Appendix C

Front End Engineering-II

Project Report

Semester-IV (Batch-2022)

STELLAR SNAP

Supervised By:

Dr. Yogesh

Assistant Professor, CUIET

Submitted By:

Pearl 2210990648(G8)

Prachi Anand 2210990660(G8)

Prachi Malik 2210990661(G8)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

Stellar Snap is an innovative social media platform designed to transform the way users discover, collect, and share visual content, drawing inspiration from the popular model of Pinterest, Pix. This platform aims to provide a seamless and engaging user experience through its advanced search algorithms and personalized recommendation systems, ensuring that the content presented aligns closely with individual user interests and preferences. At the heart of Stellar Snap's functionality is its dynamic visual boards, which allow users to organize and curate content in a highly intuitive manner. These boards serve as digital canvases where users can pin images, videos, and articles, creating collections that reflect their unique tastes and hobbies. The platform also supports following and interacting with curated collections from other users, fostering a sense of community and shared inspiration.

Stellar Snap aims to build a global network of inspiration seekers and creators, bridging geographical and cultural divides through the universal language of visual storytelling. By offering a space where creativity can flourish and connections can be made, Stellar Snap aspires to become a go-to destination for anyone looking to explore new ideas, trends, and inspirations.

Table of Content

1.Introduction

1.1 Background

Stellar Snap emerged as a visionary response to the growing demand for a more personalized and interactive visual discovery platform. Inspired by the success of Pinterest, Stellar Snap seeks to elevate the concept of visual social media by integrating advanced technological features and fostering a more dynamic user community. The genesis of Stellar Snap can be traced back to the evolving landscape of digital content consumption and the increasing importance of visual storytelling in the online space.

Market Need and Inspiration:

Pinterest, since its inception, has captivated millions with its unique approach to content curation and visual discovery. However, as users' expectations evolved, there arose a need for a platform that could offer even more tailored experiences and interactive features. Stellar Snap was conceived to fill this gap, aiming to deliver a more immersive and engaging experience. The founders recognized the potential to build a platform that not only mirrored Pinterest's success but also introduced innovations that catered to contemporary user preferences.

Technological Advancements:

At its core, Stellar Snap leverages cutting-edge technologies to enhance user experience. The platform employs sophisticated machine learning algorithms and artificial intelligence to analyze user behavior and preferences. This data-driven approach allows Stellar Snap to provide highly personalized content recommendations, ensuring that users are always presented with the most relevant and inspiring content. Additionally, the platform integrates advanced search functionalities that make discovering new content seamless and intuitive.

User-Centric Design:

The design philosophy behind Stellar Snap prioritizes ease of use and aesthetic appeal. The platform features a clean, modern interface that highlights visual content, making it the focal point of the user experience. This design choice ensures that users can effortlessly navigate through the

site, focusing on discovering and sharing visually appealing content without unnecessary distractions.

Community and Collaboration:

Stellar Snap places a strong emphasis on community building and collaborative engagement. Recognizing the value of social interaction in enhancing user experience, the platform includes features that encourage users to connect, share ideas, and collaborate on projects. Users can join interest-based groups, participate in creative challenges, and co-curate boards, fostering a sense of community and shared purpose. These features are designed to promote interaction and collaboration, turning passive browsing into an active and engaging experience.

Security and Reliability:

In an age where digital security is paramount, Stellar Snap is committed to ensuring the safety and privacy of its users. The platform incorporates robust security measures to protect user data and digital assets. This commitment to security, combined with reliable and efficient digital asset management tools, makes Stellar Snap a trustworthy repository for users' curated collections.

Vision for the Future:

Stellar Snap aims to become a global hub for inspiration and creativity, bridging cultural and geographical divides through the universal appeal of visual content. The platform aspires to continuously evolve, integrating user feedback and technological advancements to enhance its offerings. By fostering a vibrant and interactive community, Stellar Snap envisions itself as the go-to destination for anyone seeking to explore new ideas, trends, and inspirations.

1.2 Objective

The objective of Stellar Snap is to revolutionize the visual discovery and social sharing experience by providing an innovative, user-centric platform that leverages advanced technology to offer personalized content recommendations and foster a vibrant, interactive community. Specifically, Stellar Snap aims to:

1**. Enhance Visual Discovery:**

Utilize algorithms to deliver personalized and relevant content to users, ensuring a unique and engaging discovery experience.

2. **Foster Creativity and Inspiration:**

Create a platform where users can explore, curate, and share a diverse range of visual content, thereby fuelling creativity and providing endless inspiration across various interests and hobbies.

3. **Build a Dynamic Community:**

Encourage social interaction and collaboration through features such as group boards, challenges, and interactive projects, fostering a sense of community and shared passion among users.

4. **Provide a Seamless User Experience:**

Design an intuitive, aesthetically pleasing interface that prioritizes ease of use, allowing users to effortlessly navigate the platform and focus on discovering and sharing visual content.

5**. Ensure Security and Reliability:**

Implement robust security measures to protect user data and digital assets, establishing Stellar Snap as a trustworthy and reliable platform for managing personal collections.

6**. Promote Continuous Innovation:**

Stay at the forefront of technological advancements and user feedback to continuously improve the platform's features and functionalities, ensuring that Stellar Snap evolves to meet the changing needs and expectations of its users.

7. **Bridge Cultural and Geographical Divides**:

Build a global network of inspiration seekers and creators, using the universal language of visual storytelling to connect people from diverse backgrounds and foster cross-cultural exchange.

1.3 Significance

Stellar Snap holds significant value in the digital landscape as an innovative visual content discovery and social interaction platform. By leveraging advanced AI and machine learning, it delivers personalized content curation and a seamless user experience, setting new standards for digital engagement. Its dynamic features promote active user participation and creativity, fostering a vibrant and interconnected community. Additionally, Stellar Snap bridges cultural and geographical divides, facilitating global connectivity and cross-cultural exchange. The platform's robust security measures ensure the protection and reliability of users' digital assets, building trust and reliability. Furthermore, Stellar Snap enhances visual storytelling, influences cultural trends, and supports the creative economy by providing creators with a platform to showcase their work. Through these multifaceted contributions, Stellar Snap significantly impacts how users interact with visual content and connect with one another online.

2.Problem Definition and Requirements

2.1 Problem Statement

Despite the popularity and utility of existing visual discovery platforms like Pinterest, there remains a significant gap in providing a truly personalized, secure, and interactive user experience that fosters deep engagement and creativity. Current platforms often lack advanced content curation technologies and comprehensive social interaction features, leading to a one-size-fits-all approach that fails to meet diverse user needs and preferences. Additionally, concerns about data privacy and digital asset management remain inadequately addressed, undermining user trust and security. Therefore, there is a pressing need for an innovative platform that integrates cutting-edge AI for personalized content recommendations, robust security measures, and dynamic community-building tools to revolutionize how users discover, share, and interact with visual content. This is the problem Stellar Snap aims to solve, by creating a user-centric, secure, and engaging visual discovery experience.

2.2 Software Requirements

To develop and maintain Stellar Snap, a robust visual discovery and social interaction platform, a comprehensive set of software requirements is necessary. These requirements encompass both functional and non-functional aspects to ensure the platform's functionality, security, scalability, and user engagement.

Functional Requirements

1. User Registration and Authentication:
- Secure user registration with email verification.
- Social media login options (Google, Facebook, etc.).
- Two-factor authentication (2FA) for enhanced security.

2. Content Creation and Management:
- Ability to create and edit visual boards (collections of images, videos, articles).
- Upload and manage various types of media content (images, videos, GIFs).
- Tagging and categorization of content for easy discovery.

3. Search and Discovery:
- Advanced search functionality with filters (tags, categories, date, popularity).
- Personalized content recommendations using AI and machine learning algorithms.
- Trending content and suggested boards based on user interests.

4. Social Interaction:
- Follow/unfollow users and boards.
- Like, comment, and share content within the platform and on external social media.
- Direct messaging between users for private interactions.

6. Collaboration Features:
- Group boards where multiple users can contribute.
- Collaborative projects and challenges with voting and feedback mechanisms.

7. Notifications:
- Real-time notifications for likes, comments, follows, and messages.
- Email notifications for significant updates and activities.

8. Content Moderation:
- Reporting and flagging inappropriate content.
- Admin panel for reviewing and moderating reported content.

Technical Stack Used:

1. **HTML:** HTML is the standard markup language used for creating web pages. It structures the web content by using a system of elements and tags to define headings, paragraphs, links, images, and other types of content.

2. **CSS:** CSS is a stylesheet language used for describing the presentation of a document written in HTML or XML. It allows you to control the layout, colors, fonts, and overall visual style of web pages, making them look attractive and user-friendly.

3. **JS:** JavaScript is a high-level, dynamic programming language commonly used in web development to create interactive effects within web browsers. It enables the implementation of complex features on web pages, such as dynamic content updates, form validations, animations, and more.

4. **React JS:** React JS is a popular JavaScript library for building user interfaces, particularly for single-page applications. Developed by Facebook, it allows developers to create large web applications that can update and render efficiently in response to data changes. React uses a component-based architecture and a virtual DOM for performance optimization.

5. **Local Storage:** Local Storage is a web storage feature in modern web browsers that allows websites to store data persistently in the user's browser. This data is stored in key-value pairs and remains accessible even after the browser is closed and reopened. Local Storage is commonly used to save user preferences, session information, and other data that needs to persist across sessions.

2.3 Hardware Requirements

1. Developer Workstations:

- Processor: Intel Core i5 or AMD Ryzen 5
- RAM: 16GB minimum
- Storage: 512GB SSD
- Graphics: Dedicated GPU (optional, for design and front-end development tasks)
- Operating System: Windows 10/11, macOS, or Linux

2. Local Development Servers:

- Processor: Intel Xeon or AMD EPYC
- RAM: 32GB
- Storage: 1TB SSD
- Networking: Gigabit Ethernet

3. End User Workstations:

- Processor: Intel Core i3 or AMD Ryzen 3
- RAM: 8GB minimum
- Storage: 256GB SSD
- Graphics: 1GB Minimum
- Operating System: Windows 10/11, macOS, or Linux

3. Proposed Design and Methodology

3.1 Methodology

I. Project Setup

Framework and Libraries: The project is developed using React.js, a popular JavaScript library for building user interfaces. And used Create React App (CRA) to set up the project, ensuring a well-structured and standardized development environment.

Version Control: Git is used for version control, with GitHub as the repository hosting service. This facilitated collaborative development and maintain a history of changes.

Dependencies Management: npm (Node Package Manager) is used to manage project dependencies, ensuring that all necessary libraries and packages are up-to-date.

II. Component Development

Component-Based Architecture: The project used a component-based architecture, breaking down the UI into reusable and manageable components. Where each component will be developed as a separate JSX file.

State Management: React's useState and useContext hooks is used for managing state of different components.

III. Design and Styling

CSS Modules: CSS Modules or styled-components is used to style individual components, promoting modularity and preventing style conflicts.

IV. Routing

React Router: React Router will be used to handle client-side routing, enabling navigation between different views (e.g., login, signup, dashboard) without refreshing the page.

Proposed Design

I. Component Breakdown

App.js: The root component that sets up routing and global context providers. It will also include any necessary setup for state management and global styles.

Window.jsx: A higher-order component that acts as a container for various views or sections of the application. It might handle layout and common functionality shared across different parts of the app.

Card.jsx: A reusable component to display data in a card format. This component will be used across different views to present information in a consistent manner.

Login.jsx: A form component for user authentication. It will handle user input, form submission, and validation. It will interact with the backend to verify user credentials.

Signup.jsx: Similar to Login.jsx, this component will handle user registration. It will collect user details, perform validation, and submit the data to the backend to create a new user account.

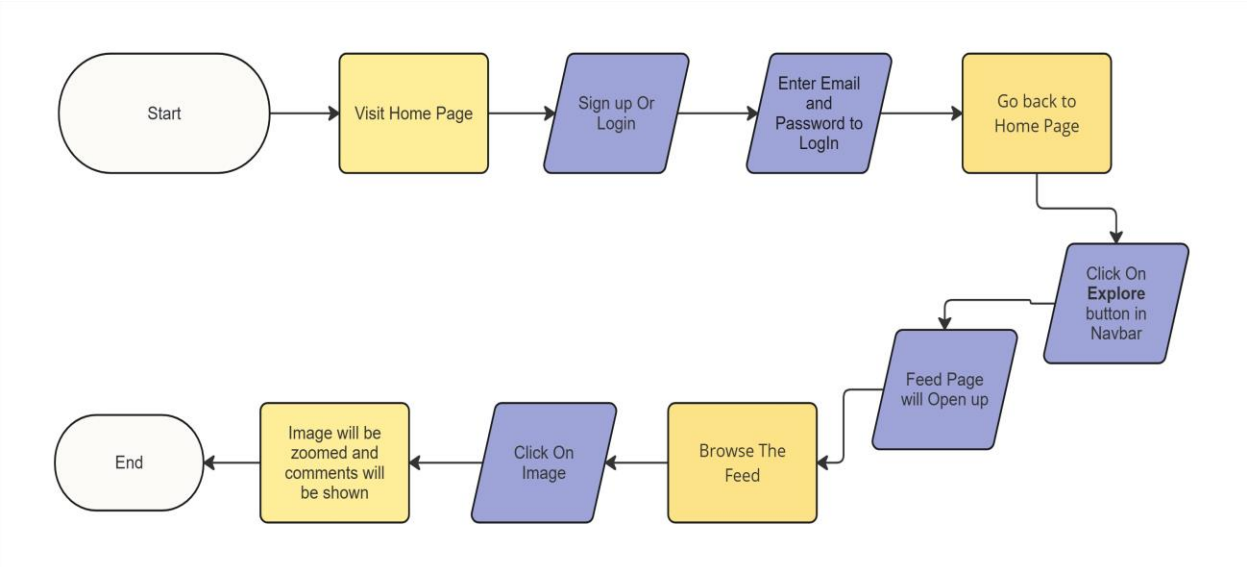
Navbar.jsx: A navigation bar component that provides links to different parts of the application. It will update based on the user's authentication status (e.g., showing different links for logged-in users).

Data.jsx: A component responsible for fetching and displaying data from the backend. It could be a dashboard or any view that requires data retrieval and presentation.

Shared.jsx: A utility component that contains shared functionalities, utilities, or styles that are used across multiple components.

Users.js: A component or module that handles user-related data and functionalities, such as user lists, user profiles, or user-specific settings.

II. User Interface Flow



When a user first visits the website, they land on the home page, marking the initial startup of the site. From here, they have the option to log in or sign up based on their preferences. If the user chooses to sign up, they will need to provide necessary details such as their email address and password. Once logged in, the user is directed to their personalized feed, which is populated with a variety of posts. This feed refreshes every time the user reloads the page, ensuring they always see the latest content.

Within their feed, users can browse through posts and engage with them by liking, commenting, and sharing. This interactive experience allows them to connect with the content and other users dynamically. Finally, when they are done, users can log out. They have the flexibility to log back in whenever they wish, returning to their feed and resuming their interactions with the posts.

Authentication Flow:

The user is presented with the Login.jsx component upon accessing the application.
If the user does not have an account, they can navigate to Signup.jsx via a link in Navbar.jsx.
Upon successful login or signup, the user is redirected to a protected route that renders **Window.jsx** containing the main application view.

Main Application Flow:

Navbar.jsx is displayed at the top, providing navigation links to different sections like dashboard, profile, settings, etc.
Data.jsx fetches and displays user-specific or general data within **Feed.jsx**
Card.jsx is used within various views to present data in a structured format. As name implies in Cards.

State Management and Data Flow:

App.js sets up global state using Context API or Redux, making state accessible to all components. Data fetching components (e.g., Data.jsx) use **useEffect** to retrieve data when mounted, updating the state and triggering re-renders. Data is fetch through Local Storage in the form of List of Objects in String form and retrieves it using **.getItem(key,value)** by parsing the string data to JSON providing the necessary components.

Algorithm Used:

Randomizer:

```
const shuffleArray = (array) => {
  let shuffledArray = array.slice();
  for (let i = shuffledArray.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    const temp = shuffledArray[i];
    shuffledArray[i] = shuffledArray[j];
    shuffledArray[j] = temp;
  }
  return shuffledArray;
};
```

In this, An Array is shuffled in random manner from the 0^{th} index to the length of Data. Hence, creating a array of unique array index elements every time the function run. As the unique index is stored in the array, meaning no repeating elements. This is a perfect algo to create the feed, Making sure that post appears only one single time.

Generating Cards:

```
<div className="container">
  {shuffledData.map((shuffledItem) => (
    <Card key={shuffledItem.id} post={shuffledItem} user={shuffledItem.user} />
  ))}
</div>
```

Using that shuffle algorithm it returns a random array. We generate cards given by their randomized id. Making the Feed interesting, randomized every single time. Hence, User will not get bored of it.

4. Results:

Home Page:

This is our Landing Page of our Website. Consisting of attractive elements defining what the website is all about. Providing Tabs like: Today, Watch, Explore, Log In, Sign Up, About, Blog etc. Log In, Sign Up are 2 separate Components which appears when User Clicks on it.

Code:

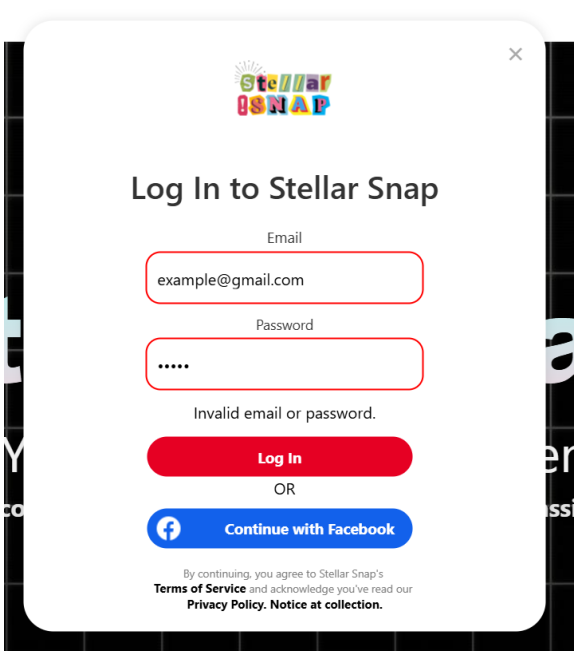
```
import React, { useState } from "react";
import "./Home.css";
import logo from "./images/logo.png";
import Login from "./Components/Login";
import Signup from "./Components/Signup";

const Home = () => {
  const [loginVisible, setLoginVisible] = useState(false);
  const [signupVisible, setSignupVisible] = useState(false);
      You, 3 weeks ago • Reactizing Phase 2
  const toggleLogin = () => {
    setLoginVisible(!loginVisible);
    setSignupVisible(false);
  };

  const toggleSignup = () => {
    setSignupVisible(!signupVisible);
    setLoginVisible(false);
  };

  const handleLogout = () => {
    localStorage.setItem("authToken","NULL");
    window.location.href = "/";
  }
```

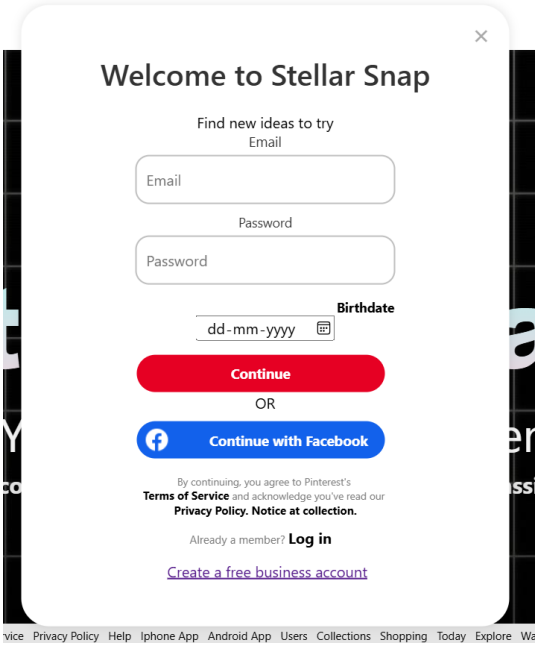
The Home page is consist of Landing page design in form of JSX & consists of 2 Components: **Login.jsx** & **SignUp.jsx**. And their states are managed by the 2 useStates, so that we can control the visibility of the components using conditional formatting. Initially, we both set them as false. And make them appear on when the clicked on the Login or Sign Up Buttons.

```
{loginVisible && <Login toggle={toggleLogin} />}
{signupVisible && <Signup toggle={toggleSignup} />}
```

Whenever clicks on the any of the buttons, There states are changes and when all of the conditions come true. In this case, component is available and state is true, the Components renders.

Login & Sign Up Page:





Above 2 Components are Login and Sign Up Components rendered when the User Clicked on it. The Login and Sign Up comes with basic user authentication and validation, making sure that only authorised and validate info is stored or accessed on the website in the Local Storage. Like in Login, If a non-Signed Up user try to access the website or a User put-in the wrong password. The Border of the input box turned red, and an error message is appeared just on upper side of the Login Button, indicating User don't exist or Invalid Email or Password.

Code:

```
import React, { useState } from "react";
import "./Login.css";
import Logo from "../images/logo.png";

const Login = (props) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");

  const handleLogin = (e) => {
    e.preventDefault();

    try {
      const users = JSON.parse(localStorage.getItem("users")) || [];

      let user = null;
      let i=0;
      for (i = 0; i < users.length; i++) {
        if (users[i].email === email && users[i].password === password) {
          user = users[i];
          break;
        }
      }

      if (user) {
        localStorage.setItem("authToken", JSON.stringify({ userID: i }));
        window.location.href = "/feed";
```

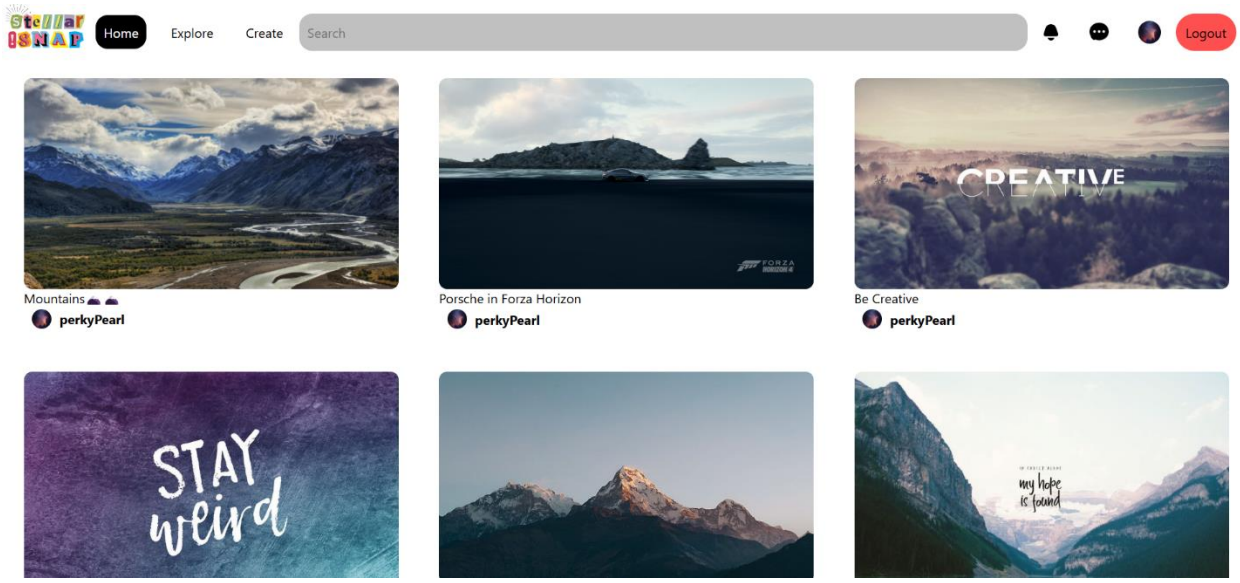
Login consist of 2 Fields as a input. And to manage does inputs, we have 2 states: email and password which is inputted by the user. Login Page also utilizes a error state which will take care of the error message that need to be displayed. It also consists of User authentication meaning that only the signed up & authentic user logs In. When the user submits the details, the users are fetched from local Storage, and the email and password is compared with all the users, If the details matched with the dataset. The User got authenticated and transferred to the feed page. If none of the things matched or the only the email is the matched, An Error message is displayed.

```
import React, { useState } from "react";
import "./Signup.css";

const Signup = (props) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [birthdate, setBirthdate] = useState("");
  const [error, setError] = useState("");

  const handleSignup = async (e) => {
    e.preventDefault();

    const newUser = {
      email: email,
      password: password,
      birthdate: birthdate,
    };

    try {
      const existingUsers = JSON.parse(localStorage.getItem("users")) || [];

      const userExists = existingUsers.some((user) => user.email === email);
      if (userExists) {
        setError("User with this email already exists.");
        return;
      }
```

Sign Up consists of 3 Fields as a input. And to manage does inputs, we have 3 states: email, password & DOB which is inputted by the user. Sign Up Page also utilizes a error state which will take care of the error message that need to be displayed. It consist of basic User validation process like '@' is present in the email or not and letting one email used by the one user only. And this all works with help of Local Storage. Every time a User signs up, A new User object is created with the details provided, and that object is then append to the already existing user which are fetched from the Database, in this case, Local Storage on if the user's email is unique. Else, It will display an error, prompting the user that the email there are using is already in use.

Also in Sign Up, It checks for the similar email is used on the account or not, making sure that 2 users don't have single type of mail. And If the User Signs up, Their Info is updated in the Database which in this case Local Storage.

Feed:

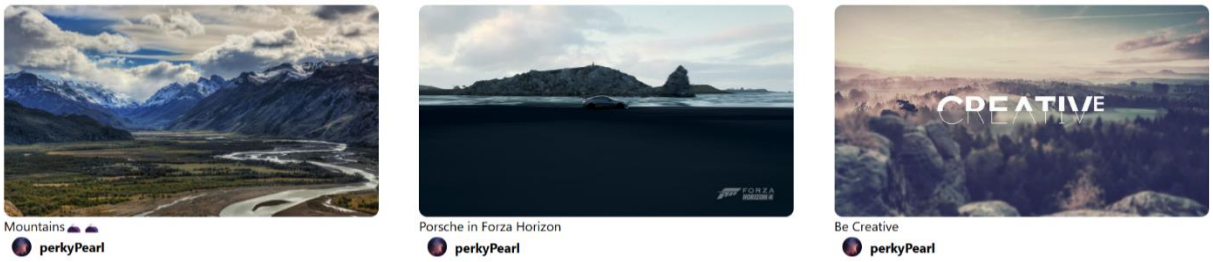


After User Logs in with their Credentials, The User is redirect to the feed. The Feed utilizes the Card components showing number of Posts in random order. The Card shows, Image, Caption and User Profile providing the basic info of the Post. The Random algo makes sure that the cards shown in random order, so that the user doesn't got bored of watching same feed. The Card's Images are fetched from the Database in this Local Storage. And Then rendered Randomly.

```
import React, { useState, useEffect } from "react";
import Navbar from "./Components/Navbar";
import Card from "./Components/card";
import "./Feed.css";

const Feed = () => {
  const [data, setData] = useState([]);

  const shuffleArray = (array) => {
    let shuffledArray = array.slice();
    for (let i = shuffledArray.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      const temp = shuffledArray[i];
      shuffledArray[i] = shuffledArray[j];
      shuffledArray[j] = temp;
    }
    return shuffledArray;
  };

  useEffect(() => {
    const storedData = localStorage.getItem('cardsData');
    if (storedData) {
      const parsedData = JSON.parse(storedData);
      setData(parsedData);
    }
  }, []);
```

The Feed is generated by loading the Card components with the details which are fetched from the Database, in this case Local Storage. When the data is fetched, it is fetched on array of objects. And that array if feed into a randomizing function called shuffleArray, It shuffles the array in such that it always different from the previous one and it always consists of unique elements. Making the Feed unique and interesting every time we reloads it.

Card:

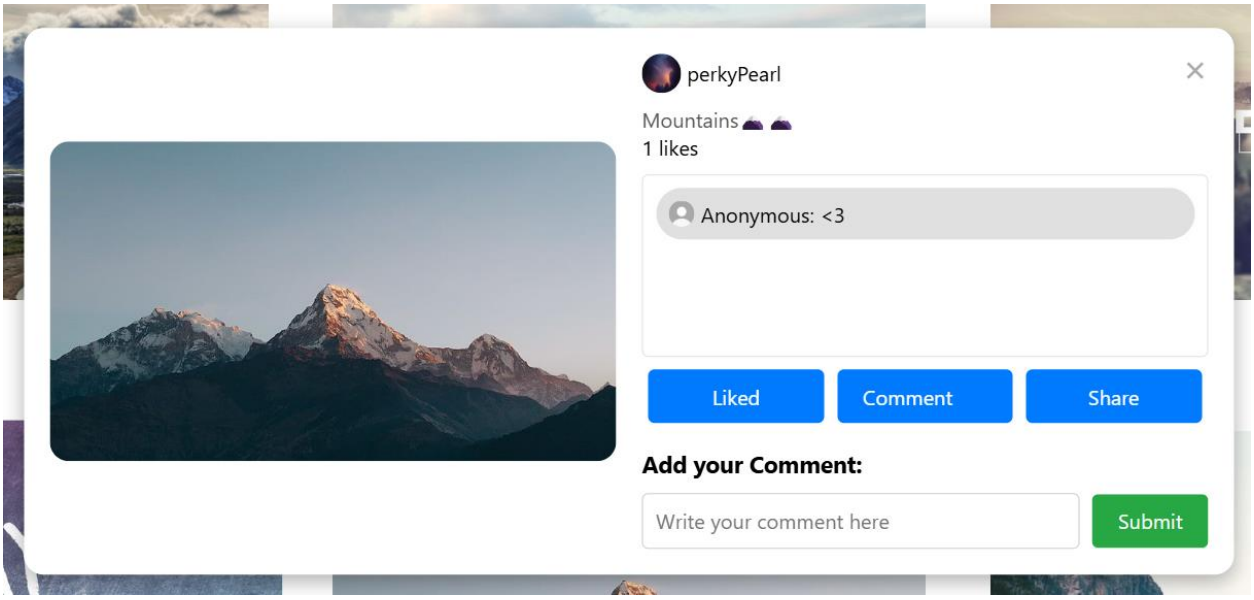


On the Feed, We can see multiple cards arranged in the ways displaying multiple posts in the single screen in minimal and organized way with Image, Caption and user who uploaded it.

```
import React, { useState } from "react";
import "./card.css";
import Window from "./Window";

const Card = (props) => {
  const { Username, PFP } = props.user;
  const [showWindow, setWindow] = useState(false);

  const toggle = () => {
    setWindow(!showWindow);
  };

  return (
    <>
      <div className="card">
        <img src={props.post.image_src} onClick={toggle} className="card-img" />
        <div className="caption">{props.post.caption}</div>
        <div className="user">
          <img src={PFP} className="usr-img" />
          <h5>{Username}</h5>
        </div>
      </div>

      {showWindow && <Window post={props.post} user={props.user} toggle={toggle} />}
    </>
  );
};
```

The Card component takes post's details passed as an argument. And it in renders in the Card Components. For the Focused View of it, We also imported the Window Component that displays the Posts in the focused View with interactable buttons. The Window Component is associated with the every Card Component Rendered. And Only Rendered when the User clicks on it, by calling the **setWindow()**, setting the state of **showWindow** from False to True or Vice Versa. Their states are managed by the **useState** hook. And Rendered using Conditional Rendering, meaning that for rendering of the Window Component, The Component will only render, if the component is available to render & the state of it sets to True.

Focused Card:

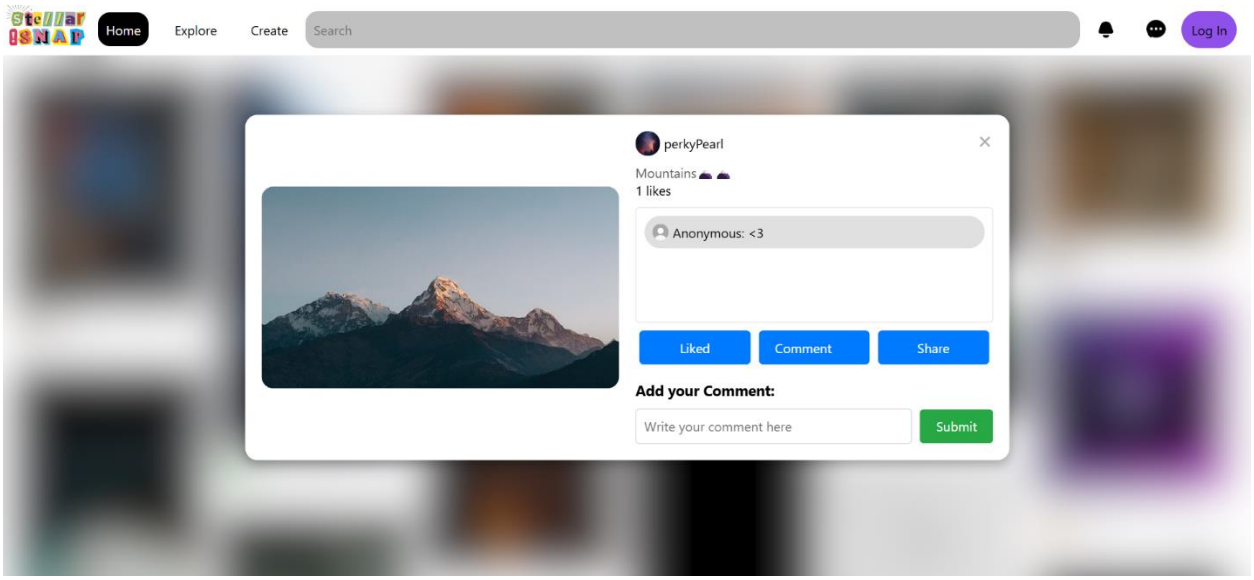


When the User clicks on the Specific Post, A pop is pops up showing the contents of the Post and Interactable buttons. The User can like, Comment and Share the post, Making the our Website Engaging for users. The Use of Local Storage to keep the Track of all the Posts, helps it to remember all the comments, Likes on the specific posts.

```
import React, { useState, useEffect } from "react";
import "./Window.css";
import user from "../images/user.jpg";
import anonymousUser from "../images/User Image.jpg"

const Window = ({ post, toggle }) => {
  const [comment, setComment] = useState("");
  const [comments, setComments] = useState([]);
  const [likes, setLikes] = useState(0);
  const [liked, setLiked] = useState(false);

  useEffect(() => {
    const storedComments = localStorage.getItem(`comments_${post.id}`);
    if (storedComments) {
      setComments(JSON.parse(storedComments));
    }

    const storedLikes = localStorage.getItem(`likes_${post.id}`);
    if (storedLikes) {
      setLikes(JSON.parse(storedLikes));
    }

    const storedLiked = localStorage.getItem(`liked_${post.id}`);
    if (storedLiked) {
      setLiked(JSON.parse(storedLiked));
    }
  }, [post]);
```

Window component takes post and toggle as an argument, to control the visibility of the component as it plays a crucial role display in the post in enlarged, focused with interactable buttons. It consists of Like, Comment & Share button for user Interactions with the post. For that we store the states of like, comment. So that, the website renders the card with the appropriate details. The details of the Post is fetch by getting it as an argument, displaying the images, likes, comments. And using the magic of Local Storage, We can fetch and store the details of the likes and comments on that posts. Also, A share button is also implemented, adding the shareable ability. What it does do, It creates a link to the past using its id and copys that to clipboard. And whenever the share button is click, the link copy to it and User can share it with anyone by simply sending it.

Shared Post:



When the User, clicks on the Share button. A URL Link, linked to that post is copied to the Clipboard. Which a user can share with anyone, and when accessed. The User is direct to the Post which is being shared. Hence, Increasing the Engagement.

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import Navbar from './Components/Navbar';
import Window from './Components/Window';
import cardsData from "./Data";
import "./Components/Window.css"
import Login from "./Components/Login"

const Post = () => {
  const { postId } = useParams();
  const [post, setPost] = useState(null);
  const [showLogin, setShowLogin] = useState(false);

  useEffect(() => {
    const filteredPost = cardsData.find(card => card.id === parseInt(postId));
    setPost(filteredPost);
  }, [postId]);

  const handleLoginToggle = () => {
    setShowLogin(true);
  };

  return (
    <>
      <Navbar />
      <div className='Background-Blured' onClick={handleLoginToggle}>
        {post && <Window post={post} />}
```

Shared post's details are retrieved from the url using React hook called **useParams**. It read the id given in the Url and then searches that post in the Database. And when the post is found, the post object then passed in window component. Thus, Rendering the Post in the Focused View with appropriate info.

About:

Other than the Home Page, Our Website also offers an About Page, which helps the user to know about our website. It consists of various components, sharing the Idea about the website like Collaborators of the Website, Insights of it and much more.

```
import React, { useState } from "react";
import "./About.css"
import Login from "./Login";
import Signup from "./Signup";
import logo from "../images/logo.png"
import img1 from '../images/[removal.ai]_cb435d56-347c-4a5c-972e-5b079a2745b8-easter-bunny-fairytale-tubikarts.png'
import img2 from '../images/[removal.ai]_cad43923-b784-4794-a368-d89c980207cc-800b78cf961d8624e3f25c98f65bf620.png'
import img3 from '../images/vision-4352694-3611155.webp'
import img4 from '../images/[removal.ai]_f8f21d96-0d29-4368-bf95-4e6947f5fbbb-job-offer-abstract-concept-vector-illustration-2
import girl from '../images/[removal.ai]_c135d541-4ede-4b28-ac01-182d33905427-images.png'
import boy from '../images/man_9-512.webp'
      prachi9124, yesterday • About us page made and css modified
const About = () => {
    const [loginVisible, setLoginVisible] = useState(false);
  const [signupVisible, setSignupVisible] = useState(false);

  const toggleLogin = () => {
    setLoginVisible(!loginVisible);
    setSignupVisible(false);
  };

  const toggleSignup = () => {
    setSignupVisible(!signupVisible);
    setLoginVisible(false);
  };
    return (
        <>
```

Same as the Homepage, It consists of all the elements of the about page in JSX form and Returns it from the Functional Component **About**. It also intergrates 2 more Components **Login** & **Sign Up** Component, Rendering through Conditional Rendering. Their states are stored in the **useState** hooks named loginVisible and signupVisible. And their state changing function **setLoginVisible** & **setSignupVisible**. Whenever a Login or Signup button is pressed, their respective toggle Function is called, changing the state of the Components.

Reference:

- W3Schools: https://www.w3schools.com/
- React: https://www.react.dev
- Google: https://www.google.in/
- FreeCodeCamp: https://www.freecodecamp.com