# Shape Viewer - Technical Assignment Submission

- **Pavani Ayanambakam**

## Overview

This submission contains the source code for a browser-based single-page application called **"Shape Viewer"**. The application allows users to import a local .shapefile containing two-dimensional shape data and renders the shape(s) on the screen.

The application is built using:

- **React** – For building the user interface as a single-page application (SPA).
- **Konva** – For rendering and manipulating 2D shapes on the canvas.
- **GitHub** – Used for version control, collaboration, and managing the project repository.
- **Vercel** – The project is deployed on Vercel for easy access and live preview.
  **Live Demo:** https://shape-viewer-rust.vercel.app/

## My Project Structure

```
src/
│— components/
│   ├── ShapeViewerPage.js → Renders shapes on the Konva canvas
│   ├── NewShapes.js → UI for creating new shapes
│— App.js → Main application component
│— App.css → Styling for the application
│— index.js → Entry point for the React app
```

## Features Implemented

- **Iteration 1: User Interface Layout:**
  Implemented the top toolbar, left menu, and shape viewport as specified in the assignment. The shape viewport dynamically adjusts to fill the available screen space.
- **Iteration 2: Shape File:**
  Developed a custom, text-based .shapefile format to store shape data (see "Shape File Format" section below). The application successfully opens and loads the contents of a shape file into memory.

- **Iteration 3: Shape Rendering:**
  Implemented rendering for rectangles and triangles based on the data in the loaded shape file.
- **Iteration 4: Polygon Support:**
  Extended the shape file format and rendering to support polygons with an arbitrary number of vertices.
- **Iteration 5: UI Enhancement:**
  Improved the user interface with a modern design, focusing on visual appeal and user-friendliness (consistent spacing, typography, and color scheme).

---

**Bonus Features Implemented**

- **Bonus Feature 1: Shape Translation:**
  Users can click and drag shapes within the shape viewport to translate (move) them.
- **Bonus Feature 2: Shape Creation:**
  A "Create Shape" section in the left menu allows users to create new rectangles, triangles, or polygons by specifying their type, size, position, and color.
- **"Save As" Functionality:**
  Added a "Save As" option to the left menu, allowing users to save the edited shape file with a new name.

---

**Technologies Used**

- **React**
- **Konva**
- **Font Awesome** (for icons)
- **CSS**

---

**Shape File Format**

The .shapefile format is a text-based format where each line represents a shape. The format is comma-separated.

**Shape Definitions:**

- **Rectangle:**
  Rectangle, x, y, z, width, height, color
- **Triangle:**
  Triangle, x, y, z, width, height, color

- **Polygon:**(for n sides)
  Polygon, $x_1$, $y_1$, $x_2$, $y_2$, ..., $x_n$, $y_n$, color

---

**Example:**

Rectangle, 10, 20, 0, 100, 200, ff0000
Triangle, 150, 250, 0, 120, 180, 00ff00
Polygon, 300, 350, 400, 450, 500, 350, 600, 400, 0000ff
Polygon, 50, 50, 100, 20, 150, 50, 130, 80, 70, 80, 8a2be2
Polygon, 200, 200, 250, 150, 300, 200, 270, 230, 230, 230, ffa500
Rectangle, 350, 250, 0, 100, 180, 32cd32
Triangle, 500, 300, 0, 130, 190, dc143c

---

**Instructions to Run the Application**

1. Extract the contents of the provided .zip file.
2. Navigate to the project directory in your terminal.
3. Run npm install to install the required dependencies.
4. Run npm start to start the development server.
5. Open your web browser and navigate to http://localhost:3000.

---

**Major Known Problems**

- Currently, the **Z-index** property in the shape file is parsed but not actively used in the rendering logic. This is included for future implementation of layering.
- I need better feedback for **invalid shape files**. Need to test the application with various different shapefiles and accordingly improve error handling.
- The application's **performance with extremely large numbers of shapes (100,000+)** has not been fully optimized. While Konva provides good performance, further optimization might be necessary for such extreme cases.

---

**References**

- React Documentation
- Konva Documentation
- Font Awesome

---

**Input Files**

I've included 5 sample shape files with **.shapefile extension** under the public folder of the project. We can use these files to test the application's loading and rendering capabilities.