: To suggest movies similar to user preferences.
Collaborative Filtering: To recommend based on trends and user behavior.
Machine Learning Models for Optimization

Machine learning models were incorporated to analyze patterns in user preferences and improve recommendation accuracy.

9.Future Scope

Future Scope of the silverscreen AI and Management System:

1.Multilingual Support

Expand the system to support multiple languages, allowing global users to interact with the AI in their native language and receive culturally relevant recommendations.

2.Enhanced Personalization

Incorporate advanced user profiling techniques, such as analyzing viewing habits and sentiment analysis, to provide highly personalized movie recommendations and anticipate user preferences more effectively.

3.Augmented Reality (AR) Integration

Integrate AR technology to create immersive movie previews or virtual movie theaters, enhancing the user experience for trailers, behind-the-scenes content, or promotional events.

4.Real-Time Industry Updates

Provide real-time updates on box office trends, awards, and streaming platform releases through continuous API integration and machine learning-driven analysis.

5.Social Engagement Features

Add interactive features such as movie quizzes, fan polls, and discussion forums to build a community of movie enthusiasts and encourage engagement.

6.Integration with Smart Assistants

Enable compatibility with popular smart assistants like Alexa, Google Assistant, and Siri to allow hands-free, voice-activated movie conversations.

7.Advanced Sentiment Analysis

Improve sentiment detection to understand user mood and recommend movies that align with their emotional state, such as uplifting films during stressful times.

10.Appendix

10.1 Source Code

HOME Page:

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600&family=Pacifico&family=Raleway:w
ght@400;700&display=swap" rel="stylesheet">
  <title>SilverScreen AI</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/home.css')}}">
</head>
<body>
  <header class="navbar">
    <div class="container">
      <h1 class="logo">SilverScreen AI</h1>
      <nav>
        <ul class="nav-links">
          <li><a href="#home">Home</a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#contact">Contact</a></li>
          <li><a href="#get-started" class="btn">GET STARTED</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <section class="hero" id="home">
    <div class="hero-content">
      <div class="hero-text">
        <h2>Which Genre Movies You Wanttt...<br>I will give you the<br>bestt recommended
movieee...</h2><br>
        <p>Experience seamless movie recommendations and information retrieval with Moviebot, your
intelligence assistant powered by Rasa NLU.</p>
        <a href="#get-started" class="btn">GET STARTED</a>
      </div>
      <div class="hero-image">
        <img src="static/assets/img/sliverscreen home page.webp" alt="Wall-E Robot">
      </div>
    </div>
  </section>
  <!--about-->
  <main >
```

<section class="about" id="about">

<h2>About SilverScreen AI</h2>
SilverScreen AI is not just another movie recommendation platform; it is a sophisticated, AI-powered assistant that brings your movie-watching experience to life. Our platform leverages the latest in Natural Language Understanding (NLU), enabling seamless conversations and highly personalized suggestions tailored specifically to your taste. Whether you're a fan of action-packed thrillers, heartwarming rom-coms, or thought-provoking documentaries, SilverScreen AI is your ultimate movie companion.

With SilverScreen AI, every recommendation is unique, learning from your preferences over time. From trending blockbusters to hidden gems, we bring a world of cinematic experiences at your fingertips. Our platform doesn’t just recommend movies; it understands your mood, genre preferences, and even past watch history, making sure every movie suggestion feels like it was handpicked for you.

<h3>Features:</h3>

Personalized Recommendations: Powered by AI and Rasa NLU, SilverScreen AI understands natural language, giving you highly customized suggestions.
Intelligent Query Handling: Ask anything about movies, whether it's genre-based, actor-related, or even trending.
Movie Discovery: Discover films you might never have come across, based on your interests and viewing history.
Real-time Updates: Stay up-to-date with new releases, ratings, and trending films.

<h2>How It Works</h2>
SilverScreen AI revolutionizes movie discovery with its intuitive chatbot interface. Whether you're undecided on what to watch tonight or looking for suggestions based on your mood, simply engage with the assistant and let it do the work. The assistant’s smart, conversational interface asks simple questions about your preferences, and based on your inputs, it curates a list of movies perfect for your occasion.

From blockbusters to indie masterpieces, SilverScreen AI offers suggestions that fit your taste, helping you explore movies that match your mood, genre preference, or even specific actors. It even learns from your previous choices, refining its recommendations as you interact more with it.

<h3>How it helps you:</h3>

Mood-based Recommendations: Feeling adventurous? In the mood for laughter? Let SilverScreen AI suggest films that match your vibe.
AI-Powered Suggestions: Not only does it learn your preferences, but it also tracks popular trends, user ratings, and the latest releases to offer the best picks.
Comprehensive Movie Information: Along with suggestions, SilverScreen AI provides detailed information about each movie, including reviews, synopsis, trailers, and ratings.

```
        </ul>
        <p>In short, <strong>SilverScreen AI</strong> is more than just a movie recommender—it's an intelligent movie companion that learns from you and delivers an evolving, personalized, and delightful cinematic journey.</p>
        </div>
      </div>
      </section>
  </main>
<!--ask me-->
  <section class="chatbox" id="get-started">
    <div class="chat-container">
      <h2>Welcome to MovieBot!</h2>
      <p>Ask me anything about movies, genres, or recommendations!</p>

      <div class="chat-box" id="chat-box">
        <!-- Chat messages will appear here -->
      </div>

      <div class="chat-input">
        <input type="text" id="user-input" placeholder="Type your message..." autocomplete="off" />
        <button onclick="sendMessage()">Send</button>
      </div>
    </div>
  </section>
<!--contact-->
  <section class="contact" id="contact">
      <header>
        <h1>Contact Us - SilverScreen AI</h1>
        <p>We'd love to hear from you! Whether you have a question, feedback, or just want to connect, don't hesitate to reach out to us.</p>
      </header>

      <section class="contact-details">
        <div class="contact-info">
          <h2>Get in Touch</h2>
          <p><strong>Email:</strong> <a href="mailto:contact@silverscreenai.com">contact@silverscreenai.com</a></p>
          <p><strong>Phone:</strong> +1 (123) 456-7890</p>
          <p><strong>Address:</strong> 123 Tech Avenue, City, Country, ZIP</p>
        </div>

        <div class="social-media">
          <h2>Follow Us</h2>
          <ul>
            <li><a href="https://facebook.com/silverscreenai">Facebook</a></li>
            <li><a href="https://twitter.com/silverscreenai">Twitter</a></li>
            <li><a href="https://linkedin.com/company/silverscreenai">LinkedIn</a></li>
            <li><a href="https://instagram.com/silverscreenai">Instagram</a></li>
```

```
            </ul>
          </div>
        </section>

        <section class="contact-form">
          <h2>Contact Form</h2>
          <form action="#" method="post">
            <label for="name">Your Name</label>
            <input type="text" id="name" name="name" required placeholder="Enter Your Name">

            <label for="email">Your Email</label>
            <input type="email" id="email" name="email" placeholder="Enter Your Email" required>

            <label for="message">Message</label>
            <textarea id="message" name="message" rows="5" placeholder="Please type your
mesage/query/suggestion" required></textarea>

            <button type="submit">Submit</button>
          </form>
        </section>
  </section>

  <footer>
    <p>&copy; 2024 SilverScreen AI. All rights reserved.</p>
  </footer>
</body>

<script>
  function sendMessage() {
    const userMessage = document.getElementById('user-input').value;
    if (userMessage.trim()) {
        // Display user message
        const userMessageElement = document.createElement('div');
        userMessageElement.classList.add('user-message');
        userMessageElement.innerHTML = `<strong>You:</strong> ${userMessage}`;
        document.getElementById('chat-box').appendChild(userMessageElement);

        // Show thinking message
        const botMessageElement = document.createElement('div');
        botMessageElement.classList.add('bot-message');
        botMessageElement.innerHTML = `<strong>MovieBot:</strong> Thinking...`;
        document.getElementById('chat-box').appendChild(botMessageElement);

        // Scroll to the latest message
        document.getElementById('chat-box').scrollTop = document.getElementById('chat-box').scrollHeight;

        // Clear input
        document.getElementById('user-input').value = '';
```

```
        // Send message to Flask backend
        fetch('http://127.0.0.1:5000/webhook', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ message: userMessage })
        })
        .then(response => response.json())
        .then(data => {
            botMessageElement.remove(); // Remove thinking message

            // Append bot's response
            const botResponseElement = document.createElement('div');
            botResponseElement.classList.add('bot-message');
            botResponseElement.innerHTML = `<strong>MovieBot:</strong> ${data.response}`;
            document.getElementById('chat-box').appendChild(botResponseElement);

            // Scroll to the latest message
            document.getElementById('chat-box').scrollTop = document.getElementById('chat-box').scrollHeight;
        })
        .catch(error => {
            console.error('Error:', error);
            botMessageElement.remove(); // Remove thinking message

            // Display error message
            const errorElement = document.createElement('div');
            errorElement.classList.add('bot-message');
            errorElement.innerHTML = `<strong>MovieBot:</strong> Sorry, there was an error processing your
request.`;
            document.getElementById('chat-box').appendChild(errorElement);
        });
    }
}

</script>

</html>
```

HOME.CSS

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }
```

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  color: #333;
}

.navbar {
  background-color: #333;
  color: #fff;
  padding: 20px 0;
  position: sticky;
  top: 0;
  z-index: 1000;
}

.container {
  width: 90%;
  max-width: 1200px;
  margin: auto;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.logo {
  font-size: 24px;
  font-weight: bold;
}

.nav-links {
  list-style: none;
  display: flex;
  gap: 20px;
}

.nav-links a {
  color: #fff;
  text-decoration: none;
}
```

```
.nav-links .btn {
  background-color: #e63946;
  padding: 10px 15px;
  border-radius: 5px;
  color: #fff;
  text-transform: uppercase;
  font-weight: bold;
}

.hero {
  display: flex;
  justify-content: center;
  align-items: center;
  text-align: left;
  padding: 50px 20px;
  background-color: #e4d2d2;
  min-height: calc(105vh - 90px); /* Adjust for navbar height */
}

.hero-content {
  display: flex;
  flex-direction: row;
  align-items: center;
  margin-top: -200px;
  gap: 100px;
}
.hero-text h2{
  font-family: "Courier", monospace;
  font-size: 35px;
}
.hero-text p{
  font-size: 15px;
  margin-bottom: 30px;
  white-space: pre-line; /* Preserve spaces and line breaks */
   width: 500px;
   font-style: italic;
}
.hero-text .btn {
  background-color: #e63946;
  padding: 12px 20px;
  color: #fff;
```

```
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
  }

  .hero-image img {
    width: 400px;
    height: 300px;
    margin-top: 20px;
    border-radius: 10px;
  }

  /* About Section */
.about {
    background-color:#CCD5AE;
    padding: 4rem 2rem;
    text-align: center;
}
#abpa{
    display: flex;
    flex-direction: row;
    gap: 50px;
}
#abpa img{
   width: 400px;
   height: 400px;
   margin-top: 300px;

}
.matter{
   min-width:300px;
}
.about h2 {
    font-size: 2.5rem;
    margin-bottom: 1rem;
}

.about p {
    font-size: 1.2rem;
    margin-bottom: 1.5rem;
```

```
    color: #555;
}

ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

ul li {
    font-size: 1.1rem;
    margin-bottom: 10px;
    color: #333;
}

h3 {
    margin-top: 2rem;
    font-size: 2rem;
}
/*ask me*/
.chatbox {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 80vh;
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  background-image: url('/static/assets/img/askme.jpg');
  padding: 20px;
}

  .chat-container {
    background-color: pink;
    border-radius: 10px;
    padding: 30px;
    width: 800px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  .chat-container h2 {
```

```
  text-align: center;
  color: #333333;
}

.chat-box {
  background-color: #e0e0e0;
  padding: 10px;
  height: 300px;
  overflow-y: auto;
  border-radius: 5px;
  margin-top: 10px;
  margin-bottom: 20px;
}

.chat-input {
  display: flex;
  justify-content: space-between;
}

.chat-input input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 5px;
}

.chat-input button {
  padding: 10px 20px;
  background-color: #333;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.chat-input button:hover {
  background-color: #555;
}

.user-message {
  text-align: left;
```

```
    margin: 10px 0;
    font-size: 14px;
  }

  .bot-message {
    text-align: left;
    margin: 10px 0;
    font-size: 14px;
    color: #007bff;
  }
/* Contact Section */
.contact {
    background-color: #34495e;
    padding: 4rem 2rem;
    color: white;
}

.contact header h1 {
    font-size: 2.5rem;
    margin-bottom: 1rem;
}

.contact header p {
    font-size: 1.2rem;
    margin-bottom: 2rem;
}

.contact-details {
    display: flex;
    justify-content: space-between;
    gap: 20px;
    margin-bottom: 3rem;
    margin-left: 100px;

}

.contact-info,
.social-media {
    width: 48%;
}
```

```
.contact-info p,
.social-media p {
    font-size: 1.1rem;
    margin-bottom: 10px;
}

.social-media ul {
    list-style-type: none;
    display: flex;
    gap: 15px;
}

.social-media a {
    color: white;
    transition: color 0.3s;
}

.social-media a:hover {
    color: #f39c12;
}

.contact-form {
    background-color: white;
    padding: 2rem;
    border-radius: 5px;
}

.contact-form h2 {
    font-size: 2rem;
    margin-bottom: 1.5rem;
    text-align: center;
}

.contact-form label {
    display: block;
    font-size: 1.1rem;
    margin-bottom: 0.5rem;
}

.contact-form input,
.contact-form textarea {
```

```
    width: 100%;
    padding: 10px;
    margin-bottom: 1rem;
    font-size: 1rem;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.contact-form button {
    background-color: #f39c12;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    font-size: 1.2rem;
    transition: background-color 0.3s;
}

.contact-form button:hover {
    background-color: #e67e22;
}

/* Footer */
footer {
    background-color: #2c3e50;
    color: white;
    text-align: center;
    padding: 1.5rem 0;
}

footer p {
    font-size: 1rem;
}
```

APP.PY

```
from flask import Flask, render_template, request, jsonify
import requests

RASA_API_URL = 'http://127.0.0.1:5005/webhooks/rest/webhook'


app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')


@app.route('/webhook', methods=['POST'])
def webhook():
    user_message = request.json['message']
    print("User Message:", user_message)

    # Send the user message to RASA API and get the response
    rasa_response = requests.post(RASA_API_URL, json={'message': user_message})
    rasa_response_json = rasa_response.json()

    print("Rasa response:", rasa_response_json)

    # Get the bot's response or provide a fallback message
    bot_response = rasa_response_json[0]['text'] if rasa_response_json else "Sorry, I am not trained for
this! :("

    # Return the bot's response
    return jsonify({'response': bot_response})

if __name__ == '__main__':
    app.run(debug=True)
```

10.2

GITHUB LINK: **https://github.com/subramanyamposhala/MAJOR_PROJECT**

PROJECT DEMO LINK :CLICK HERE