Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
An Autonomous Institute Affiliated To University Of Mumbai
Munshi Nagar,Andheri (W) Mumbai 400 058

C.I.T.L. EXPERIMENT 5

Submitted By:

Akash Panicker	2021300089
Mahesh Patil	2021300095
Rohit Phalke	2021300100
Adwait Purao	2021300101

Submitted To:

Prof. Sunil Ghane

Inventory Management System

Aim:
Create a web mashup of web services using open source framework

Problem Statement:

Develop an inventory management system for a retail store that efficiently tracks and manages the inventory of products. The system should provide real-time updates on stock levels, generate alerts for low stock items, enable easy addition and removal of products, and offer insights into sales trends to optimize restocking decisions.

Theory:

A web service is a software system designed to allow different applications to communicate with each other over the internet or a network.
It provides a standardized way for software components, regardless of their programming languages or platforms, to exchange data and perform various tasks.
Web services enable the integration of different systems, making it possible to create distributed and interoperable applications.
APIs that provide services that a normal developer couldn't feasibly implement offer a wide range of benefits. These APIs typically serve as a bridge between developers and complex, specialized services or data sources.

Web Mashup:

A web mashup is a web application or web page that combines content and functionality from multiple sources or web services to create a unified and enriched user experience.
In a web mashup, data or services from different websites or APIs are integrated to provide users with a single, cohesive interface and access to various types of information or services.
The term "mashup" comes from the idea of mixing or combining elements from various sources, similar to how DJs create music remixes by blending different tracks.

Benefits of Web mashup:

1. Web mashups consolidate data from multiple sources for user convenience.
2. They create a more user-friendly and engaging experience with unified interfaces.
3. Mashups aggregate data from diverse providers into a single platform.
4. Users can customize web mashups to match their specific needs and preferences.
5. They stimulate innovation and creativity among developers.
6. Mashups serve as a single entry point for accessing distributed information and services.
7. Developers benefit from efficiency gains by reusing existing APIs and services.
8. They are compatible with various platforms, ensuring widespread accessibility.
9. Mashups offer economic value for businesses, improving user engagement and retention.
10. Real-time data integration is a strong suit of web mashups, making them ideal for applications requiring dynamic information.

Types of Web Mashup:

1. Server-Side Mashups:
    - Server-side mashups involve data integration and processing on the web server that hosts the application.
    - The server retrieves data from various sources, processes it, and sends the aggregated content to the client's web browser for display.

    - Disadvantages:
        a. Increased server load: Server-side processing can lead to higher server resource utilization, potentially impacting scalability and performance.
        b. Limited interactivity: Server-side mashups may offer less interactivity, as the client primarily receives pre-processed data.

    - Advantages:
        a. Data security and control: Sensitive data can be processed and controlled on the server, reducing exposure to potential security risks.

b. Centralized management: Data aggregation and processing logic can be maintained centrally, making it easier to update and manage.

2. Client-Side Mashups:
   - Client-side mashups involve data integration and processing within the user's web browser.
   - The client (browser) retrieves data from multiple sources and combines them, creating an integrated view or interaction for the user.

   - Disadvantage:
     a. Security considerations: Client-side processing may expose data from different sources to potential security risks, such as cross-site scripting (XSS) attacks.
     b. Limited control: Control over data integration and presentation may be dispersed across different client devices, making updates and maintenance less centralized.

   - Advantages:
     a. Reduced server load: Client-side mashups offload data processing from the server, potentially improving server scalability and reducing server-side resource usage.
     b. Enhanced interactivity: Client-side mashups often provide a more interactive and responsive user experience as data processing occurs in real-time within the user's browser.

Screenshots:

1. Stripe:
   Stripe provides a payment gateway which can be used to conduct payments on the website.

Code:

Client Side Code:

```
const confirmDelivery = async (id) => {
    try {
      const dataaa = await axios.post("/getcurrorderinfo", { id }, {
withCredentials: true });


      if (dataaa.status === 200) {
        const { totalprice, products, c_email, v_email } = dataaa.data;
        console.log(v_email);
        console.log(totalprice);
        console.log(products);

        const makePayment = async () => {
          try {
            const data = {
              email: v_email,
              price: totalprice,
            };

            const stripe = await
loadStripe("pk_test_51NoPhzSDorTgDdu4kL6wVjVUHIfN6t9nU5lphavBmFhm8w7OCFx9T
8ffZ1pfCKpTj2HyqLn2XMNKBCAliKCJ38Ke00zP4A69Yi");

            const body = {
              products: data,
            };
            const headers = {
              "Content-Type": "application/json",
            };

            const response = await fetch("/create-checkout-session", {
```

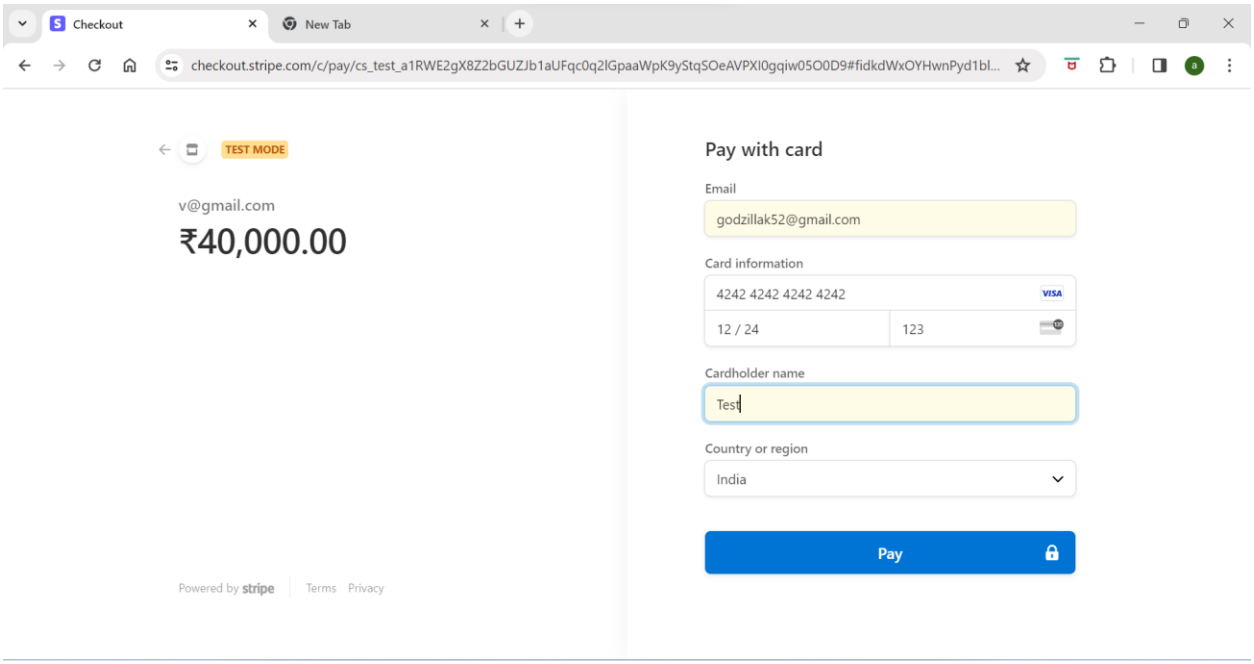
```
            method: "POST",
            headers: headers,
            body: JSON.stringify(body),
          });

          const session = await response.json();

          const result = await stripe.redirectToCheckout({
            sessionId: session.id,
          });
          console.log("Stripe Checkout result:", result);

          if (result.error) {
            console.error(result.error);
          }
        } catch (paymentError) {
          console.error("Payment error:", paymentError);
        }
      };

      // Call the makePayment function if needed
      makePayment();
    }
  } catch (error) {
    if (error.response) {
      toast.error(error.response.data.error);
    } else {
      toast.error("Some error occurred");
    }
  }
```

Server Side Code:

```
require('dotenv').config();
const stripe = require("stripe")(process.env.STRIPE_SECRET_KEY);

router.post("/create-checkout-session", async (req, res) => {
  const { products } = req.body;
  console.log(products);
```

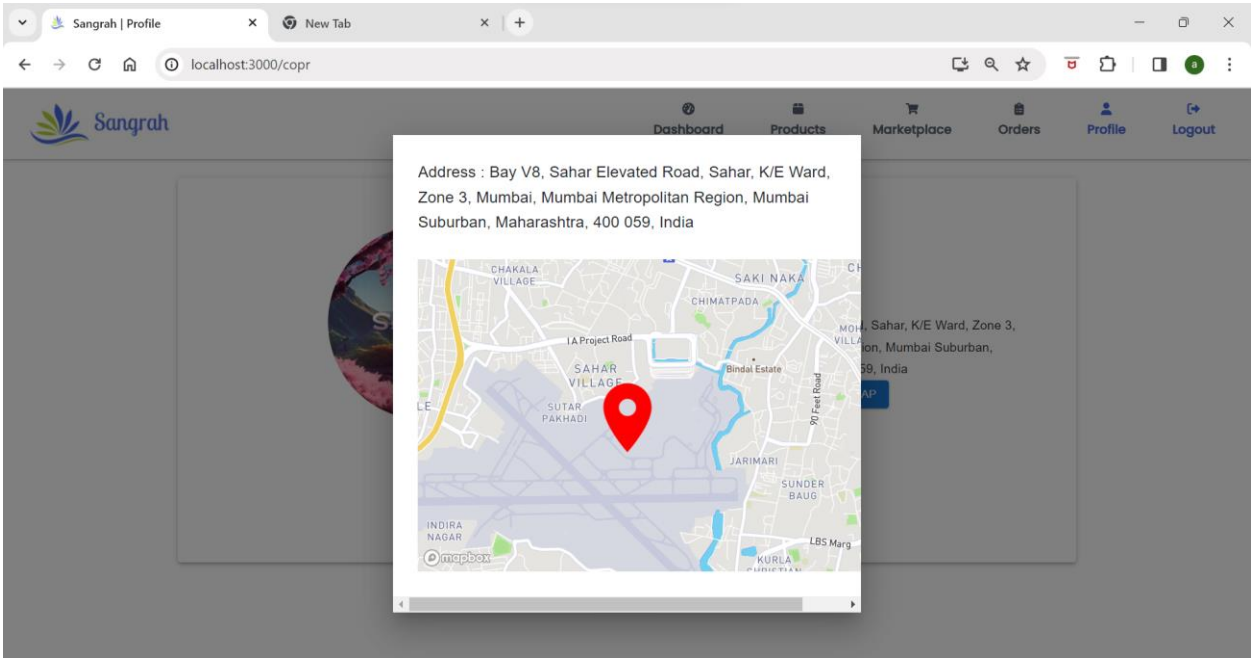
```
  // Create lineItems dynamically based on products
  const lineItem = {
      price_data: {
          currency: "inr",
          product_data: {
              name: products.email, // Assuming each product has a 'name'
property
              images: [], // You can add images if you have them
          },
          unit_amount: products.price * 100, // Assuming each product has
a 'donationAmount' property
      },
      quantity: 1, // You can adjust the quantity as needed
  };
  console.log(lineItem);

  const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
      line_items: [lineItem],
      mode: "payment",
      success_url: "http://localhost:3000/orders",
      cancel_url: "http://localhost:3000/codb",
  });

  res.json({ id: session.id });
});
```

Screenshots:



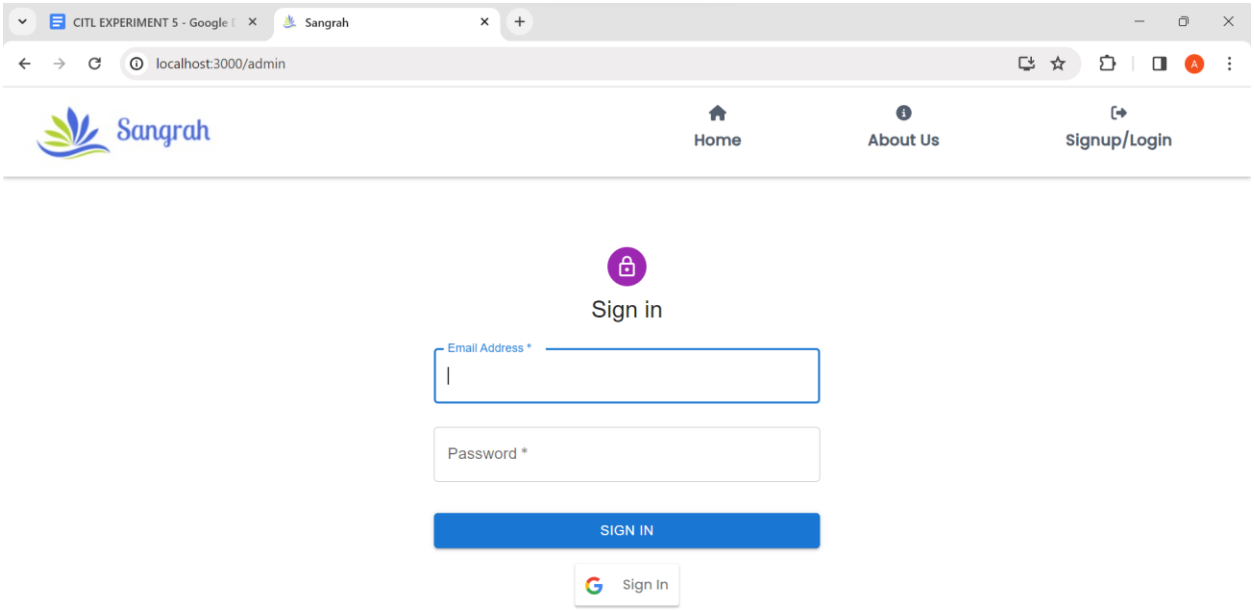
2. MapBox:

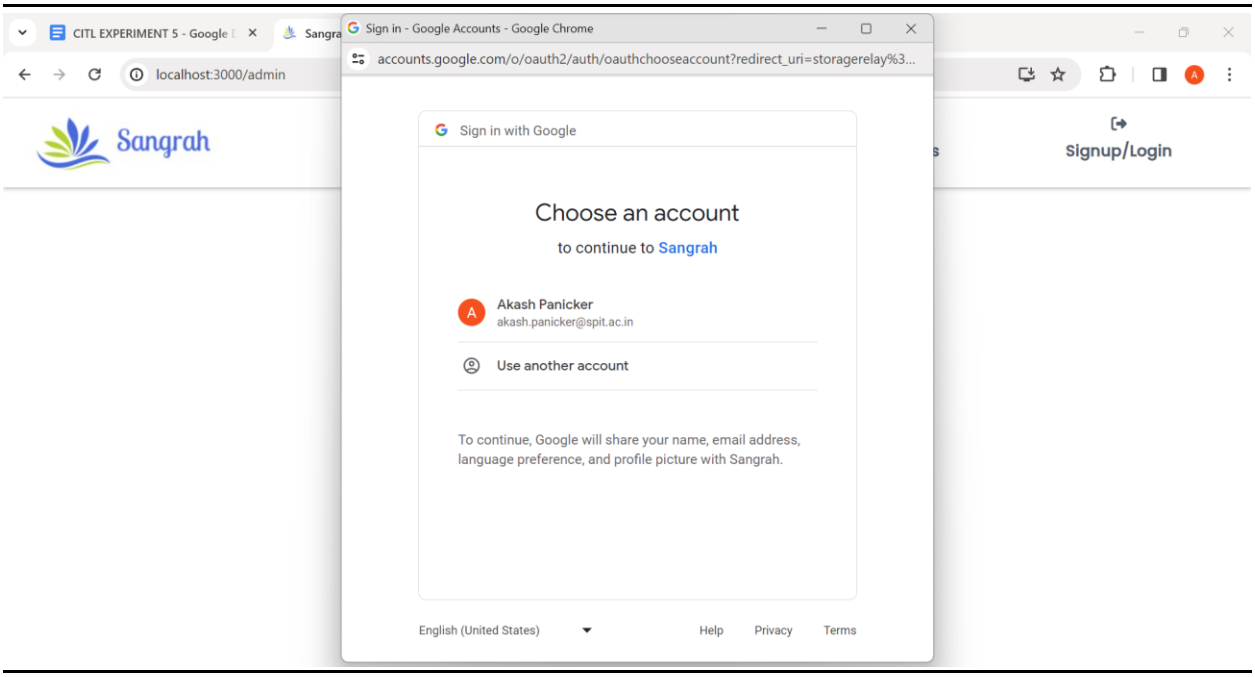
Screenshots:

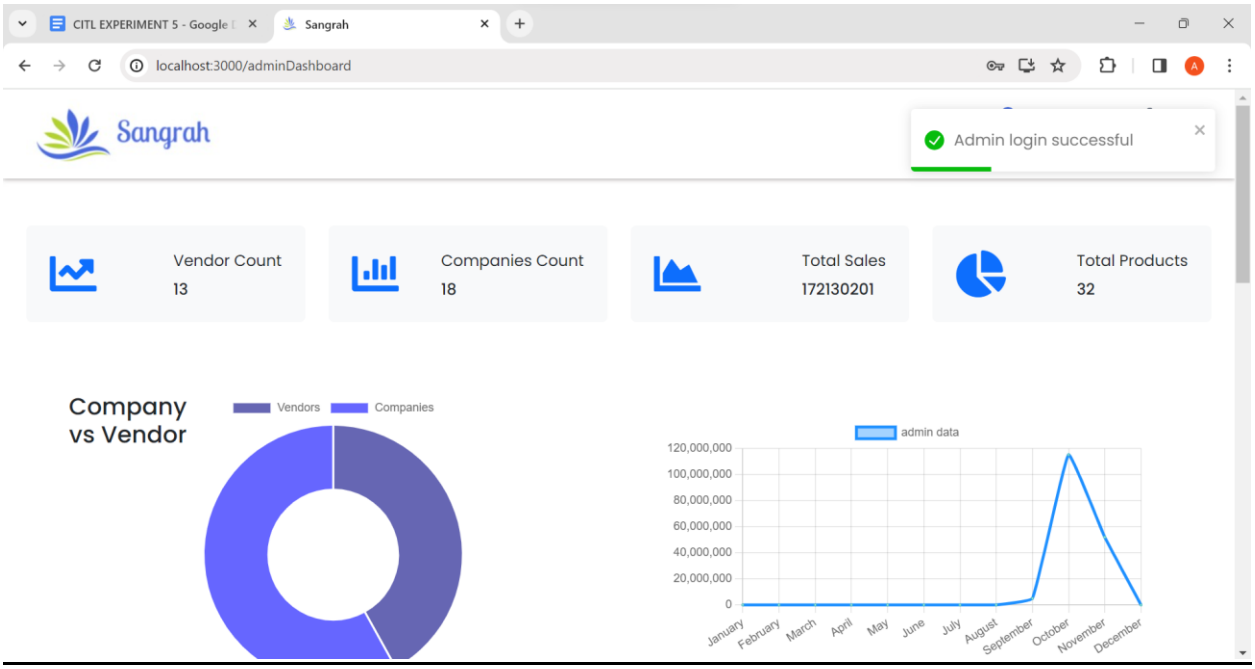


3. Google Auth:

Screenshots:







Conclusion:

By performing this experiment, we understood the concept of web mashups in web development. We understood what It is, its benefits and how to do it in MERN Stack.
By creating this web mashup, we have gained useful incremental experience with both server-side and client-side web development. We learned how to fetch and process data from APIs, how to use an open-source web framework, and how to create dynamic web pages.