Project Report

Table of Contents

1. Introduction

The Geometric Shapes Drawing Application project demonstrates object-oriented programming (OOP) principles through a practical drawing application. Developed using Java Spring Boot and React.js, the project involves designing a geometric shapes data model, creating an interactive GUI, and implementing features like shape manipulation and persistent storage. This report outlines the design and implementation process, key design decisions, and provides a guide to using the application.

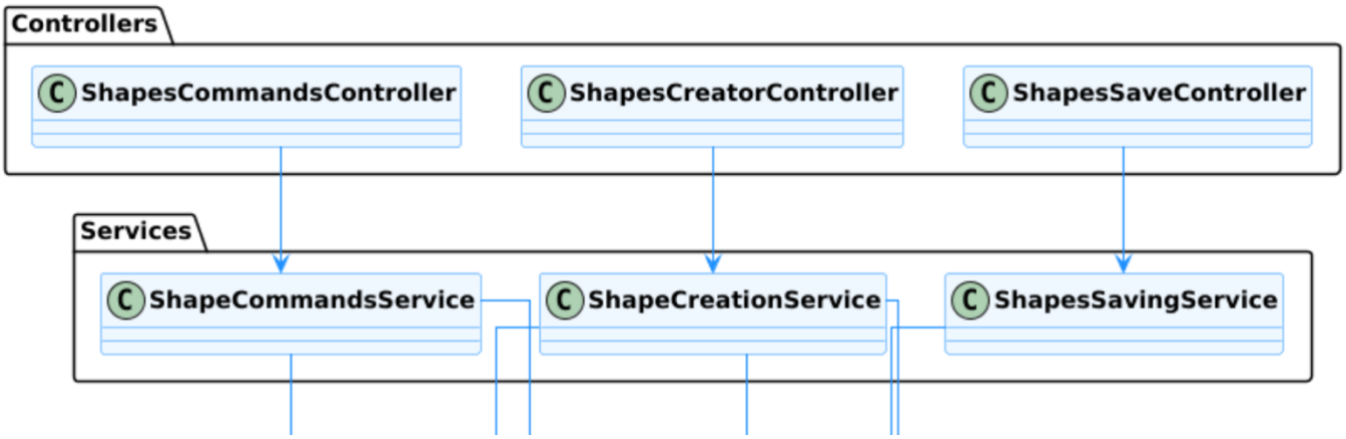
2. Technologies Used

- **Frontend**
  - **React : For building a dynamic and responsive user interface.**
  - **React-Konva : For advanced canvas manipulation.**
  - **React-Color : For a user-friendly color picker.**
  - **FontAwesome : For visually appealing icons.**
  - **Axios : For making HTTP requests to the backend.**
- **Backend**
  - **Spring Boot : For managing RESTful APIs and handling backend logic.**
- **Data Format**
  - **JSON: Drawings are saved as JSON files for easy storage and reusability.**

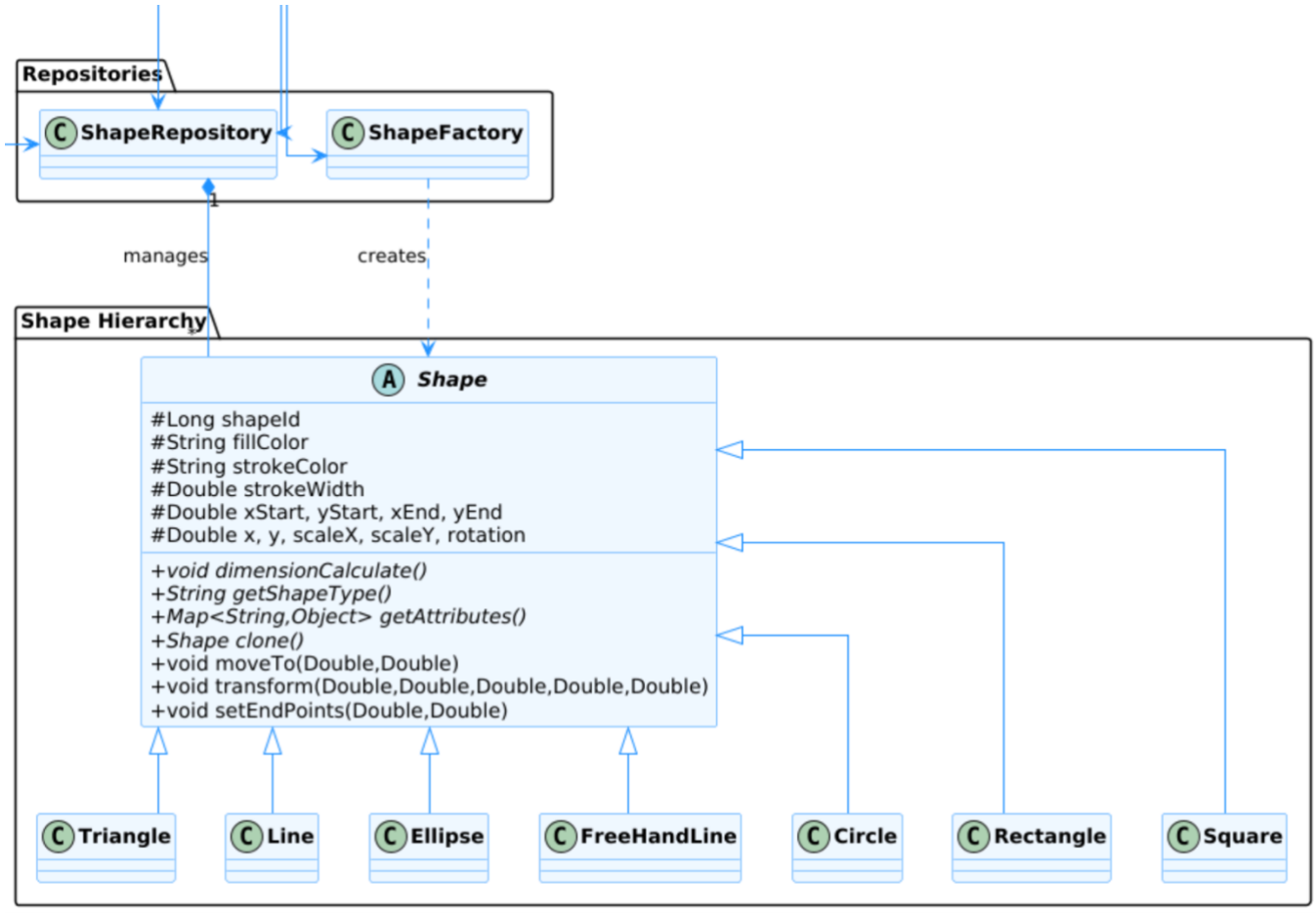
3. UML Diagram

- **Simplified UML Link: https://ibb.co/VDkMTf9**

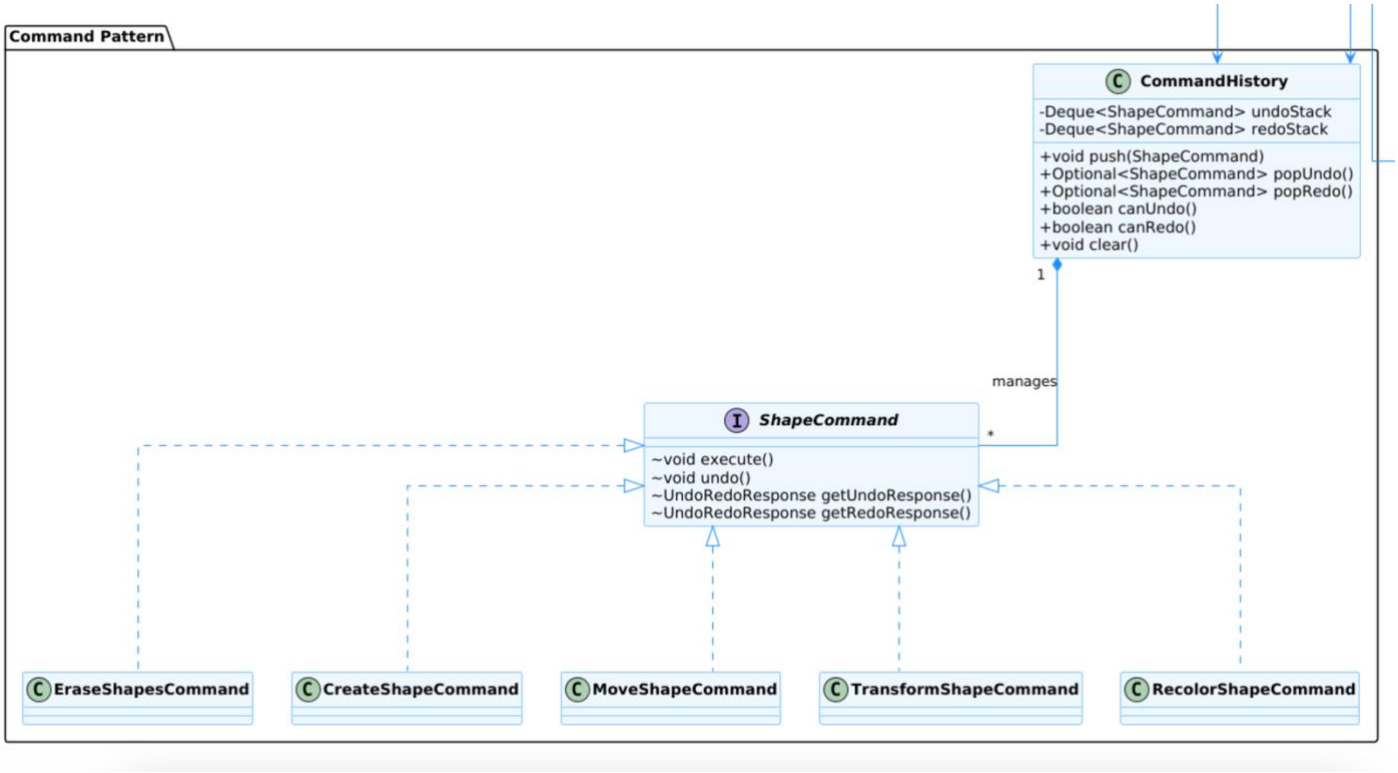
Controllers & Services



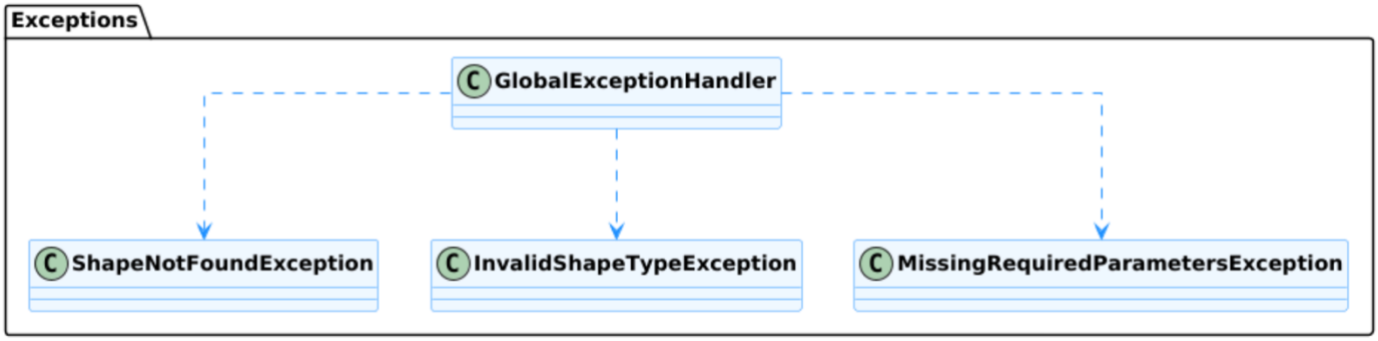
Repositories & Shapes



Commands



Exceptions



- **Detailed UML Link: https://ibb.co/LgFq4QJ**

4. Used Design Patterns

- **Command Pattern**

  The Command pattern is used to encapsulate all actions that can be performed on shapes (e.g., Draw, Color, Resize, Move, Delete). This allows for easy implementation of undo and redo functionality by storing command objects in a dequeue.

- **Prototype Pattern**

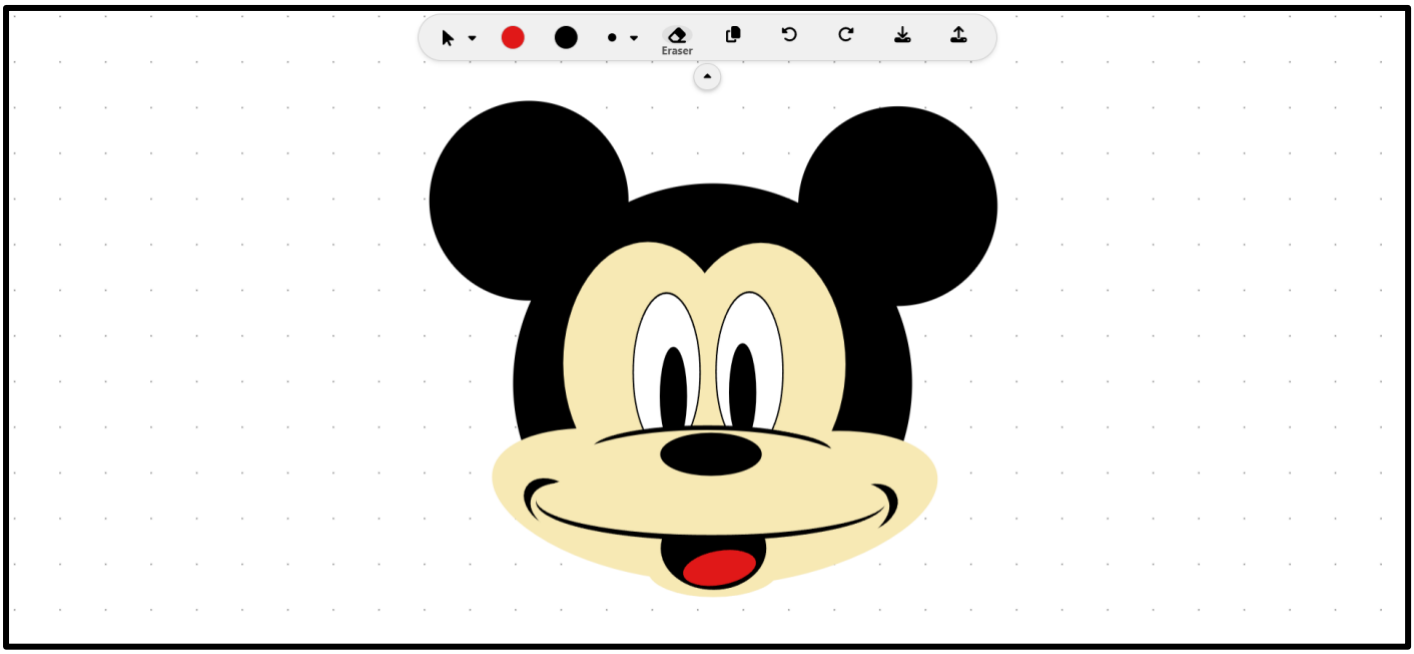
  The Prototype pattern is implemented to efficiently copy shapes. This enables users to duplicate existing shapes without having to recreate them from scratch, improving performance and usability.

- **Factory Pattern**

  The Factory pattern is employed to create different geometric shapes. This design pattern simplifies the creation process and makes the application easily extendable by adding new shapes without modifying existing code.

- **Dependency Injection (springboot)**
- **Singleton Pattern (springboot)**

5. UI Snapshots



Demo link:

https://drive.google.com/file/d/17fLkiejRrd4LN7Vh9MnThk5QejRYCckf/view?usp=drivesdk

6.User Guide

1. **Drawing Shapes:**
    - Select a shape from the toolbar.
    - Click on the canvas to place the shape.
2. **Coloring Shapes:**
    - Select a shape.
    - Choose a color from the color picker.
3. **Resizing Shapes:**
    - Select a shape.
    - Drag the handles to resize.
4. **Moving Shapes:**
    - Select a shape.
    - Drag to move it to a new location.
5. **Copying Shapes:**
    - Select a shape.
    - Click the copy button to duplicate the shape.
6. **Deleting Shapes:**
    - Select a shape.
    - Click the delete button to remove it.
7. **Undo/Redo Actions:**
    - Use the undo and redo buttons to revert or reapply actions.
8. **Saving and Loading Drawings:**
    - Use the save button to save the drawing in XML or JSON format.
    - Use the load button to load a previously saved drawing.