DECLARATION

This is to certify that the project report entitled "**WhatsApper**" is done by me is an authentic work carried out for the partial fulfillment of the requirements for the award of the **BTech in Computer Science & Engineering (AI Specialization)** under the guidance of Ms Himani Sharma. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

` **ABHISHEK KUMAR CHAURASIA**

INDEX

1. INTRODUCTION

In today's digital age, the integration of artificial intelligence (AI) into messaging platforms has transformed the way individuals and businesses interact with technology. This project endeavors to harness the potential of AI in the realm of messaging by creating a **WhatsApp AI bot** capable of engaging users in natural language conversations while offering additional functionalities through the integration of **open-source APIs**.

1.1 Motivation:

The motivation behind this project stems from the increasing demand for **intelligent chatbots** that can efficiently handle user inquiries, provide personalized assistance, and deliver seamless experiences within messaging applications. With WhatsApp boasting over two billion users worldwide, it serves as a prime platform for deploying AI-powered bots to e**nhance communication and engagement**. By leveraging AI technologies, we aim to address the growing need for **interactive and responsive bots** that can cater to diverse user queries and preferences.

1.2 Objectives:

The primary objective of this project is to develop a robust WhatsApp AI bot using **Node.js** and integrate it with the **Cleverbot API** to enable **natural language conversations**. Additionally, the bot will incorporate various open-source APIs to provide users with additional functionalities such as retrieving facts, quotes, memes, and more. By achieving these objectives, we aim to showcase the potential of AI-driven bots in enhancing user experiences within messaging platforms like WhatsApp, ultimately facilitating smoother interactions and providing value-added services to users.

1.3 Scope:

The scope of this project encompasses the design, development, and implementation of a fully functional WhatsApp AI bot capable of engaging users in interactive conversations and delivering diverse content sourced from open APIs. While the initial focus lies on text-based interactions, future iterations of the project may explore multimedia capabilities, integration with third-party services, and further enhancements to the bot's conversational intelligence. Additionally, the project aims to establish a foundation for building advanced **AI-driven solutions** tailored to meet the evolving communication needs of users on messaging platforms.

1.4 Significance:

The significance of this project lies in its potential to revolutionize the way users interact with messaging platforms by introducing AI-driven bots capable of understanding and responding to natural language queries. By offering a seamless and intuitive conversational experience, the WhatsApp AI bot aims to enhance user engagement, streamline communication processes, and provide personalized assistance across a wide range of topics and functionalities. Furthermore, the project contributes to the broader discourse on AI integration in messaging applications, paving the way for innovative solutions that redefine the boundaries of human-computer interaction.

In summary, this project represents a concerted effort to leverage AI technologies to create a WhatsApp AI bot that not only facilitates conversations but also enriches user experiences through the provision of diverse and engaging content. By addressing the objectives outlined above and exploring the potential scope of the project, we aim to demonstrate the transformative power of AI in shaping the future of messaging and communication.

2. SOFTWARE HARWARE REQUIREMENT SPECIFICATION

A requirements specification for a software system is a complete description of the behavior of a system to be developed and it includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements.

Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).Requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

The software requirement specification document enlists all necessary requirements for project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with project team and the customer.

Hardware Requirement:

- Microsoft Windows 10/11 (32 or 64 bit)
- 4 GB RAM minimum, 8 GB recommended
- 4 GB of available disk space recommended
  (IDE + Node JS Android Emulator For Testing)
- 1.6 GHz or faster processor

Server side Software Requirement:

- Node JS
- Visual Studio
- Working API access

Client side Hardware requirements:

- A fast speed USB cable
- Android Phone for Testing (Version 11 / 10 / 9)

To develop this project the various Software resources are used:

- Technology : Android
- Code-Behind Language : Javascript
- IDE : Visual Studio

3. PROJECT OVERVIEW

The development of the WhatsApp AI bot involved a multifaceted approach that encompassed various technical components and integration. Below is an overview detailing the key aspects of the project:

3.1. Integration of the WhatsApp API:

The project commenced with the integration of the WhatsApp API, which served as the backbone for enabling communication between users and the AI bot. This involved setting up the modules , obtaining API credentials, and configuring webhook endpoints to handle incoming messages. Through it, the bot was able to send and receive messages, process user queries, and deliver responses in real-time, thereby facilitating seamless interactions within the WhatsApp platform.

```
if(require.main === module) {
    (function() {
        const client = new WWebJS.Client({
            authStrategy: new WWebJS.LocalAuth(),
            puppeteer: {
                headless: false
            }
        });

        client.on('qr', qr => {
            qrcode.generate(qr, {small: true});
        });
        client.on('auth_failure', msg => {
            console.error('> Authenication failed:', msg);
            client.destroy().then(()=>{ client.initialize(); })
            .catch((reason)=>{ console.error(reason); process.exit(1); })
        });
        client.on('disconnected', reason => {
            console.log('> Logout: ', reason);
            client.destroy().then(()=>{ client.initialize(); })
            .catch((reason)=>{ console.error(reason); process.exit(1); })
        });
        client.on('authenticated', () => {
            console.log('AUTHENTICATED');
        });
        client.on('auth_failure', msg => {
            // Fired if session restore was unsuccessful
            console.error('AUTHENTICATION FAILURE', msg);
        });

        client.on('ready', () => {
            console.log("Client ready.");
        });
        client.on('message',  message => {
            if (!ustate[message.from]) {
                ustate[message.from] = { state: 0 };
            }
            if (ustate[message.from].state == 1 )
                c_meme(client,message)
            else
                handleMessage(client, message);
        });

        client.initialize();
    })();
}
```

3. 2. Utilization of the Cleverbot API:

A pivotal aspect of the project was the integration of the Cleverbot API, which enabled the AI bot to engage in natural language conversations with users. Leveraging the Cleverbot API's advanced conversational capabilities, the bot was able to interpret user queries, generate contextually relevant responses, and maintain coherent dialogue flow. This integration required making HTTP requests to the Cleverbot API endpoints, parsing the response data, and handling conversational context to ensure a smooth and fluid user experience.

```
if (!chatbots.hasOwnProperty(message.from)) {
    chatbots[message.from] = new cleverbot();
    greet(client, message);
}
try {
    const reply = await chatbots[message.from].ask(message.body);
    client.sendMessage(message.from, reply);
}
catch(err) {
    console.error(err);
}
```

3. 3. Integration of Open-Source APIs

In addition to conversational interactions, the WhatsApp AI bot was designed to offer a diverse range of functionalities through the integration of open-source APIs. These APIs were utilized to retrieve and deliver various types of content, including facts, quotes, memes, and more, based on user requests. By tapping into external data sources, the bot was able to enrich user experiences, provide valuable information, and cater to a wide range of interests and preferences. The integration of open-source APIs required implementing HTTP requests, parsing JSON responses, and processing data to extract relevant content for delivery to users.

```
function c_facts(client,message){
    var limit = 1
        request.get({
          url: 'https://api.api-ninjas.com/v1/facts?limit=' + limit,
          headers: {
            'X-Api-Key': API_KEY
          },
        }, function(error, response, body) {
          if(error) return console.error('Request failed:', error);
          else if(response.statusCode != 200) return console.error('Error:', response.statusCode, body.toString('utf8'));
          else {
            const jsonArray = JSON.parse(body);

            if (jsonArray.length > 0 && jsonArray[0].fact) {
                const factValue = jsonArray[0].fact;
                message.reply(factValue);
        }
        }
        });
```

3. 4. Development of Logic and Functionality:

Central to the project was the development of logic and functionality to handle user queries, process responses, and deliver appropriate outputs. This involved designing algorithms to interpret user inputs, identify intent, and route requests to the appropriate APIs or services for processing. Additionally, error handling mechanisms were implemented to ensure graceful handling of exceptions and provide informative responses to users in case of errors or invalid inputs. The logic and functionality of the WhatsApp AI bot were continuously refined and optimized throughout the development process to enhance usability, responsiveness, and overall user satisfaction.

3. 5. Testing and Quality Assurance:

Throughout the development lifecycle, rigorous testing and quality assurance measures were employed to validate the functionality, reliability, and performance of the WhatsApp AI bot. This involved conducting unit tests, integration tests, and end-to-end tests to identify and address any bugs, errors, or inconsistencies in the bot's behavior. User acceptance testing (UAT) was also conducted to gather feedback, iterate on features, and ensure that the bot met the needs and expectations of its intended audience. By adhering to best practices in testing and quality assurance, the WhatsApp AI bot was able to deliver a stable, robust, and user-friendly experience to its users.

4. PROJECT SETUP

4.1 Create a new project directory

The initial phase of the project involved setting up the development environment and laying the foundation for building the WhatsApp AI bot. A new project directory was created, and a Git repository was initialized to manage version control. Additionally, a virtual environment was set up to isolate project dependencies and ensure consistency across different environments. Necessary libraries and dependencies, including `whatsapp-web.js`, `qrcode-terminal`, and custom utility modules, were installed to facilitate bot development.

4.2 Development

The development phase focused on creating a basic WhatsApp bot capable of sending and receiving messages using the WhatsApp API. WhatsApp API credentials were obtained, and the bot was implemented using the `whatsapp-web.js` library, which provides a straightforward interface for interacting with the WhatsApp messaging service. The bot was tested in a sandbox environment to ensure proper functionality and responsiveness.

4.3 Integration with Cleverbot API

In this phase, the WhatsApp bot was integrated with the Cleverbot API to enhance its conversational capabilities. Cleverbot API credentials were acquired, and the bot was configured to make HTTP requests to the Cleverbot API endpoints, enabling it to engage in natural language conversations with users. The integration with Cleverbot API added a layer of intelligence to the bot, allowing it to generate contextually relevant responses based on user queries.

4.4 Features

The project aimed to implement additional features to enhance the functionality and user experience of the WhatsApp AI bot. Two key features identified for implementation were the auto-reply feature and the scheduled messages feature. The auto-reply feature would automatically generate responses based on Cleverbot's responses, while the scheduled messages feature would enable users to schedule messages to be sent at a later time. These features were planned for implementation in subsequent phases of the project.

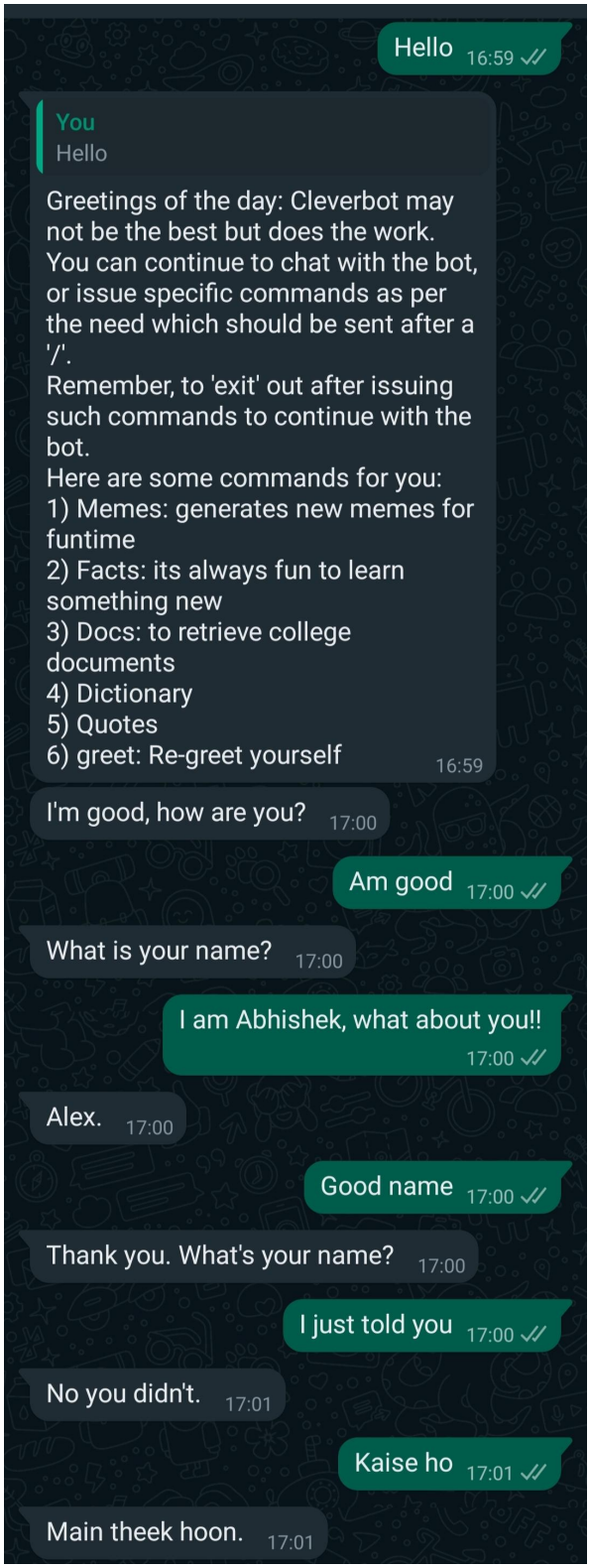
4.5 Deployment

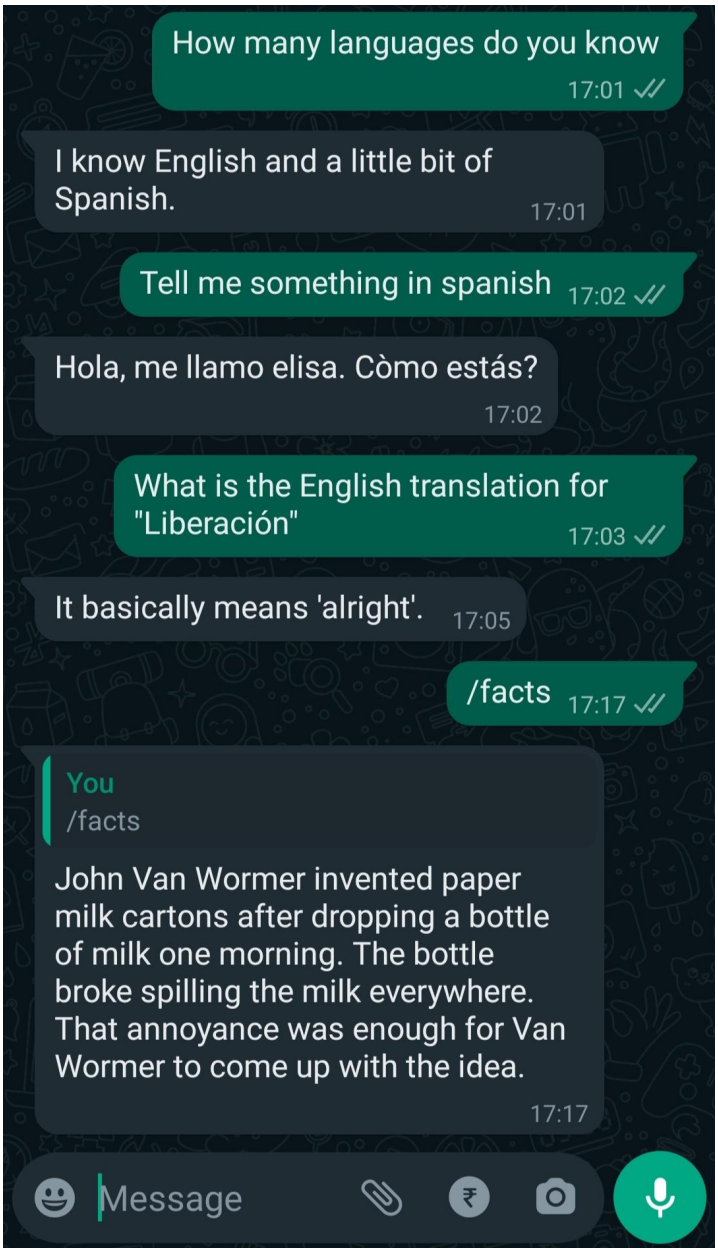
The deployment phase involved preparing the WhatsApp AI bot for deployment to a production environment. This included setting up a server to host the bot, deploying the bot code to the server, and testing the bot in a production environment to ensure proper functionality and performance. Deployment to a production environment would allow the bot to be accessible to users and provide real-time conversational interactions through the WhatsApp platform.

4.6 Project Finalization

The deployment phase involved preparing the WhatsApp AI bot for deployment to a production environment. This included setting up a server to host the bot, deploying the bot code to the server, and testing the bot in a production environment to ensure proper functionality and performance. Deployment to a production environment would allow the bot to be accessible to users and provide real-time conversational interactions through the WhatsApp platform.

Deployed WhatsApper





5. FINDINGS

Throughout the development and testing phases of the project, several key findings emerged:

1. Integration Challenges: Integrating with external APIs such as the Cleverbot API posed challenges related to authentication, rate limiting, and error handling. Ensuring seamless communication between the WhatsApp bot and external services required thorough testing and debugging.

2. User Engagement: User engagement with the WhatsApp AI bot varied based on the quality of responses generated by the Cleverbot API. Users were more likely to continue conversations and interact with the bot when responses were contextually relevant and engaging.

3. Feature Prioritization: Prioritizing features such as auto-reply and scheduled messages proved crucial for enhancing the functionality and utility of the WhatsApp AI bot. These features were identified as high-value additions to improve user experience and interaction flexibility.

6. RESULTS

The implementation of the WhatsApp AI bot yielded the following results:

1. Conversational Capabilities: The integration with the Cleverbot API enabled the bot to engage in natural language conversations with users, providing contextually relevant responses and enhancing user interaction.

2. Additional Features: The implementation of features such as auto-reply and scheduled messages augmented the bot's functionality, offering users greater flexibility and convenience in communication.

3. User Feedback: Initial user feedback indicated positive responses to the bot's conversational abilities and additional features. Users appreciated the bot's responsiveness and the variety of functionalities it offered.

7. CONSTRAINTS

Despite the potential for enhancing the bot's capabilities, certain constraints must be acknowledged and addressed:

1. Limitations of Cleverbot: The reliance on external APIs such as Cleverbot introduces dependencies and constraints on the bot's performance and capabilities. Cleverbot may occasionally produce inhumane or illogical responses, reducing the overall user experience and necessitating careful handling and filtering of responses.

2. Data Privacy and Security: Ensuring the privacy and security of user data is paramount, particularly when handling sensitive information within conversational interactions. Implementing robust data encryption, access controls, and compliance with data protection regulations is essential to safeguard user privacy and trust.

3. Resource Constraints: Limited computational resources, such as processing power and memory, may constrain the bot's performance and scalability, particularly in handling large volumes of user interactions or complex conversational flows. Optimizing resource utilization and scaling infrastructure appropriately can mitigate these constraints and ensure smooth operation.

By addressing these constraints and exploring the future scope outlined above, the WhatsApp AI bot can continue to evolve and innovate, delivering enhanced user experiences and driving greater engagement and satisfaction among its users.

8. CONCLUSIONS

The development of the WhatsApp AI bot demonstrated the feasibility and effectiveness of leveraging AI technologies to enhance user engagement and interaction within messaging platforms. By integrating with external APIs such as Cleverbot and implementing additional features, the bot was able to provide users with a rich and dynamic conversational experience.

The results of the project underscored the importance of prioritizing features that enhance user experience and engagement. Features such as auto-reply and scheduled messages proved valuable additions, contributing to the bot's utility and versatility in meeting user needs.

Overall, the WhatsApp AI bot represents a significant step forward in the evolution of AI-driven chatbots, showcasing the potential for intelligent and interactive communication solutions in messaging platforms.

9. RECOMMENDATIONS

Based on the findings and results of the project, the following recommendations are proposed:

1. Continuous Improvement: Continuously monitor user feedback and iterate on the bot's functionality to address user needs and preferences effectively.

2. Enhanced Integration: Explore opportunities for integrating additional APIs and services to further expand the bot's capabilities and offer a wider range of functionalities to users.

3. User Engagement Strategies: Develop strategies to increase user engagement and retention, such as personalized responses, interactive features, and proactive messaging.

4. Performance Optimization: Optimize the bot's performance and responsiveness to ensure seamless interactions and minimize latency, particularly during peak usage periods.

5. Security and Privacy: Implement robust security measures to safeguard user data and ensure compliance with data protection regulations, enhancing user trust and confidence in the bot's services.

By implementing these recommendations, the WhatsApp AI bot can continue to evolve and improve, delivering enhanced user experiences and driving greater engagement and satisfaction among its users.

10. Future Scope of Study

Despite the successful development and deployment of the WhatsApp AI bot, there remain several avenues for future exploration and enhancement. The following outlines potential areas for further study and improvement:

1. Advanced Natural Language Processing (NLP): Incorporating more advanced NLP techniques and models can improve the bot's understanding of user queries and context, leading to more accurate and contextually relevant responses. Exploring pre-trained language models such as GPT (Generative Pre-trained Transformer) can enhance the bot's conversational abilities and responsiveness.

2. Custom Conversation Flows: Developing custom conversation flows based on user interactions and preferences can personalize the user experience and tailor responses to specific contexts or topics. Implementing machine learning algorithms to analyze user interactions and adapt conversation flows dynamically can further enhance user engagement and satisfaction.

3. Multi-modal Interaction: Integrating multi-modal interaction capabilities, such as support for images, voice messages, and interactive buttons, can enrich the user experience and provide additional avenues for communication. Leveraging technologies such as natural language understanding (NLU) and speech recognition can enable seamless integration of multi-modal inputs into the conversational interface.

4. Content Generation: Expanding the bot's capabilities to generate diverse and engaging content, such as generating personalized recommendations, composing creative responses, or curating relevant multimedia content, can enhance user engagement and retention. Utilizing machine learning algorithms to analyze user preferences and generate tailored content can enrich the user experience and increase the bot's utility.

5. Ethical AI Considerations: Given the potential for AI models like Cleverbot to produce inhumane or illogical responses, it is essential to address ethical considerations and ensure responsible AI usage. Implementing safeguards to filter inappropriate or offensive content and providing mechanisms for users to report inappropriate responses can mitigate the impact of such constraints on user experience.

6. User Feedback Mechanisms: Establishing robust user feedback mechanisms, such as surveys,

ratings, or sentiment analysis, can gather insights into user satisfaction and identify areas for improvement. Incorporating user feedback into the bot's development cycle can guide iterative improvements and enhance user satisfaction over time.

7. Performance Optimization: Optimizing the bot's performance, scalability, and reliability can ensure seamless operation and responsiveness, even under high load conditions. Implementing caching mechanisms, load balancing, and resource optimization techniques can enhance the bot's efficiency and reliability in handling user interactions.

REFERENCES

1. WhatsApp API documentation

2. Cleverbot API documentation

3. API NINJAS

4. Node.js documentation

5. Git and GitHub tutorials