

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

Conclusion

Dependencies
Future Aspects
Conclusion

OOPs Project Presentation

Snakes and Ladders

Y. Chauhan¹ S. Ladhe¹ S. Chinchkar¹ S. Kumar¹

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

18th October, 2023



Table of Contents

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

Conclusion

Dependencies
Future Aspects
Conclusion

- 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
- 5 Conclusion
 - Dependencies
 - Future Aspects
 - Conclusion



Introduction

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

Conclusion

Dependencies
Future Aspects
Conclusion

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

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

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

Conclusion

Dependencies
Future Aspects
Conclusion

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

Functional Requirements

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

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

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

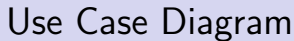
Conclusion

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.



Yuvraj,
Shreyas,
Sneha, Suraj

Software Requirement Specifications

Functional Requirements

Non-Function Requirements

Use-Case Diagram

OOA

- Identifying Classes
- Inherited Classes
- Class Diagram

Conclusion

- Dependencies
- Future Aspects
- Conclusion

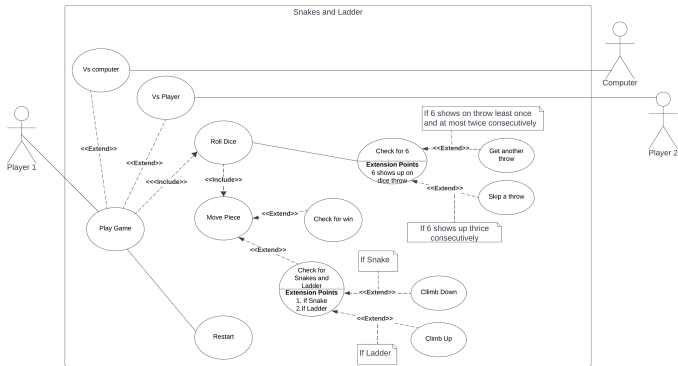


Figure: Use Case Diagram



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

Conclusion

Dependencies
Future Aspects
Conclusion

1. Front End

Front End

- React JS
- CSS (Tailwind)

2. Back-End

3. Version Control

4. UI/UX



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

Conclusion

Dependencies
Future Aspects
Conclusion

1. Front End
2. Back-End

Back End

- Django

3. Version Control
4. UI/UX



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

Conclusion

Dependencies
Future Aspects
Conclusion

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

Version Control

- Git
- GitHub

4. UI/UX



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

Conclusion

Dependencies

Future Aspects

Conclusion

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

UI/UX

- Canva
- Figma



Object Oriented Analysis

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

Conclusion

Dependencies
Future Aspects
Conclusion

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



Identifying Classes

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

Conclusion

Dependencies
Future Aspects
Conclusion

- Player
- Dice
- Cell
- Board
- Game



Inherited Classes

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

Conclusion

Dependencies
Future Aspects
Conclusion

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

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

Conclusion

Dependencies
Future Aspects
Conclusion

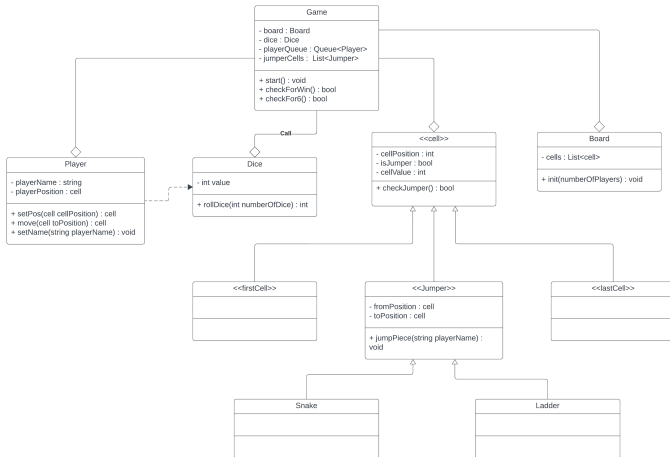


Figure: Class Diagram



Dependencies

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

Conclusion

Dependencies

Future Aspects
Conclusion

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



Future Aspects

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

Conclusion

Dependencies
Future Aspects
Conclusion

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



Conclusion

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

Conclusion

Dependencies
Future Aspects
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.