Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

OOPs Project Presentation

Snakes and Ladders

Y. Chauhan[1] S. Ladhe[1] S. Chinchkar[1] S. Kumar[1]

[1]Computer Science and Engineering
Indian Institute of Information Technology - Vadodara, International Campus Diu

18th October, 2023

Table of Contents

1 Introduction

2 Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

3 Technology Stack

4 OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

5 Conclusion
- Dependencies
- Future Aspects
- Conclusion

Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

Introduction

Snakes and Ladders

- The Snakes and Ladders Game is a digital recreation of the classic board game. The primary aim is to provide an enjoyable and interactive gaming experience for players of all ages.
- Snakes and Ladders is one of the most recognizable board games today. Originated in ancient India around the 13th century AD, the game was designed to teach children the cause and effect of good and bad deeds.

Objectives

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications
Functional Requirements
Non-Functional Requirements
Use-Case Diagram

Technology Stack

OOA
Identifying Classes
Inherited Classes
Class Diagram
OOAD Principles

Conclusion
Dependencies
Future Aspects
Conclusion

- Our project aims to demonstrate the effectiveness of an Object-Oriented approach in solving complex problems.
- We'll showcase how abstraction and inheritance enhance efficient product design.
- Object-oriented concepts streamline debugging and optimize the CI/CD Pipeline.
- The Web-App interface ensures compatibility across all devices and eliminates support concerns.
- A simple Web-App guarantees playability on any device with internet access and a browser.

SRS and Use Case Diagram

1. Functional Requirements
2. Non-Functional Requirements
3. Use Case Diagram

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Functional Requirements

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

- **Die Rolling:** Implement random die roll functionality (1-6).
- **Player Movement:** Move the player's game piece based on the die roll.
- **Consecutive 6s Rule:** Detect three consecutive 6s and void the last 6.
- **Normal Block:** Move the player to the designated block.
- **Snake Head Block:** Move the player to the corresponding snake's tail block.
- **Ladder Bottom Block:** Move the player to the corresponding ladder's top block.

Non-Functional Requirements

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

- **User Interface:** Intuitive and visually appealing user interface.
- **Performance:** Smooth game play with responsive controls.
- **Compatibility:** The game should run on popular web browsers.
- **Security:** Ensure data privacy and prevent cheating.

Use Case Diagram

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

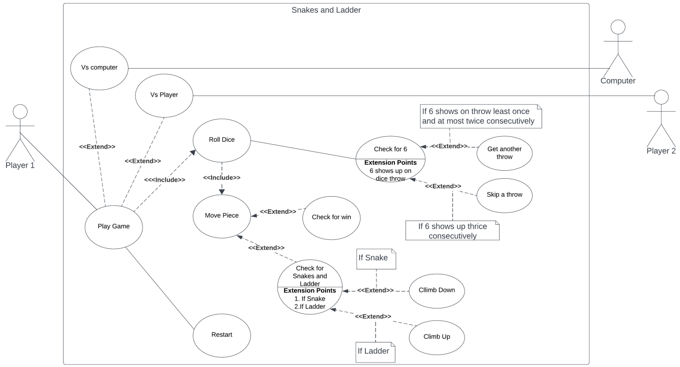

Figure: Use Case Diagram

General Functionality

- The game first starts by first giving the control to player to roll the dice where as soon as player clicks the icon of dice, by using a random number generator a number between 1 and 6 is generated.
- When the number generated the computer inputs it and moves the respective piece accordingly. These two processes are repeated until one of the player reaches 100.
- If the player lands on a snake mouth his/her piece will travel down to its tail cell.
- If the player lands on a ladders start his/her piece will travel up the ladder.
- First player to reach 100 will win the game.

Introduction
Software Requirement Specifications
Functional Requirements
Non-Functional Requirements
Use-Case Diagram
Technology Stack
OOA
Identifying Classes
Inherited Classes
Class Diagram
OOAD Principles
Conclusion
Dependencies
Future Aspects
Conclusion

Technology Stack

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

1. Front End

Front End

- React JS
- CSS

Technology Stack

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

1. Front End

Contributors

- Shreyas Ladhe
- Sneha Chinchkar

Technology Stack

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

1. Front End
2. Back-End

Back End

- Django
- Python

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Technology Stack

1. Front End
2. Back-End

Contributors

- Yuvraj Chauhan
- Suraj Kumar

Technology Stack

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

1. Front End
2. Back-End
3. Version Control

Version Control

- Git
- GitHub

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Technology Stack

1. Front End
2. Back-End
3. Version Control

Contributors

- Yuvraj Chauhan
- Shreyas Ladhe
- Sneha Chinchkar
- Suraj Kumar

Technology Stack

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

1. Front End
2. Back-End
3. Version Control
4. UI/UX

UI/UX

- Canva

Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

Technology Stack

1. Front End
2. Back-End
3. Version Control
4. UI/UX

Contributors

- Shreyas Ladhe

Object Oriented Analysis

Yuvraj, Shreyas, Sneha, Suraj

Introduction
Software Requirement Specifications
Functional Requirements
Non-Functional Requirements
Use-Case Diagram
Technology Stack
OOA
Identifying Classes
Inherited Classes
Class Diagram
OOAD Principles
Conclusion
Dependencies
Future Aspects
Conclusion

1. Identifying Classes
2. Inherited Classes
3. Class Diagram

Identifying Classes

- Player
- Dice
- Cell
- Board
- Game

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

Inherited Classes

- Inherited Class from Cell:
  - firstCell
  - Jumper
  - lastCell
- Inherited Class from Jumper
  - Snakes
  - Ladders

Introduction
Software Requirement Specifications
Functional Requirements
Non-Functional Requirements
Use-Case Diagram
Technology Stack
OOA
Identifying Classes
Inherited Classes
Class Diagram
OOAD Principles
Conclusion
Dependencies
Future Aspects
Conclusion

Class Diagram

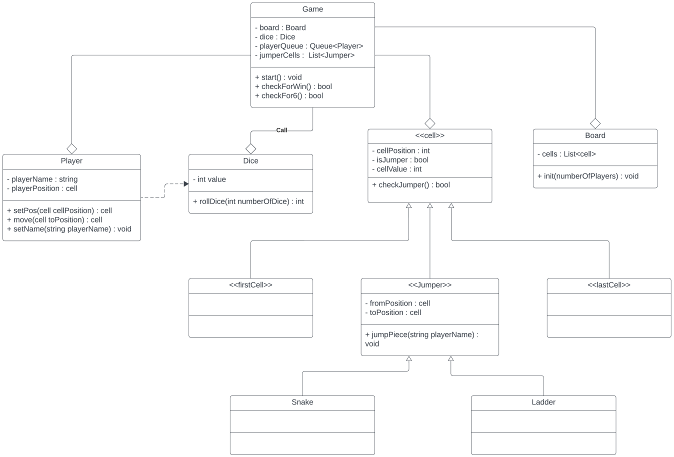

Figure: Class Diagram

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

OOAD Principles

1. Encapsulation

Encapsulation

- We created classes and **hid the implementation** from the user.
- We **stopped external entities** from accessing data from classes.
- We **bound the attributes and methods together** and **wrapped it up in a capsule**.

OOAD Principles

Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

1. Encapsulation
2. Abstraction

Abstraction

- From a large pool of data, only a **selected data** was chosen to be displayed to the user.
- Only **essential Attributes** were shown to the user.
- We **hid unnecessary information** from the user.

IIITV-ICD

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications
- Functional Requirements
- Non-Functional Requirements
- Use-Case Diagram

Technology Stack

OOA
- Identifying Classes
- Inherited Classes
- Class Diagram
- OOAD Principles

Conclusion
- Dependencies
- Future Aspects
- Conclusion

OOAD Principles

1. Encapsulation
2. Abstraction
3. Modularity

Modularity

- We broke down a complex problem into smaller parts, or a problem of manageable size.
- Now, by looking at each class, it becomes significantly clear how our system works in harmony.

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

OOAD Principles

1. Encapsulation
2. Abstraction
3. Modularity
4. Hierarchy

Hierarchy

- We used **IS-A** hierarchy or **Abstraction hierarchy** to create different classes for cells.

OOPs Project Presentation

Yuvraj, Shreyas, Sneha, Suraj

Introduction

Software Requirement Specifications
Functional Requirements
Non-Functional Requirements
Use-Case Diagram

Technology Stack

OOA
Identifying Classes
Inherited Classes
Class Diagram
OOAD Principles

Conclusion
Dependencies
Future Aspects
Conclusion

Dependencies

- Requires a modern browser to run.
- Multiplayer functionality is not available
- Interface might get overloaded due to multiple requests (dice roll).

IIITV-ICD

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Future Aspects

- Multiplayer functionality
- Save game progress and loading saved game
- Incorporating Cloud hosting and saving game progress on Cloud.

Introduction

Software Requirement Specifications

Functional Requirements

Non-Functional Requirements

Use-Case Diagram

Technology Stack

OOA

Identifying Classes

Inherited Classes

Class Diagram

OOAD Principles

Conclusion

Dependencies

Future Aspects

Conclusion

Conclusion

Conclusion

Our Object-Oriented "Snakes and Ladders" project demonstrates the power of OOP principles for efficient, maintainable games. Future plans include multiplayer, saved game progress, and cloud integration.