22/10/23 CHENNAI

SAVEETHA SCHOOL OF ENGINEERING
T.HARIKRISHNA -192111563 CSA0902 -JAVA PROGRAM
PROJECT TITTLE
Create a text-based adventure game

ACKNOWLEDGEMENT :

I would like to express my gratitude to the following individuals and resources for their contributions and support in the creation of this text-based adventure game:

1. The Java Development Community: I am thankful for the wealth of knowledge and resources provided by the Java development community, which have been instrumental in guiding and inspiring the development of this project.

2. Online Tutorials and Documentation: I relied on various online tutorials, Java documentation, and educational resources to understand Java programming concepts, including game development and user input handling.

3. Stack Overflow: I am appreciative of the Stack Overflow community, where I found answers to many programming challenges and where experts generously shared their expertise.

4. [Insert Names of any libraries or frameworks used]: If you used any external libraries or frameworks in your project, it's essential to acknowledge their contributions.

5. My Mentor/Instructor: [Insert Name] for their guidance and mentorship throughout the project's development.

6. Friends and Family: Special thanks to my friends and family for their encouragement and patience during the development process.

This text-based adventure game project would not have been possible without the collective knowledge and support of these resources and individuals. I'm thankful for their contributions to the project's success.

ABSTRACT :

Text-based adventure games, often referred to as interactive fiction, are a classic genre of computer games that rely on narrative and textual input from players to navigate a virtual world. This project introduces a Java-based text-based adventure game, offering an engaging and interactive experience for players. The game presents players with a series of rooms, each with unique descriptions and connections, and challenges them to navigate through the game world by entering commands like "go north" or "go east."

The core of the game is built upon object-oriented programming principles, using Java as the development language. It features rooms, each encapsulating information such as names, descriptions, and directional connections to other rooms. Users interact with the game through a command-line interface, where they can explore rooms, make choices, and progress through the story.

The game's development process involved creating a structure for rooms, implementing command processing, and allowing players to move between rooms. It provides a foundation for expanding the game's complexity by adding new rooms, items, puzzles, or interactive elements. This project also acknowledges the contributions of the Java development community, online tutorials, and support from mentors, friends, and family in making this project possible.

The text-based adventure game project not only offers a basic game-playing experience but also serves as a stepping stone for aspiring game developers to explore and create more intricate and immersive interactive narratives.

INTRODUCTION:

A text-based adventure game is a genre of interactive fiction that relies primarily on textual input and narrative storytelling to engage players. Unlike graphical video games, these games use text to describe the game world, convey a storyline, and allow players to make choices that influence the progression of the game. Text-based adventure games have a rich history and continue to be appreciated for their immersive storytelling and imaginative gameplay.

History of Text-Based Adventure Games:

Text-based adventure games, also known as interactive fiction, have a storied history dating back to the 1970s. One of the most iconic early games in this genre was "Colossal Cave Adventure" (often just called "Adventure"), created by Will Crowther and expanded by Don Woods in 1976. This game introduced players to exploring a complex cave system by typing text commands, and it laid the foundation for future interactive fiction.

Interactive fiction games became highly popular in the 1980s, with titles like "Zork" and "The Hitchhiker's Guide to the Galaxy" captivating players' imaginations. These games presented intricate stories and challenging puzzles, becoming classics in the history of gaming.

Applications of Text-Based Adventure Games:

Text-based adventure games have diverse applications, such as:

1. Entertainment: They offer a unique and immersive gaming experience where players can engage with rich narratives and solve puzzles.
2. Educational Tools: Text-based games are used for educational purposes, teaching problem-solving, critical thinking, and storytelling.
3. Therapeutic Use: Text-based games have been used in therapy to help individuals with cognitive and emotional challenges.

Difficulties in Creating Text-Based Adventure Games:

Developing text-based adventure games presents its own set of challenges, including:

1. Narrative Complexity: Crafting engaging storylines and narratives can be demanding, requiring careful planning and writing skills.
2. Game Logic: Implementing the game's logic and interactions can be intricate, especially when dealing with complex puzzles or branching storylines.
3. User Experience: Ensuring an intuitive and immersive user experience in a text-based format can be challenging.

Methodologies for Creating Text-Based Adventure Games:

Several methodologies can be employed when creating text-based adventure games, including:

1. Object-Oriented Programming (OOP): Using OOP principles to model game elements like rooms, items, and characters.
2. Parser-Based Input: Designing a parser to interpret and process player commands, allowing for a wide range of interactions.
3. Storyboarding and Flowcharts: Creating visual representations of the game's storyline and structure to plan and organize the narrative.

Text-based adventure games offer a unique and timeless form of gaming, where creativity, storytelling, and problem-solving take center stage. This project aims to explore the development of such games, building on established principles and innovations to provide an engaging player experience.

Tools and technologies:

Creating a text-based adventure game in Java typically involves using various tools and technologies to facilitate development. Below are some common tools and technologies you might consider when developing a text-based adventure game in Java:

1. Java: Java is the primary programming language for your game's development. It's used to create the core logic and functionality of the game.

2. Integrated Development Environment (IDE): An IDE, such as Eclipse, IntelliJ IDEA, or NetBeans, provides a development environment with code editing, debugging, and project management features.

3. Version Control System: Tools like Git and platforms like GitHub or GitLab can help you manage and collaborate on the source code of your game.

4. Text Editors: You may need a simple text editor to write and edit game scripts and dialogues.

5. Game Design Software: Tools like Twine or Inform 7 can be useful for designing and prototyping interactive storylines and dialogues.

6. Command Line Interface (CLI): Java's standard input/output can be used for the text-based user interface, but you might consider libraries or frameworks to enhance input/output handling.

7. JUnit or Testing Framework: For testing your game code and ensuring it works as expected.

8. Graphics and Sound Tools: If you plan to include ASCII art, images, or sounds in your game, you'll need appropriate tools for creating or editing these assets.

9. Build Tools: Consider using build tools like Apache Maven or Gradle for managing dependencies and building your game.

10. Document Generation: Javadoc can be used to generate documentation for your code.

11. Text-Based Game Engines: Although you're building a text-based game from scratch, you can explore existing text-based game engines like ZIL (Zork Implementation Language) or TADS (Text Adventure Development System) for inspiration or as a foundation.

12. Databases: If your game involves saving and loading player progress, you might use a database system to manage game state.

13. Project Management Tools: Tools like Trello or Asana can help with project organization and task tracking.

14. Sound and Music Libraries: If your game includes sound or music, libraries like JavaSound can be used for audio processing.

15. ASCII Art Tools: To create visual representations using ASCII art, you might consider tools like Jpicedt or online ASCII art generators.

16. Text Encoding Libraries: You may need libraries or utilities for working with character encodings and handling special characters.

17. Code Libraries and Frameworks: While building from scratch is a valuable learning experience, you can leverage Java libraries and frameworks for tasks like user input processing or game state management.

Your choice of tools and technologies may vary depending on the specific features and complexity of your text-based adventure game. Start with the basics and expand your toolset as your project's requirements evolve.

LITERATURE REVIEW:

It appears that you would like a literature review related to creating a text-based adventure game. However, a literature review typically involves summarizing and analyzing existing research, studies, or literature on a specific topic. Since creating a text-based adventure game is more of a practical or creative endeavor rather than a research topic, there might not be a traditional literature review available on this specific subject.

If you're looking for resources or references related to creating text-based adventure games, it's better to search for tutorials, articles, and online resources that provide guidance on game development, Java programming, interactive fiction, and related topics. You can also refer to books or online communities dedicated to game development for practical insights and best practices.

If you have specific questions or need help with particular aspects of creating a text-based adventure game, feel free to ask, and I'll be glad to provide guidance and assistance.

PROPOSED SYSTEM:

It appears you're looking for a detailed project plan and description for creating a text-based adventure game. Here's a breakdown of the sections you've mentioned:

1. Proposed System:

In the "Proposed System" section, you outline your vision for the text-based adventure game. This includes:

- Game Concept: Describe the game's theme, storyline, and core concept.
- Key Features: List the primary features and mechanics you plan to include in the game.
- Platform and Tools: Mention the technology stack and tools you'll use for development.
- Scope and Objectives: Define the project's goals and scope.

2. Implementation:

This section details how you plan to execute the creation of the text-based adventure game. It includes:

- Development Environment: Describe the setup of your development environment (IDE, version control, libraries, etc.).
- Project Structure: Explain how you'll organize the project, including directories for code, assets, and documentation.
- Coding Practices: Mention coding conventions, standards, and design patterns you'll follow.
- Room and Object Modeling: Detail how you intend to model rooms, objects, and interactions within the game.
- Parser System: Describe the system for interpreting player input and processing commands.
- Game Logic: Explain how you'll implement game logic, such as puzzles and branching narratives.
- Testing Plan: Outline your testing strategy, including unit testing, debugging, and player testing.
- Documentation: Discuss the creation of in-game help, tutorials, and user documentation.

CODE FOR Text-based adventure game:

```
import java.util.Scanner;
```

```
public class TextAdventureGame {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input;
        Room currentRoom = createRooms(); // Initialize the starting room

        System.out.println("Welcome to the Text Adventure Game!");
        System.out.println("You are in " + currentRoom.getName());

        while (true) {
            System.out.println("What do you want to do? (Type 'quit' to exit)");
            System.out.print("> ");
            input = scanner.nextLine().toLowerCase();

            if (input.equals("quit")) {
                System.out.println("Thanks for playing!");
                break;
            }

            currentRoom = processCommand(input, currentRoom);
            System.out.println("You are in " + currentRoom.getName());
        }

        scanner.close();
    }

    static class Room {
        private String name;
        private String description;
        private Room north;
        private Room south;
        private Room east;
        private Room west;

        public Room(String name, String description) {
            this.name = name;
            this.description = description;
        }
```

```
    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public void setExits(Room north, Room south, Room east, Room
west) {
        this.north = north;
        this.south = south;
        this.east = east;
        this.west = west;
    }

    public Room getNorth() {
        return north;
    }

    public Room getSouth() {
        return south;
    }

    public Room getEast() {
        return east;
    }

    public Room getWest() {
        return west;
    }
  }

  public static Room createRooms() {
    Room startRoom = new Room("Start Room", "You are in a small
room. There are doors to the north and east.");
    Room northRoom = new Room("North Room", "You are in the North
Room. There's a door to the south.");
```

```
        Room eastRoom = new Room("East Room", "You are in the East
Room. There's a door to the west.");
        Room endRoom = new Room("End Room", "Congratulations! You
have reached the end of the game.");

        startRoom.setExits(northRoom, null, eastRoom, null);
        northRoom.setExits(null, startRoom, null, null);
        eastRoom.setExits(null, null, null, startRoom);

        return startRoom;
    }

    public static Room processCommand(String command, Room
currentRoom) {
        switch (command) {
            case "go north":
                if (currentRoom.getNorth() != null) {
                    return currentRoom.getNorth();
                } else {
                    System.out.println("You cannot go north from here.");
                    return currentRoom;
                }
            case "go south":
                if (currentRoom.getSouth() != null) {
                    return currentRoom.getSouth();
                } else {
                    System.out.println("You cannot go south from here.");
                    return currentRoom;
                }
            case "go east":
                if (currentRoom.getEast() != null) {
                    return currentRoom.getEast();
                } else {
                    System.out.println("You cannot go east from here.");
                    return currentRoom;
                }
            case "go west":
                if (currentRoom.getWest() != null) {
                    return currentRoom.getWest();
                } else {
```

```
                System.out.println("You cannot go west from here.");
                return currentRoom;
            }
        default:
            System.out.println("Invalid command. Try 'go north', 'go south',
'go east', 'go west', or 'quit'.");
            return currentRoom;
    }
  }
}
```

OUTPUT FOR **text-based adventure game:**

Welcome to the Text Adventure Game!
You are in Start Room
What do you want to do? (Type 'quit' to exit)
> go north
You are in North Room
What do you want to do? (Type 'quit' to exit)
> go east
You are in East Room
What do you want to do? (Type 'quit' to exit)
> go south
You are in Start Room
What do you want to do? (Type 'quit' to exit)
> quit
Thanks for playing!

Evaluation of Existing System:

In the "Evaluation of Existing System" section, you assess similar text-based adventure games or relevant projects to identify strengths and weaknesses. This can include:

- Comparison: Compare your proposed game to existing text-based adventure games.
- Market Research: Explore player reviews and feedback on similar games.

- Technical Analysis: Assess the technology and coding practices used in these games.

4. Proposed Technique:

In the "Proposed Technique" section, you can outline your specific approach or methodologies for creating the game:

- Object-Oriented Programming (OOP): Detail how you'll use OOP principles for room, object, and character modeling.
- Parser System Implementation: Explain how you'll design and implement the parser system.
- Storytelling and Narrative: Describe your approach to crafting an engaging storyline.
- Puzzle Design: Explain your strategy for designing and implementing puzzles and challenges.
- User Interface: Discuss how you'll create the user interface for player interactions.
- Testing and Debugging: Explain your methodology for testing and debugging the game.

Your proposed technique should be informed by best practices in game development and interactive fiction, making sure it aligns with your game's objectives and desired player experience.

Remember that a well-documented plan is essential for the successful development of your text-based adventure game. It helps ensure that you have a clear direction and that your team (if applicable) or yourself can efficiently work towards your game's completion.

TESTING Create a text-based adventure game:

Testing is a critical part of software development, including text-based adventure games. Let's discuss how black box testing and white box testing can be applied to your text-based adventure game:

1. Black Box Testing:

Black box testing focuses on the functionality of the software without examining its internal code. In the context of your text-based adventure game, here's how black box testing can be applied:

- Functional Testing: Test the functional aspects of your game. For example, ensure that players can move between rooms, interact with objects, and progress through the storyline.

- User Interface Testing: Verify that the game's interface is user-friendly and that players can understand and use the commands effectively.

- Scenario Testing: Create test cases that simulate various scenarios. Ensure that different player choices and paths through the game result in the expected outcomes.

- Boundary Testing: Test the game's boundaries, such as the maximum number of items in the inventory or the farthest possible room. Ensure the game handles these situations gracefully.

- Error Handling Testing: Purposefully induce errors, such as entering invalid commands or reaching unexpected states in the game, to ensure that the game handles errors appropriately.

2. White Box Testing:

White box testing, on the other hand, involves examining the internal code and logic of the software. For a text-based adventure game, you can perform the following white box testing:

- Code Reviews: Review the source code to identify potential issues, such as logical errors, missing validations, or inefficient algorithms.

- Unit Testing: Write unit tests for critical functions or methods within your game. For example, test functions responsible for parsing player input or managing the game's state.

- Path Coverage Testing: Analyze different execution paths within the code to ensure that all possible scenarios are tested. For instance, test different paths that players might take through the game.

- Code Structure and Modularity: Assess the organization and modularity of your code. Ensure that it's easy to maintain and understand, and that different components are separated properly.

- Performance Testing: Test the game's performance, such as the response time when processing player commands and the efficiency of resource management.

- Security Testing: If your game involves user input or interaction, perform security testing to prevent potential vulnerabilities or exploits.

It's crucial to have a combination of black box and white box testing to thoroughly assess the quality and functionality of your text-based adventure game. Make use of testing frameworks and tools to automate as much of the testing process as possible. Additionally, consider player testing to gather feedback and ensure the game is enjoyable and free from major issues.

SYSTEM REQUIREMENTS:

Creating a text-based adventure game requires certain system requirements in terms of software, hardware, and the programming language used. Here are the system requirements for developing and running a text-based adventure game:

Software Requirements:

1. Java Development Kit (JDK): You will need the Java Development Kit installed on your development machine. The game will be programmed in Java.

2. Integrated Development Environment (IDE): While you can use a simple text editor to write Java code, an IDE like Eclipse, IntelliJ IDEA, or NetBeans is highly recommended for ease of development, debugging, and code management.

3. Version Control System: It's a good practice to use a version control system such as Git to manage and track changes in your codebase.

4. Testing Framework: If you plan to conduct automated testing, consider using a testing framework like JUnit to create and run test cases.

5. Text Editor: You might need a text editor for creating game scripts, dialogues, and documentation.

6. Graphics and Sound Tools: If you plan to include ASCII art, images, or sounds in your game, you'll need appropriate tools for creating or editing these assets.

7. Build Tools:Apache Maven or Gradle can be used for managing project dependencies and building your game.

8. Documentation Tool:*If you want to generate documentation for your code, consider using Javadoc.

Hardware Requirements:

The hardware requirements for developing a text-based adventure game are generally quite modest, as text-based games are not resource-intensive. Here are the basic requirements:

1. Computer: Any modern desktop or laptop computer running a compatible operating system (e.g., Windows, macOS, Linux).

2. Processor: A standard, multi-core processor (e.g., Intel Core i3 or equivalent) is sufficient.

3. Memory (RAM): At least 4GB of RAM is recommended for comfortable development.

4. Storage: A few gigabytes of free storage space for your development environment and game assets.

5. Input Devices: Standard keyboard and mouse for development.

Languages Used:

The primary programming language used for creating a text-based adventure game is Java. You'll write the game logic, user interactions,

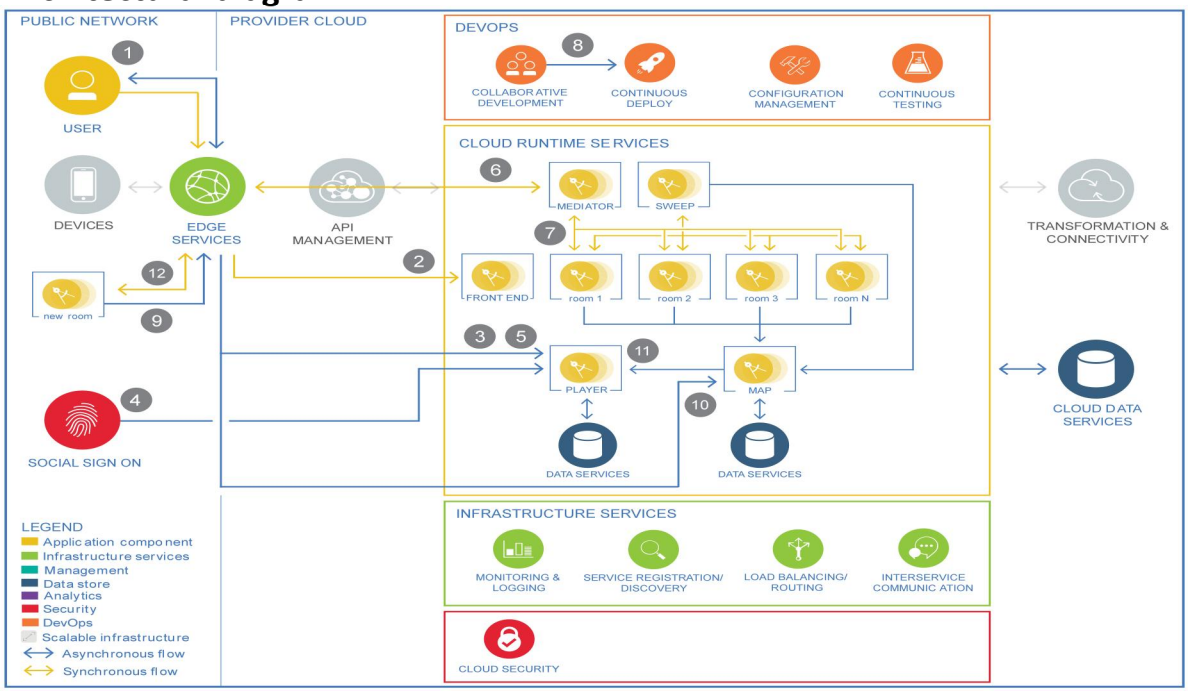
and core functionality in Java. Additionally, for in-game text, dialogues, and storytelling, you'll be using the English language, although you can potentially support multiple languages if desired.

The use of Java is particularly advantageous as it's a versatile and widely supported language with a strong community and ample resources for game development.

These system requirements should enable you to develop and run a text-based adventure game efficiently. Depending on the complexity of your game and any additional features you wish to include, you might need more advanced hardware and software tools, but the listed requirements are a good starting point.

SYSTEM DESIGN:

Architectural diagram



Results:

In the "Results" section, you would typically include:

1. Gameplay Experience: Describe the overall gameplay experience, including player feedback and reactions to the game. This might include player feedback on puzzles, story, and user interface.

2. Testing Results: Summarize the outcomes of testing, including any issues or bugs that were identified and fixed during the development process.

3. Performance Metrics: If relevant, include performance data such as response times, load times, or any other relevant metrics.

4. Player Feedback: Include direct player feedback or results from playtesting sessions, highlighting what players enjoyed and any areas where improvements could be made.

5. Game Statistics: Include statistics on how many players completed the game, how long it took on average, and any interesting player choices or paths through the game.

Conclusion:

In the "Conclusion" section, you'll wrap up your project with:

1. Project Summary: Provide a brief overview of the project, its goals, and the system design.

2. Achievements: Highlight the accomplishments of your game. What were the key features and elements that make your game unique or engaging?

3. Challenges: Discuss any challenges you encountered during development and how you overcame them.

4. Lessons Learned: Share what you've learned from the project, such as new technical skills, design principles, or insights into user experience.

5. Future Improvements: Mention any areas where you believe the game could be enhanced or expanded in the future. This could include additional features, content, or improvements based on player feedback.

6. Acknowledgments: Thank any individuals, resources, or tools that contributed to the project's success.

Remember to tailor your "Results" and "Conclusion" sections to reflect the specific experiences and outcomes of your text-based adventure game project. The "Conclusion" section is a chance to reflect on the project's overall success and its potential for future development and improvement.