

## 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. Chauhan<sup>1</sup>   S. Ladhe<sup>1</sup>   S. Chinchkar<sup>1</sup>   S. Kumar<sup>1</sup>

<sup>1</sup>Computer 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.

# 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

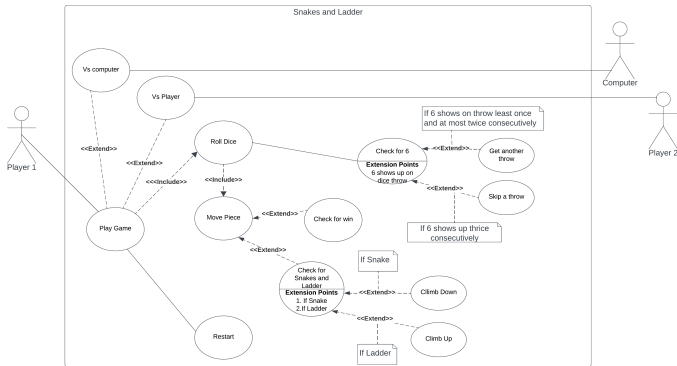


Figure: Use Case Diagram



# General Functionality

## 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

- 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.



# 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



# 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

### Contributors

- Shreyas Ladhe
- Sneha Chinchkar



# 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
- Python



# 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

## 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

## Conclusion

Dependencies  
Future Aspects  
Conclusion

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

## Version Control

- Git
- GitHub



# 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

## 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

## Conclusion

Dependencies

Future Aspects

Conclusion

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

## UI/UX

■ Canva





# 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

## Contributors

- Shreyas Ladhe



# 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

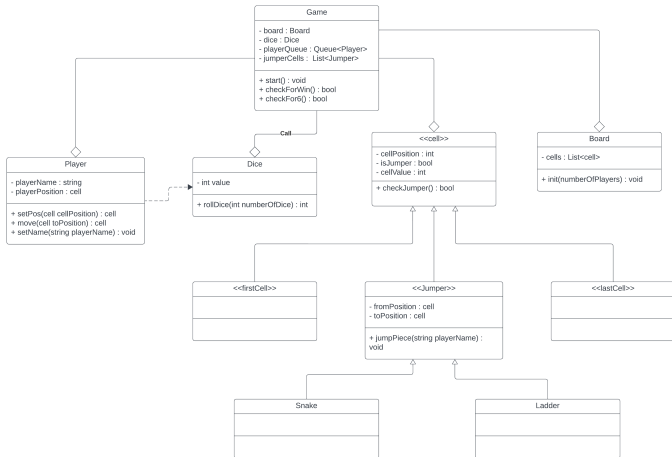


Figure: Class Diagram



# 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.