**Problem Statement: Development of a Digital Ledger Application (KhataBook)**

**Overview:**

Small businesses often face challenges in maintaining accurate and organized financial records due to the absence of digital solutions tailored to their needs. Traditional ledger systems are cumbersome, prone to errors, and time-consuming to manage. To address these challenges, we aim to develop a user-friendly digital ledger application, named "KhataBook," to streamline financial management for small businesses. The specific objectives of the KhataBook app development project include:

**Simplify Financial Management:** Develop an intuitive and easy-to-use platform that simplifies the recording, tracking, and management of financial transactions for small businesses. By replacing manual ledger systems with a digital solution, the app aims to reduce the time and effort required for accounting tasks while minimizing the risk of errors and discrepancies.

**Facilitate Real-time Transaction Recording:** Enable users to record transactions as they occur, ensuring that financial data is captured accurately and in real-time. By providing instant access to transaction recording features, the app allows users to maintain up-to-date financial records and track their business activities more effectively.

**Automate Calculations and Summaries:** Implement automated calculation features to perform calculations such as totals, balances, and summaries automatically. By automating tedious calculations traditionally done manually, the app helps users save time and ensures accuracy in financial calculations.

**Enhance Transaction Analysis and Reporting:** Provide tools and functionalities for analysing transaction data, generating reports, and visualizing financial performance metrics. By offering insights into business trends, patterns, and financial health, the app empowers users to make informed decisions and identify areas for improvement in their business operations.

**Ensure Data Security and Privacy:** Prioritize the security and privacy of user data by implementing robust security measures such as encryption, authentication, and access controls. By safeguarding sensitive financial information, the app instils trust and confidence among users and mitigates the risk of data breaches or unauthorized access.

**Support Scalability and Adaptability:** Build the app with scalability and adaptability in mind to accommodate the evolving needs and growth of small businesses over time. Ensure that the app can scale seamlessly to handle increased transaction volumes and incorporate new features or integrations as needed to meet changing business requirements.

**Key Features:**

**User Authentication:** Implement secure user authentication mechanisms to ensure that only authorized users can access the app.

**Transaction Recording:** Enable users to record various types of transactions, including sales, expenses, payments, and credits, with relevant details such as date, amount, description, and category.

**Transaction Management:** Provide functionalities for viewing, editing, and deleting transactions, allowing users to maintain accurate and up-to-date financial records.

**Transaction Analysis:** Offer tools for analysing transaction data, including generating reports, charts, and summaries to gain insights into business performance and trends.

**Reminder and Alerts:** Implement notification features to remind users of pending payments, overdue invoices, and other important financial events.

**Data Security:** Ensure the security and privacy of user data by implementing encryption, access controls, and other security measures to prevent unauthorized access or data breaches.

**Accessibility:** Design the app with a user-friendly interface and intuitive navigation to cater to users with varying levels of technological expertise.

**Target Audience:**

The KhataBook app is tailored to meet the needs of Indian shop vendors, encompassing a wide range of businesses including local grocery stores, street vendors, small-scale retailers, and independent merchants operating across various sectors. With a focus on catering to the unique challenges and requirements faced by this demographic, the app aims to serve as an indispensable tool for enhancing financial management, optimizing operational efficiency, and fostering business growth.

Overview of the Target Audience:

Indian shop vendors represent a diverse and vibrant segment of the economy, playing a crucial role in the retail landscape of the country. From bustling marketplaces to quiet neighbourhood stores, these vendors serve as pillars of their communities, providing essential goods and services to customers while contributing to local economies and livelihoods.

Characteristics and Challenges:

Indian shop vendors operate in a dynamic and competitive environment characterized by factors such as fluctuating consumer demand, seasonality, price sensitivity, and regulatory compliance. Despite their integral role in the retail ecosystem, many vendors face significant challenges in managing their finances effectively, primarily due to the lack of access to modern accounting tools and resources.

Role of the KhataBook App:

The KhataBook app is designed to address these challenges and empower Indian shop vendors with a comprehensive and user-friendly solution for managing their finances. By leveraging the capabilities of modern technology and digital innovation, the app aims to revolutionize the way shop vendors track, record, and analyse their financial transactions, thereby enabling them to streamline operations, improve decision-making, and achieve greater financial stability and success.

The app provides a simple and intuitive platform for recording various types of transactions, including sales, purchases, expenses, and payments. With user-friendly interfaces and customizable templates, shop vendors can quickly and accurately capture transaction details, eliminating the need for manual paperwork or tallying.

By digitizing financial data and offering real-time reporting and analysis tools, the app gives shop vendors instant visibility into their business performance, cash flow status, and profitability metrics. Interactive dashboards, charts, and graphs enable vendors to track trends, monitor key performance indicators (KPIs), and identify opportunities for improvement or cost savings.

**Use Case: Managing Shop Records with KhataBook**

**Actors:**

* Shop Owner: The owner of a small shop in India.
* Customers: Individuals who purchase goods from the shop.

**Preconditions:**

* The KhataBook app is installed and configured on the shop owner's smartphone.
* The shop owner has registered an account on the KhataBook platform.

**Main Flow:**

**Recording Transactions:**

The shop owner opens the KhataBook app.

Upon a customer's purchase, the shop owner records the transaction by adding the customer's name, the items purchased, and the amount owed.

Optionally, the shop owner can add additional details such as the date of purchase or any specific notes related to the transaction.

**Viewing Transactions:**

The shop owner can view a list of all recorded transactions within the app. Transactions are organized chronologically, making it easy for the shop owner to track sales over time.

**Updating Transactions:**

If there are any changes to a recorded transaction (e.g., a refund or adjustment), the shop owner can easily update the transaction details within the app. This ensures that the shop owner maintains accurate records of all transactions.

**Managing Customer Accounts:**

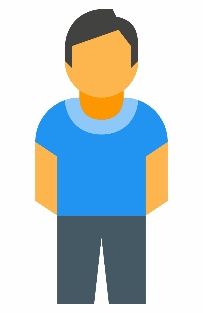
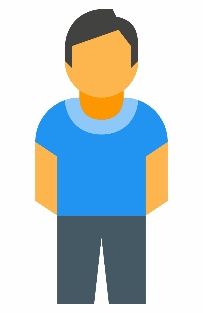
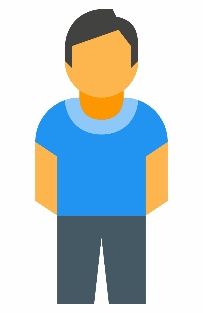
The shop owner can create individual accounts for frequent customers within the app. This allows for better customer relationship management and personalized service.

**Backup and Security:**

KhataBook automatically backs up transaction data to the cloud, ensuring that records are securely stored and accessible from any device. The app employs robust security measures to protect sensitive customer and business data.

**Data Flow Diagrams (DFD)**

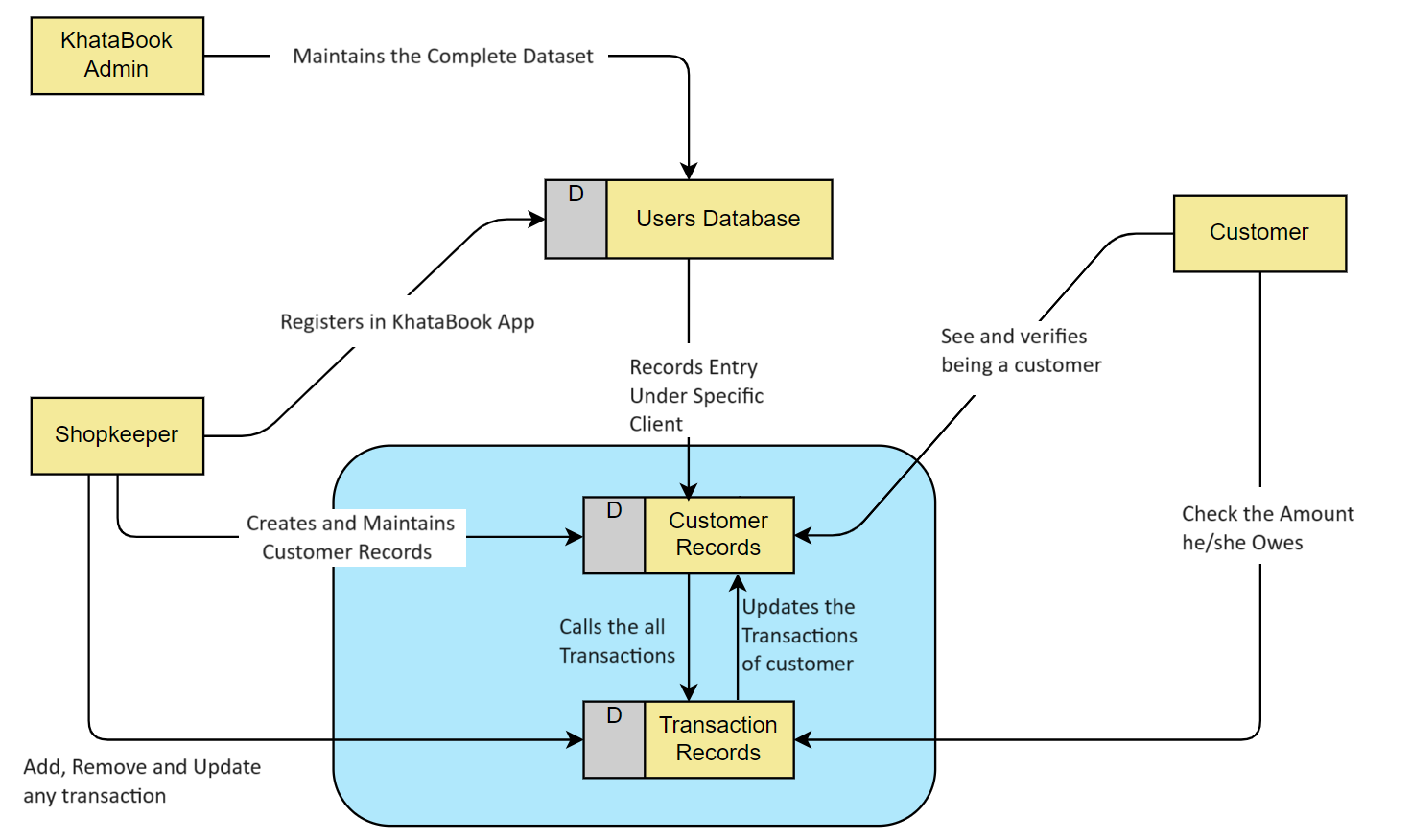
**Users:**



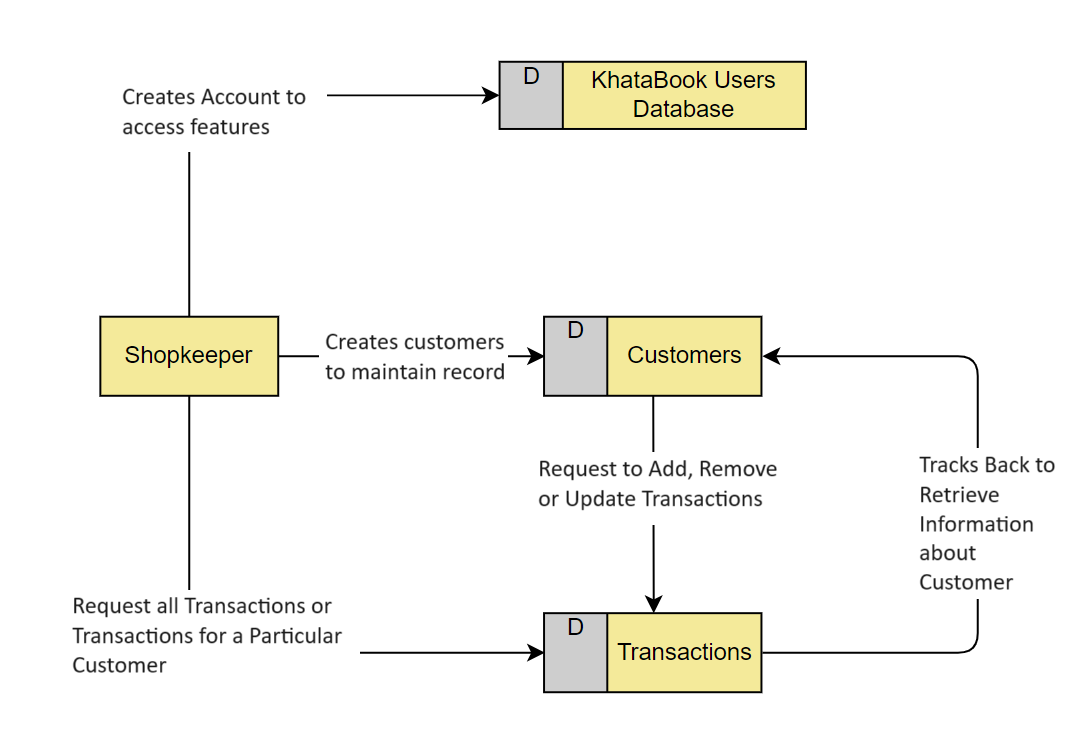
KhataBook Admin Khatabook Users Customers

(Shopkeepers)

**Complete Data Flow Illustration**



**Shopkeeper’s Data flow Diagram:**

****

**Level 0 DFD:**

**Processes:**

Record Management: This process involves the creation, editing, and deletion of journal entries.

**Data Stores:**

Journal Entries: This data store contains all the journal entries created by the user.

**External Entities:**

User: The primary external entity interacting with the KhataBook app.

**Data Flow:**

Create/Edit/Delete Entry: This data flow represents the user's actions of creating, editing, or deleting journal entries.

Display Entries: This data flow represents the retrieval of journal entries from the data store to display them to the user.

**Level 1 DFD:**

**Processes:**

Create Entry: This process handles the creation of new journal entries.

Edit Entry: This process allows the user to edit existing journal entries.

Delete Entry: This process handles the deletion of journal entries.

**Data Stores:**

Journal Entries: This data store contains all the journal entries created by the user.

**External Entities:**

User: The primary external entity interacting with the KhataBook app.

**Data Flows:**

Create Entry: Data flow representing the creation of a new journal entry by the user.

Edit Entry: Data flow representing the editing of an existing journal entry by the user.

Delete Entry: Data flow representing the deletion of a journal entry by the user.

Display Entries: Data flow representing the retrieval of journal entries from the data store to display them to the user.

**Screens (HTML + CSS)**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>KhataBook | SignUp Form</title>

<link rel="stylesheet" type="text/css" href="style.css" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css">

</head>

<body>

<div class="nav">

KhataBook .

</div>

<div class="container" id="container">

<div class="form-container sign-up">

<form action="/sign\_up" method="POST">

<h1>Create Account</h1>

<div class="social-icons">

<a href="#" class="icon"><i class="fa-brands fa-google-plus-g"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-facebook-f"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-github"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-linkedin-in"></i></a>

</div>

<span>or use your email for registeration</span>

<input type="text" class="box" id="name" name="name" placeholder="Name" required />

<input type="email" class="box" id="email" name="email" placeholder="Email" required />

<input type="number" class="box" id="phno" name="phno" placeholder="Mobile" required />

<input type="password" class="box" id="password" name="password" placeholder="Password" required />

<input type="submit" value="Submit" id="submit" />

</form>

</div>

<div class="form-container sign-in">

<form action="/login" method="POST">

<h1>Sign In</h1>

<div class="social-icons">

<a href="#" class="icon"><i class="fa-brands fa-google-plus-g"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-facebook-f"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-github"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-linkedin-in"></i></a>

</div>

<span>or use your email password</span>

<input type="email" id="email" name="email" placeholder="Email" required>

<input type="password" id="password" name="password" placeholder="Password" required>

<a href="#">Forget Your Password?</a>

<button type="submit">Sign In</button>

</form>

</div>

<div class="toggle-container">

<div class="toggle">

<div class="toggle-panel toggle-left">

<h1>Welcome Back!</h1>

<p>Enter your personal details to use all of site features</p>

<button class="hidden" id="login">Sign In</button>

</div>

<div class="toggle-panel toggle-right">

<h1>Hello, Friend!</h1>

<p>Register with your personal details to use all of site features</p>

<button class="hidden" id="register">Sign Up</button>

</div>

</div>

</div>

</div>

<script>

const container = document.getElementById('container');

const registerBtn = document.getElementById('register');

const loginBtn = document.getElementById('login');

registerBtn.addEventListener('click', () => {

container.classList.add("active");

});

loginBtn.addEventListener('click', () => {

container.classList.remove("active");

});

</script>

</body>

</html>

**style.css**

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600;700&display=swap');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Montserrat', sans-serif;

}

body {

background-color: #c9d6ff;

background: linear-gradient(to right, #e2e2e2, #c9d6ff);

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

height: 100vh;

}

.footer {

min-height: 100px;

}

.box {

max-height: fit-content;

width: 80vw;

}

.nav {

width: 98vw;

padding: 30px;

color: #333;

font-weight: bolder;

font-size: 2rem;

}

.container {

background-color: #fff;

border-radius: 30px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.35);

position: relative;

overflow: hidden;

width: 768px;

max-width: 100%;

min-height: 480px;

}

.container p {

font-size: 14px;

line-height: 20px;

letter-spacing: 0.3px;

margin: 20px 0;

}

.container span {

font-size: 12px;

}

.container a {

color: #333;

font-size: 13px;

text-decoration: none;

margin: 15px 0 10px;

}

.container button {

background-color: #512da8;

color: #fff;

font-size: 12px;

padding: 10px 45px;

border: 1px solid transparent;

border-radius: 8px;

font-weight: 600;

letter-spacing: 0.5px;

text-transform: uppercase;

margin-top: 10px;

cursor: pointer;

}

.container input[type=submit] {

max-width: 300px;

background-color: #512da8;

color: #fff;

font-size: 12px;

padding: 10px 45px;

border: 1px solid transparent;

border-radius: 8px;

font-weight: 600;

letter-spacing: 0.5px;

text-transform: uppercase;

margin-top: 10px;

cursor: pointer;

}

.container button.hidden {

background-color: transparent;

border-color: #fff;

}

.container form {

background-color: #fff;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

padding: 0 40px;

height: 100%;

}

.container input {

background-color: #eee;

border: none;

margin: 8px 0;

padding: 10px 15px;

font-size: 13px;

border-radius: 8px;

width: 100%;

outline: none;

}

.form-container {

position: absolute;

top: 0;

height: 100%;

transition: all 0.6s ease-in-out;

}

.sign-in {

left: 0;

width: 50%;

z-index: 2;

}

.container.active .sign-in {

transform: translateX(100%);

}

.sign-up {

left: 0;

width: 50%;

opacity: 0;

z-index: 1;

}

.container.active .sign-up {

transform: translateX(100%);

opacity: 1;

z-index: 5;

animation: move 0.6s;

}

.bgcont {

background-color: #512da8;

}

table {

display: flex;

flex-direction: column;

}

tr {

display: flex;

flex-direction: row;

}

@keyframes move {

0%,

49.99% {

opacity: 0;

z-index: 1;

}

50%,

100% {

opacity: 1;

z-index: 5;

}

}

.social-icons {

margin: 20px 0;

}

.social-icons a {

border: 1px solid #ccc;

border-radius: 20%;

display: inline-flex;

justify-content: center;

align-items: center;

margin: 0 3px;

width: 40px;

height: 40px;

}

.toggle-container {

position: absolute;

top: 0;

left: 50%;

width: 50%;

height: 100%;

overflow: hidden;

transition: all 0.6s ease-in-out;

border-radius: 150px 0 0 100px;

z-index: 1000;

}

.container.active .toggle-container {

transform: translateX(-100%);

border-radius: 0 150px 100px 0;

}

.toggle {

background-color: #512da8;

height: 100%;

background: linear-gradient(to right, #5c6bc0, #512da8);

color: #fff;

position: relative;

left: -100%;

height: 100%;

width: 200%;

transform: translateX(0);

transition: all 0.6s ease-in-out;

}

.container.active .toggle {

transform: translateX(50%);

}

.toggle-panel {

position: absolute;

width: 50%;

height: 100%;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

padding: 0 30px;

text-align: center;

top: 0;

transform: translateX(0);

transition: all 0.6s ease-in-out;

}

.toggle-left {

transform: translateX(-200%);

}

.container.active .toggle-left {

transform: translateX(0);

}

.toggle-right {

right: 0;

transform: translateX(0);

}

.container.active .toggle-right {

transform: translateX(200%);

}

**home.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>KhataBook | Welcome Page</title>

<link rel="stylesheet" href="home.css">

<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@24,400,0,0" />

</head>

<body>

<div class="nav">KhataBook .</div>

<div class="wrapper">

<section class="home">

<div class="container">

<div class="row">

<div class="home-text">

<h1>Welcome to

<span class="magic">

<span class="magic-star">

<svg viewBox="0 0 512 512">

<path

d="M512 255.1c0 11.34-7.406 20.86-18.44 23.64l-171.3 42.78l-42.78 171.1C276.7 504.6 267.2 512 255.9 512s-20.84-7.406-23.62-18.44l-42.66-171.2L18.47 279.6C7.406 276.8 0 267.3 0 255.1c0-11.34 7.406-20.83 18.44-23.61l171.2-42.78l42.78-171.1C235.2 7.406 244.7 0 256 0s20.84 7.406 23.62 18.44l42.78 171.2l171.2 42.78C504.6 235.2 512 244.6 512 255.1z" />

</svg>

</span>

<span class="magic-star">

<svg viewBox="0 0 512 512">

<path

d="M512 255.1c0 11.34-7.406 20.86-18.44 23.64l-171.3 42.78l-42.78 171.1C276.7 504.6 267.2 512 255.9 512s-20.84-7.406-23.62-18.44l-42.66-171.2L18.47 279.6C7.406 276.8 0 267.3 0 255.1c0-11.34 7.406-20.83 18.44-23.61l171.2-42.78l42.78-171.1C235.2 7.406 244.7 0 256 0s20.84 7.406 23.62 18.44l42.78 171.2l171.2 42.78C504.6 235.2 512 244.6 512 255.1z" />

</svg>

</span>

<span class="magic-star">

<svg viewBox="0 0 512 512">

<path

d="M512 255.1c0 11.34-7.406 20.86-18.44 23.64l-171.3 42.78l-42.78 171.1C276.7 504.6 267.2 512 255.9 512s-20.84-7.406-23.62-18.44l-42.66-171.2L18.47 279.6C7.406 276.8 0 267.3 0 255.1c0-11.34 7.406-20.83 18.44-23.61l171.2-42.78l42.78-171.1C235.2 7.406 244.7 0 256 0s20.84 7.406 23.62 18.44l42.78 171.2l171.2 42.78C504.6 235.2 512 244.6 512 255.1z" />

</svg>

</span>

<span class="magic-text">KhataBook </span>

</span>

</h1>

<p class="animate-text">

<span>Simplifying your digital experience!</span>

<span>Managinging your journal entries!</span>

<span>Simplifying your expense tracking!</span>

<span>Automating your cyber experience!</span>

</p>

</div>

</div>

</div>

</section>

<p>KhataBook is a user-friendly platform designed to streamline your online interactions. With seamless and

error free finance management, KhataBook ensures a smooth and secure experience for all users.</p>

<div class="bottom">

<div class="errorBox" id="errorBox">

<span id="errorSpan">User Not found</span>

</div>

<a href="./createCustomer.html">

<button type="submit">Create a New Customer</button> <br>

</a>

or <br><br>

<form action="/searchCustomer" method="POST">

<div class="search">

<span class="material-symbols-outlined">search</span>

<input type="search" class="searchInput" name="userName" id="userName"

placeholder="Search Existing User" required>

</div>

<button type="submit" onclick="save()">Search</button>

</form>

<a href="./completeLogs.html">

<button type="submit">View All Transactions</button> <br>

</a>

</div>

</div>

<div class="footer"></div>

<script>

let indexabc = 0,

interval = 1000;

const rand = (min, max) =>

Math.floor(Math.random() \* (max - min + 1)) + min;

const animate = star => {

star.style.setProperty("--star-left", `${rand(-10, 100)}%`);

star.style.setProperty("--star-top", `${rand(-40, 80)}%`);

star.style.animation = "none";

star.offsetHeight;

star.style.animation = "";}

for (const star of document.getElementsByClassName("magic-star")) {

setTimeout(() => {

animate(star);

setInterval(() => animate(star), 1000);

}, indexabc++ \* (interval / 3))

}

</script>

<script>

function save() {

const name = document.getElementById('userName').value;

localStorage.setItem("khatabookUser", name);

}

const txts = document.querySelector(".animate-text").children,

txtsLen = txts.length;

let index = 0;

const textInTimer = 3000,

textOutTimer = 2800;

function animateText() {

for (let i = 0; i < txtsLen; i++) {txts[i].classList.remove("text-in", "text-out");}

txts[index].classList.add("text-in");

setTimeout(function () {

txts[index].classList.add("text-out");

}, textOutTimer)

setTimeout(function () {

if (index == txtsLen - 1) {

index = 0;

}else {

index++;

}

animateText();

}, textInTimer);

}

window.onload = animateText;

</script>

</body>

</html>

**home.css**

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600;700&display=swap');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Montserrat', sans-serif;

}

:root {

--purple: rgb(123, 31, 162);

--violet: rgb(103, 58, 183);

--pink: rgb(244, 143, 177);

}

@keyframes background-pan {

from {

background-position: 0% center;

} to {

background-position: -200% center;

}

}

@keyframes scale {

from, to {

transform: scale(0);

}

50% {

transform: scale(1);

}

}

@keyframes rotate {

from {

transform: rotate(0deg);

} to {

transform: rotate(180deg);

}

}

h1>.magic {

display: inline-block;

position: relative;

}

h1>.magic>.magic-star {

--size: clamp(20px, 1.5vw, 30px);

animation: scale 700ms ease forwards;

display: block;

height: var(--size);

left: var(--star-left);

position: absolute;

top: var(--star-top);

width: var(--size);

}

h1>.magic>.magic-star>svg {

animation: rotate 1000ms linear infinite;

display: block;

opacity: 0.7;

}

h1>.magic>.magic-star>svg>path { fill: var(--violet);}

h1>.magic>.magic-text {

animation: background-pan 3s linear infinite;

background: linear-gradient(to right,

var(--purple),

var(--violet),

var(--pink),

var(--purple));

background-size: 200%;

background-clip: text;

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

white-space: nowrap;

}

body {

background-color: #c9d6ff;

background: linear-gradient(to right, #e2e2e2, #c9d6ff);

}

.nav {

width: 98vw;

padding: 30px;

color: #333;

font-weight: bolder;

font-size: 2rem;

}

.wrapper {

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

margin-top: 30;

}

h2 {

font-weight: lighter;

margin-top: 20px;

overflow: hidden;

}

.footer {

height: 100px;

}

p {

max-width: 65%;

margin-top: 40px;

}

.home {

display: flex;

flex-wrap: wrap;

padding: 50px 15px;

padding-bottom: 0;

position: relative;

align-items: center;

justify-content: center;

}

.home::before {

content: '';

position: absolute;

left: 0;

top: 0;

width: 100%;

height: 100%;

z-index: 1;

}

.home .home-text {

text-align: center;

position: relative;

z-index: 2;

}

.home .home-text h1 {

color: #333;

font-size: 50px;

line-height: 56px;

font-weight: 600;

}

.home .home-text p {

margin: 0;

overflow: hidden;

}

.home .home-text p span {

font-size: 20px;

color: #4c2cff;

font-weight: 700;

display: inline-block;

line-height: 46px;

display: none;

}

.home .home-text p span.text-in {

display: block;

animation: textIn .5s ease;

}

.home .home-text p span.text-out {

animation: textOut .5s ease;

}

.bottom {

margin-top: 50px;

text-align: center;

}

.errorBox {

padding: 10px;

margin: 3px;

color: rgb(255, 70, 70);

display: none;

}

button {

margin: 10px;

padding: 10px 15px;

width: 400px;

background-color: #4c2cff;

color: white;

border-radius: 5px;

border: none;

font-size: large;

}

button:hover {

cursor: pointer;

}

button:active {

background-color: #2100dc;

}

.search {

width: max-content;

display: flex;

align-items: center;

padding: 14px;

border-radius: 28px;

background: #f6f6f6;

}

.searchInput {

width: 352px;

font-size: 16px;

color: #040404;

margin-left: 14px;

outline: none;

border: none;

background: transparent;

flex: 1;

}

.searchInput::placeholder {

color: rgba(0, 0, 0, 0.75);

}

@keyframes textIn {

0% {

transform: translateY(100%);

} 100% {

transform: translateY(0%);

}

}

@keyframes textOut {

0% {

transform: translateY(0%);

} 100% {

transform: translateY(-100%);

}

}

**createCustomer.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>KhataBook | Create User</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="nav">

KhataBook .

</div>

<div class="container" id="container">

<form action="/createCustomer" method="POST">

<h1>Create New Customer</h1>

<p>Enter the details of new customer</p>

<input type="text" class="box" id="name" name="name" placeholder="Name" required />

<input type="number" class="box" id="phno" name="phno" placeholder="Mobile" required />

<input type="submit" value="Submit" id="submit" />

</form>

</div>

</body>

</html>

**details.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>KhataBook | Expense Tracker</title>

<link rel="stylesheet" href="./details.css" />

</head>

<body>

<div class="nav"> KhataBook . </div>

<h1 id="userName">User Name</h1>

<main>

<header>

<div>

<h5>Total Balance</h5>

<span id="balance">₹0.00</span>

</div>

<div>

<h5>Money Out</h5>

<span id="income">₹0.00</span>

</div>

<div>

<h5>Money In</h5>

<span id="expense">₹0.00</span>

</div>

</header>

<section>

<h3>Transactions</h3>

<ul id="transactionList"></ul>

<div id="status"></div>

</section>

<section>

<h3>Add Transaction</h3>

<form id="transactionForm">

<div>

<label for="type">

<input type="checkbox" name="type" id="type" />

<div class="option">

<span>Money Out</span>

<span>Money In</span>

</div>

</label>

</div>

<div>

<label for="name">Note</label>

<input type="text" name="name" required />

</div>

<div>

<label for="amount">Amount</label>

<input type="number" name="amount" value="0" min="0.01" step="0.01" required />

</div>

<div>

<label for="date">Date</label>

<input type="date" name="date" required />

</div>

<button type="submit">Submit</button>

</form>

</section>

</main>

<script>

allCookies = document.cookie;

let name = "khatabookname=";

let decodedCookie = decodeURIComponent(document.cookie);

let ca = decodedCookie.split(';');

let s = "";

for (let i = 0; i < ca.length; i++) {

let c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

s = c.substring(name.length, c.length);

}

}

document.getElementById('userName').innerHTML = `${s}`;

const list = document.getElementById("transactionList");

const form = document.getElementById("transactionForm");

const status = document.getElementById("status");

const balance = document.getElementById("balance");

const income = document.getElementById("income");

const expense = document.getElementById("expense");

form.addEventListener("submit", addData);

const formatter = new Intl.NumberFormat("en-US", {

style: "currency",

currency: "INR",

signDisplay: "always",

});

function fetchData() {

let inc = 0, out = 0, tot = 0;

balance.innerHTML = `₹0`;

income.innerHTML = `₹0`;

expense.innerHTML = `₹0`;

fetch('/api/data')

.then(response => response.json())

.then(data => {

console.log('Success:', data);

list.innerHTML = "";

status.textContent = "";

if (data.length === 0) {

status.textContent = "No transactions.";

return;

}

data.forEach(({ \_id, name, note, amount, date, type }) => {

if (name == s) {

const sign = "Money Out" === type ? 1 : -1;

if (sign == 1) {

inc += amount;

} else {

out += amount;

}

const li = document.createElement("li");

li.innerHTML = `

<div class="name">

<h4>${note}</h4>

<p>${new Date(date).toLocaleDateString()}</p>

</div>

<div class="amount ${type}">

<span>${formatter.format(amount \* sign)}</span>

</div>

<div class="action">

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" onclick="deleteTransaction('${\_id}')">

<path stroke-linecap="round" stroke-linejoin="round" d="M9.75 9.75l4.5 4.5m0-4.5l-4.5 4.5M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />

</svg>

</div>

`;

list.appendChild(li);

}

let bal = inc - out;

balance.innerHTML = `₹${bal}`;

income.innerHTML = `₹${inc}`;

expense.innerHTML = `₹${out}`;

});

})

.catch((error) => {

console.error('Error:', error);

});

}

function addData(e) {

e.preventDefault();

const formData = new FormData(this);

const data = {

name: s,

note: formData.get("name"),

amount: parseFloat(formData.get("amount")),

date: new Date(formData.get("date")),

type: "on" === formData.get("type") ? "Money In" : "Money Out",

};

this.reset();

fetch('/api/add', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(data),

})

.then(response => response.json())

.then(data => {

console.log('Success:', data);

fetchData();

})

.catch((error) => {

console.error('Error:', error);

});

}

function deleteTransaction(id) {

fetch(`/api/delete/${id}`, {

method: 'DELETE',

}).then(response => {

if (!response.ok) {

throw new Error('Network response was not ok');

} return response.json();

})

.then(data => {

fetchData();

console.log('Success:', data);

})

.catch(error => { console.error('Error:', error); });

}

window.onload = fetchData();

</script>

</body>

</html>

**details.css**

@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap");

:root {

--bg-color: #f6f8fa;

--text-color: #1d1f27;

--main-color: #3a3dbe;

}

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

body {

font-family: "Poppins", sans-serif;

background-color: #c9d6ff;

background: linear-gradient(to right, #e2e2e2, #c9d6ff);

color: var(--text-color);

}

.nav {

width: 98vw;

padding: 30px;

color: #333;

font-weight: bolder;

font-size: 2rem;

}

h1 {

text-align: center;

font-size: 1.5rem;

}

h3,header,ul {

margin-bottom: 0.5rem;

}

main {

max-width: 500px;

margin: 1rem auto;

background-color: #fff;

padding: 1rem;

box-shadow: 0 3px 5px rgba(0, 0, 0, 0.1);

}

header {

background-color: var(--main-color);

color: #fff;

padding: 1rem;

text-align: center;

border-radius: 5px;

display: flex;

flex-wrap: wrap;

}

header div {

padding: 5px;

}

header div:first-child {

flex-basis: 100%;

font-size: 2.25rem;

font-weight: 600;

}

header div:nth-child(n + 2) {

flex-basis: 50%;

font-size: 1.25rem;

font-weight: 500; }

header h5 {

font-size: 0.75rem;

font-weight: 600;

text-transform: uppercase;

}

form {

display: flex;

flex-wrap: wrap;

gap: 10px;

}

form input:not(#type),

form button {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

margin-bottom: 5px;

height: 42px;

font-family: "Poppins", sans-serif;

font-size: 1rem;

}

form button {

background-color: var(--main-color);

color: #fff;

font-size: 0.75rem;

font-weight: 600;

text-transform: uppercase;

box-shadow: 0 3px 5px rgba(0, 0, 0, 0.1);

}

form label {

font-size: 0.75rem;

font-weight: 600;

text-transform: uppercase;

}

form div:nth-child(-n + 2) {

flex-basis: 100%;

}

form div:nth-child(n + 3) {

flex-basis: calc(50% - 5px);

}

input#type {

appearance: none;

position: absolute;

}

.option {

display: flex;

align-items: center;

justify-content: center;

padding: 10px;

background: #eee;

border-radius: 5px;

position: relative;

}

.option span {

width: 50%;

text-align: center;

cursor: pointer;

z-index: 2;

}

.option::before {

content: "";

position: absolute;

top: 5px;

left: 0;

background-color: #fff;

height: calc(100% - 10px);

width: calc(50% - 10px);

transform: translateX(5px);

border-radius: inherit;

box-shadow: 0 2px 2px rgba(0, 0, 0, 0.1);

transition: all 200ms;

}

input#type:checked~.option::before {

left: 50%;

}

ul {

list-style-type: none;

}

ul li {

display: flex;

justify-content: space-between;

align-items: flex-start;

padding: 5px 10px;

position: relative;

}

ul li:hover {

background: rgba(0, 0, 0, 0.1);

}

.name {

flex: 1;

}

.name h4 {

font-size: 1rem;

font-weight: 600;

text-transform: capitalize;

}

.name p {

font-size: 0.8rem;

color: #555;

}

.amount {

font-weight: 600;

}

.amount.income {

color: yellowgreen;

}

.amount.expense {

color: indianred;

}

.action {

position: absolute;

top: 0;

right: 0;

background-color: #f00;

color: #fff;

height: 100%;

width: 50px;

display: grid;

place-items: center;

transform: scaleX(0);

transform-origin: right;

transition: all 300ms;

}

ul li:hover .action {

transform: scaleX(1);

}

.action svg {

width: 36px;

height: 36px;

cursor: pointer;

}

#status {

text-align: center;

margin-bottom: 0.5rem;

}

**completeLogs.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>KhataBook | Complete Logs</title>

<link rel="stylesheet" href="logs.css">

</head>

<body>

<div class="nav">

KhataBook .

</div>

<div class="box">

<h1>All Transactions</h1>

<div class="table">

<section>

<table> <tr>

<th>Name</th>

<th>Product</th>

<th>Date</th>

<th>Amount</th>

</tr> </table>

<hr>

<table id="transactionList"></table>

<div id="status"></div>

</section>

</div>

</div>

<div class="short">

<div class="inc">

Money To Collect: ₹<span id="outMon"></span>

</div>

<div class="exp">

Total Money Collected: ₹<span id="inMon"></span>

</div>

</div>

<div class="footer"></div>

<script>

const list = document.getElementById("transactionList");

const outMon = document.getElementById("outMon");

const inMon = document.getElementById("inMon");

const formatter = new Intl.NumberFormat("en-US", {

style: "currency",

currency: "INR",

signDisplay: "always",

});

let out = 0;

let inc = 0;

fetch('/api/data')

.then(response => response.json())

.then(data => {

console.log('Success:', data);

list.innerHTML = "";

status.textContent = "";

if (data.length === 0) {

status.textContent = "No transactions.";

return; }

data.forEach(({ \_id, name, note, amount, date, type }) => {

const sign = "Money Out" === type ? 1 : -1;

if (sign == 1) {

out += amount;

} else {

inc += amount;

}

const li = document.createElement("tr");

li.innerHTML = `

<td class="name">

<h4>${name}</h4>

</td>

<td class="note">

<h4>${note}</h4>

</td>

<td class="date">

<p>${new Date(date).toLocaleDateString()}</p>

</td>

<td class="amount ${type}">

<span>${formatter.format(amount \* sign)}</span>

</td>

`;

list.appendChild(li);

});

outMon.innerHTML = `${out}`;

inMon.innerHTML = `${inc}`;

})

.catch((error) => {

console.error('Error:', error);

});

</script>

</body>

</html>

**logs.css**

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600;700&display=swap');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Montserrat', sans-serif;

} body {

background-color: #c9d6ff;

background: linear-gradient(to right, #e2e2e2, #c9d6ff);

height: 100vh;

}

.footer {

min-height: 100px;

}

.box {

max-height: fit-content;

width: 98vw;

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between;

}

.nav {

width: 98vw;

padding: 30px;

color: #333;

font-weight: bolder;

font-size: 2rem;

}

table {

width: 80vw;

gap: inherit;

font-size: large;

font-weight: 400;

}

table tr {

margin: 15px 0px;

}

td {

padding-top: 15px;

}

section {

background-color: rgba(93, 81, 255, 0.23);

padding: 20px;

border-radius: 5px;

border: 3px solid white;

}

hr {

width: 80vw;

border: 1px solid white;

border-radius: 3px;

} .short {

display: flex;

flex-direction: column;

width: 98vw;

padding-right: 150px;

align-items: flex-end;

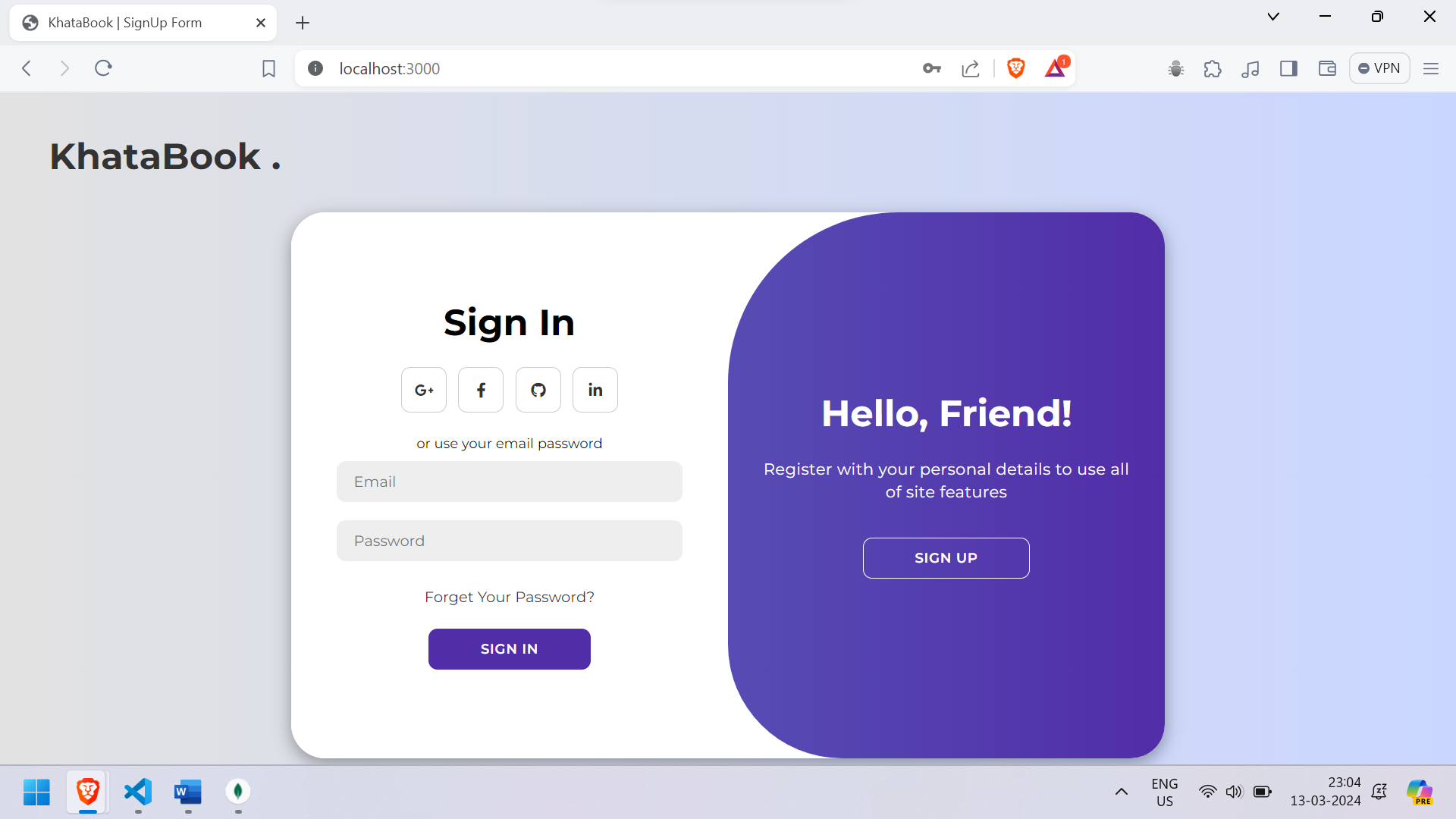
justify-content: center;

font-size: 1.5rem;

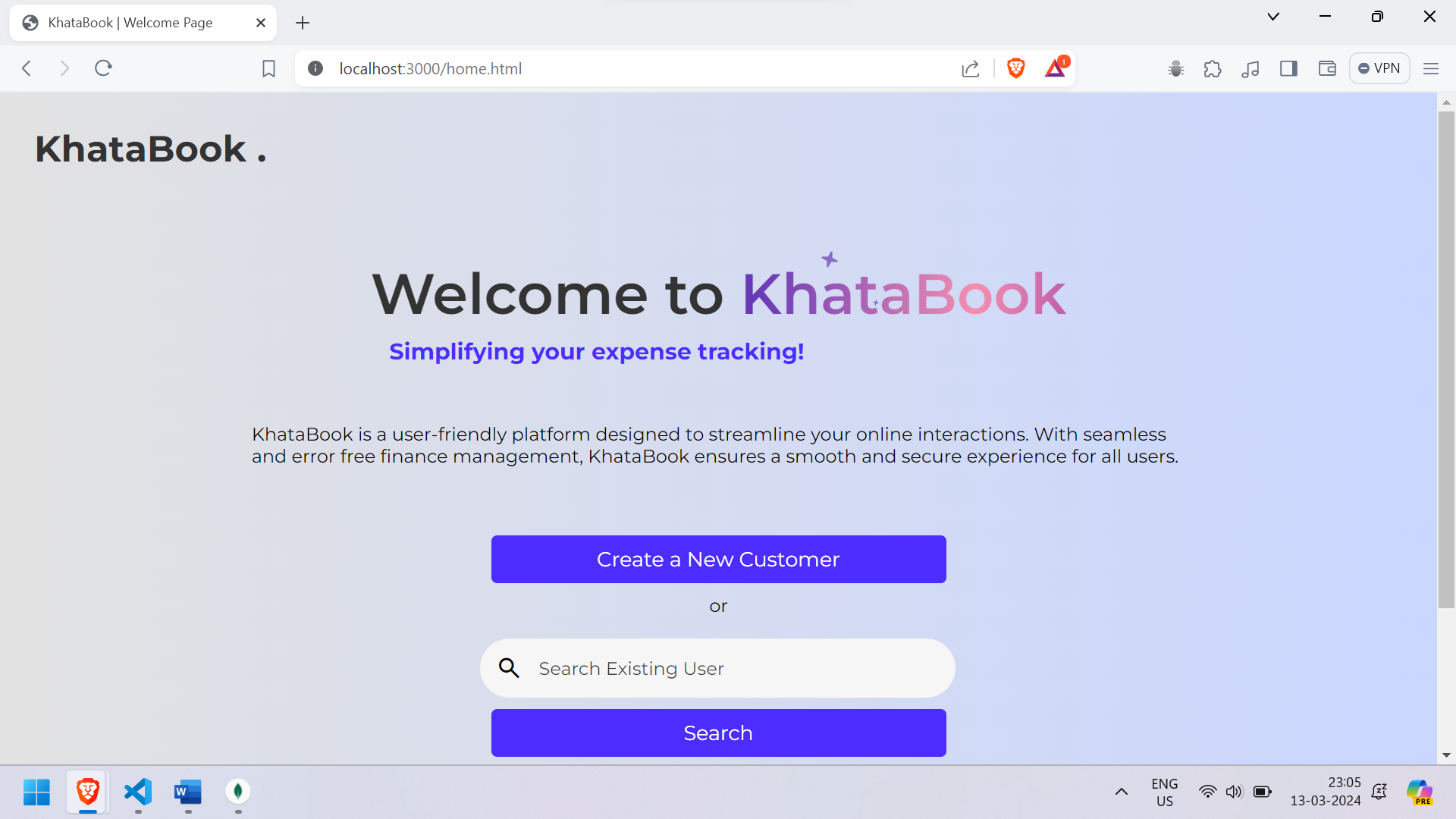
}

**Screens (Outputs)**

**Login-Registration Page:**

****

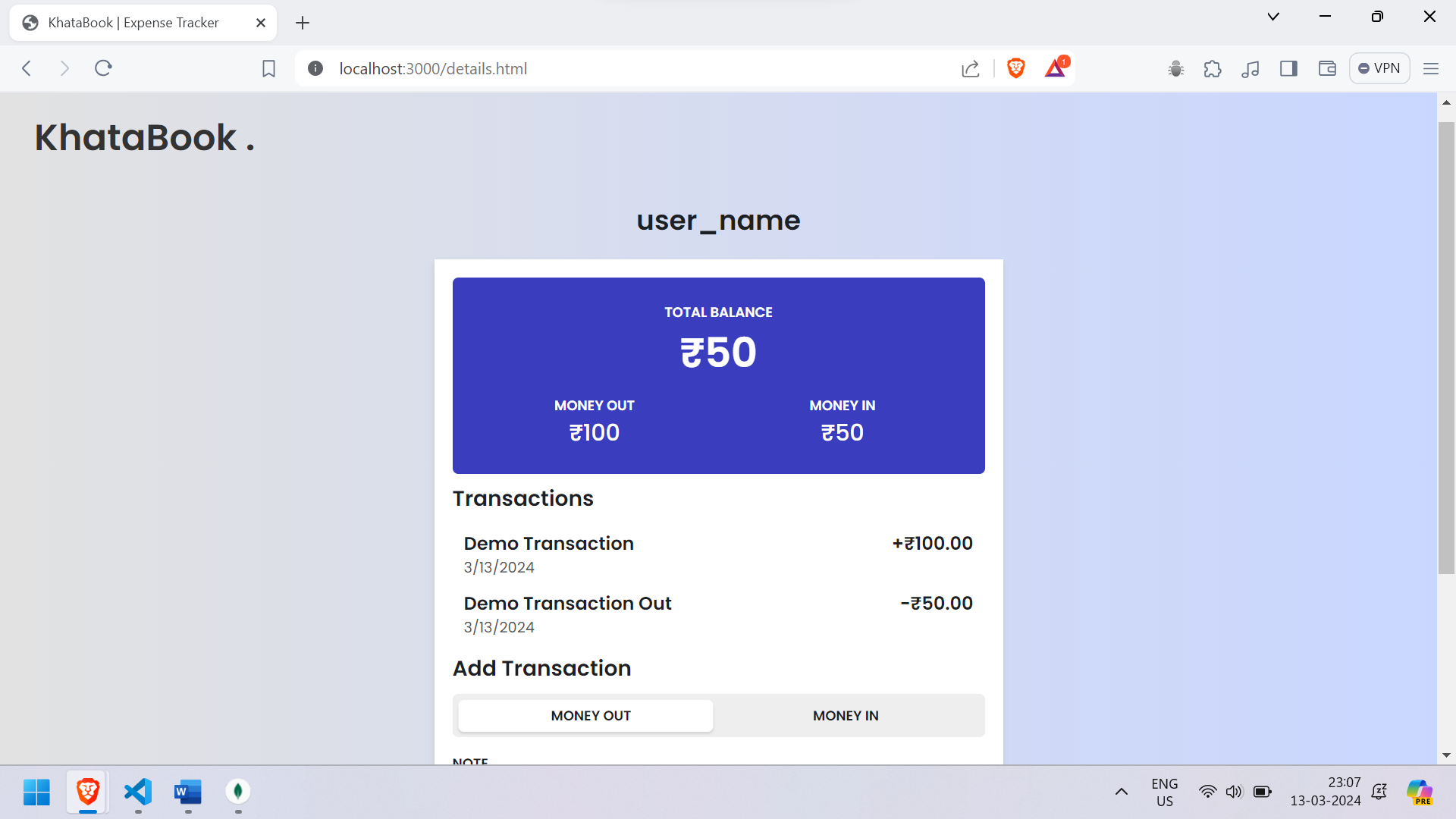
**Home Page:**

****

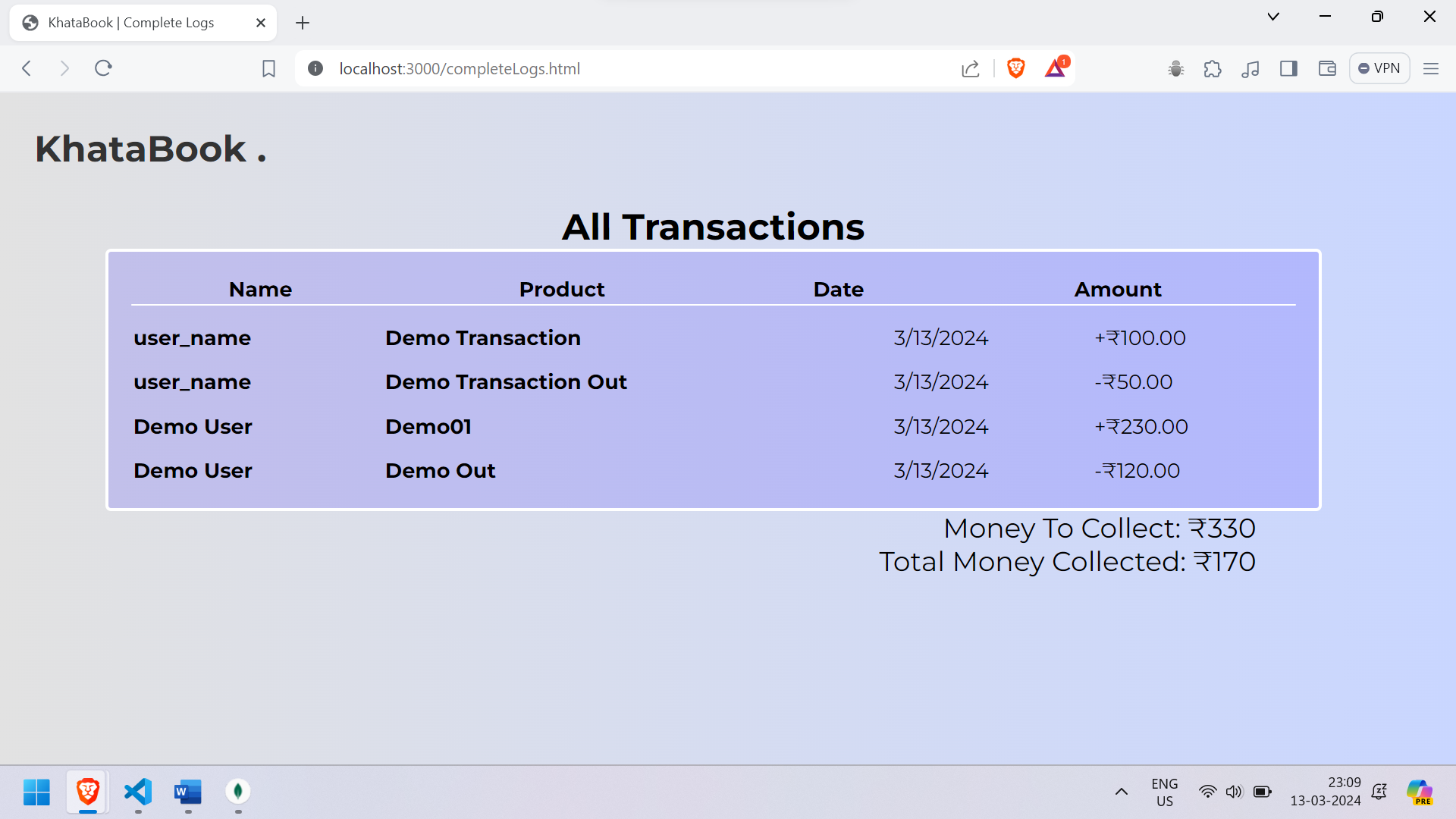
**Create Customer:**

****

**Customer Details:**

****

**Complete Logs**

****

**Database**

**MongoDB**

MongoDB is an open-source NoSQL database management program. NoSQL (Not only SQL) is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

MongoDB is used for high-volume data storage, helping organizations store large amounts of data while still performing rapidly. Organizations also use MongoDB for its ad-hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution and other features.

Structured Query Language (SQL) is a standardized programming language that is used to manage relational databases. SQL normalizes data as schemas and tables, and every table has a fixed structure.

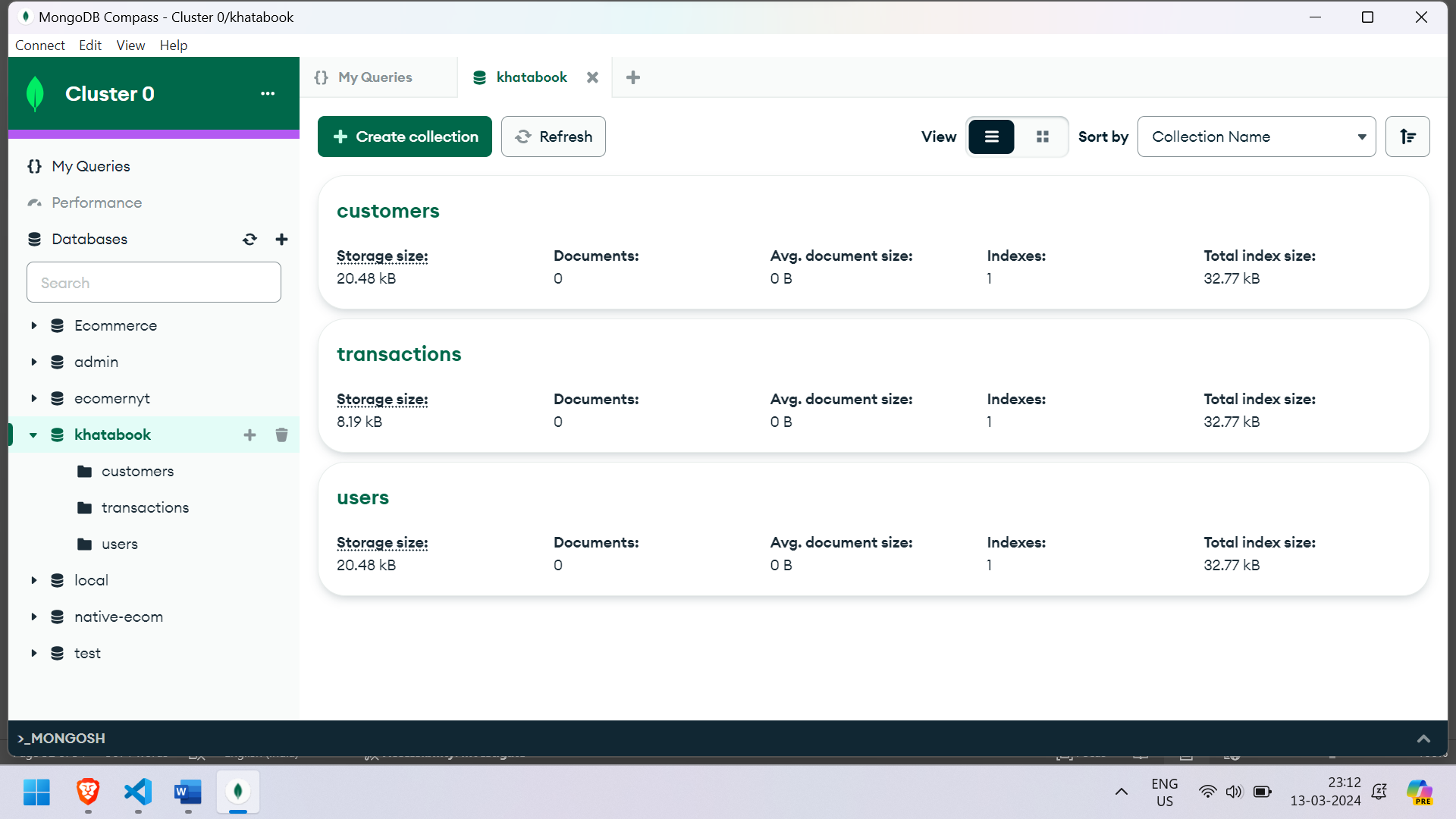
Instead of using tables and rows as in relational databases, as a NoSQL database, the MongoDB architecture is made up of collections and documents. Documents are made up of Key-value pairs -- MongoDB's basic unit of data. Collections, the equivalent of SQL tables, contain document sets. MongoDB offers support for many programming languages, such as C, C++, C#, Go, Java, Python, Ruby and Swift.

**How does MongoDB work?**

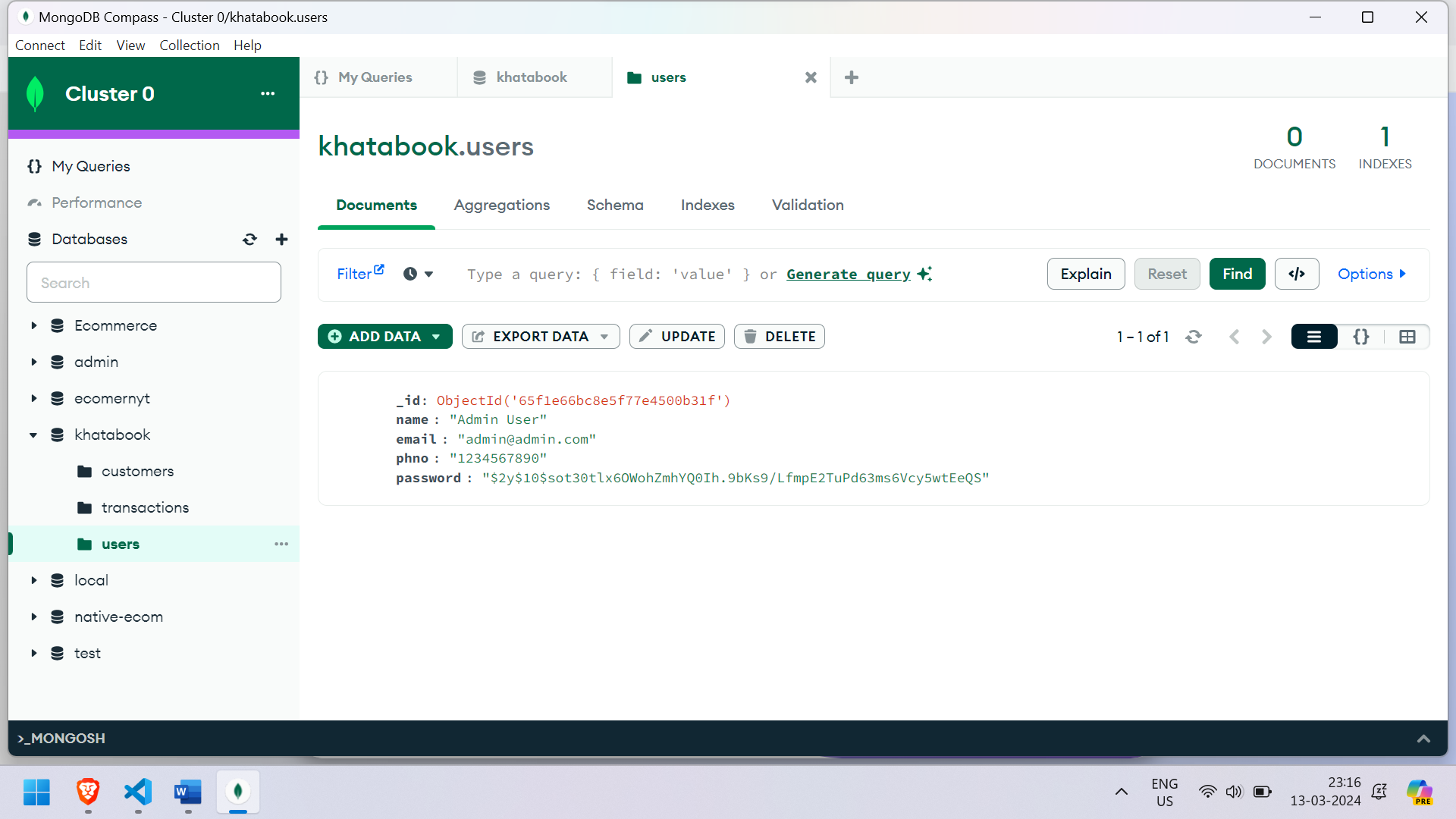
MongoDB environments provide users with a server to create databases with MongoDB. MongoDB stores data as records that are made up of collections and documents.

Documents contain the data the user wants to store in the MongoDB database. Documents are composed of field and value pairs. They are the basic unit of data in MongoDB. The documents are similar to JavaScript Object Notation (JSON) but use a variant called Binary JSON (BSON). The benefit of using BSON is that it accommodates more data types. The fields in these documents are like the columns in a relational database. Values contained can be a variety of data types, including other documents, arrays and arrays of documents, according to the MongoDB user manual. Documents will also incorporate a primary key as a unique identifier. A document's structure is changed by adding or deleting new or existing fields.

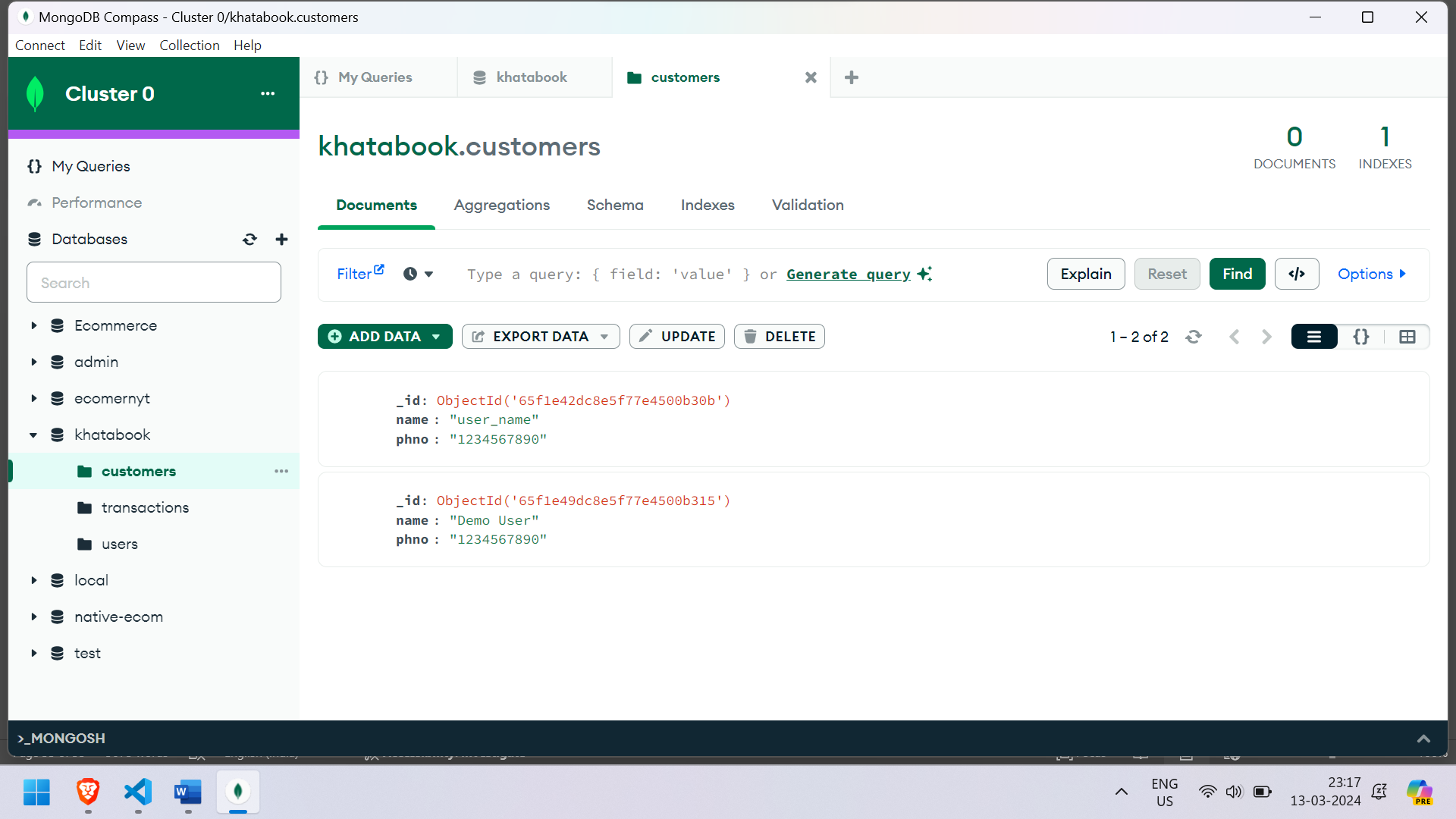
**KhataBook Database:**

****

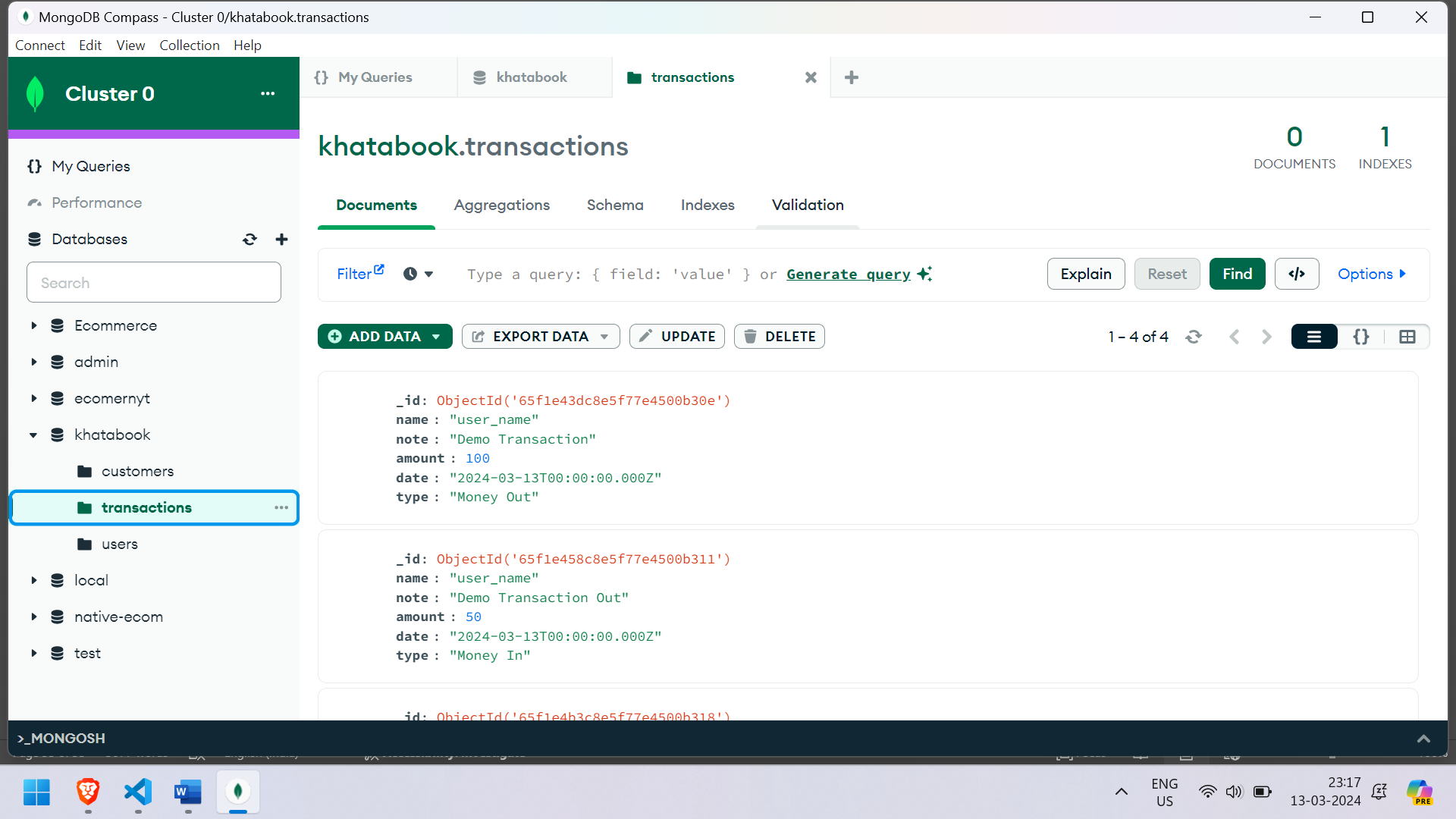
**Users database:**

****

**Customers Database:**

****

**Transactions Database:**

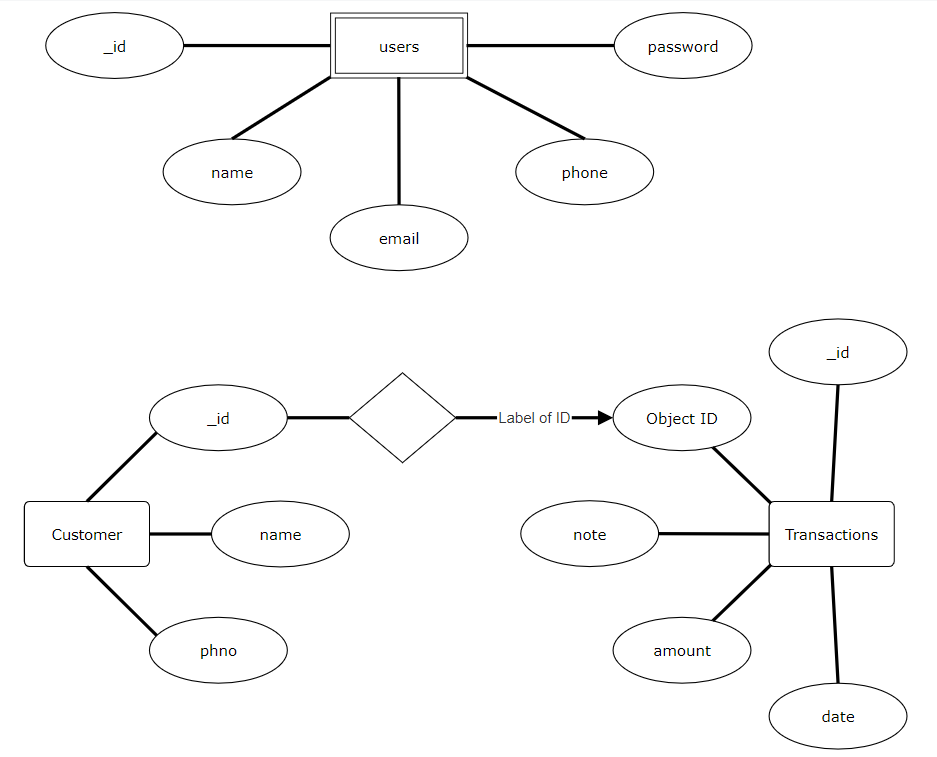
****

**Entity Relationship Diagram**

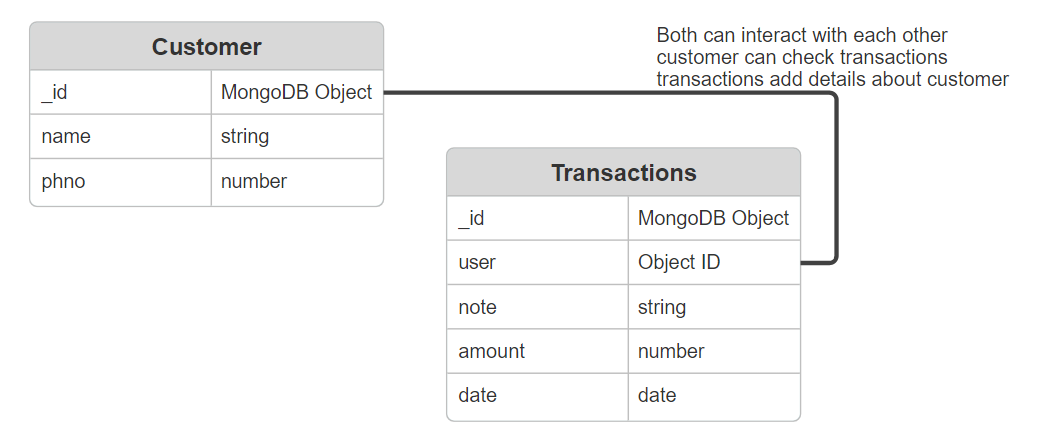
The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

**Complete Relationship Model:**

****

**Customer – Transaction Relationship:**

****

**Coding (JavaScript)**

**index.js**

const express = require("express");

const bodyParser = require("body-parser");

const cookieParser = require("cookie-parser")

const mongoose = require("mongoose");

const app = express();

app.use(bodyParser.json());

app.use(cookieParser());

app.use(express.static("public"));

app.use(bodyParser.urlencoded({extended: true,}));

mongoose.connect("mongodb+srv://kartikginwal07:uF8TBPDR6xvzCjls@cluster0.bk1on2o.mongodb.net/khatabook");

var db = mongoose.connection;

db.on("error", () => console.log("Error in Connecting to Database"));

db.once("open", () => console.log("Connected to Database"));

app.post("/sign\_up", (req, res) => {

var name = req.body.name;

var email = req.body.email;

var phno = req.body.phno;

var password = req.body.password;

var data = {

name: name,

email: email,

phno: phno,

password: password,

};

db.collection("users").insertOne(data, (err, collection) => {

if (err) throw err;

console.log("Record Inserted Successfully");

});

return res.redirect("index.html");

});

app.post("/login", (req, res) => {

const userCollection = db.collection("users");

userCollection.findOne(

{ email: req.body.email, password: req.body.password },

(err, user) => {

if (user) {

console.log("Login successful");

res.redirect("home.html");

} else {

res.status(500).send("Login failed");

}

}

);

});

app.post("/createCustomer", (req, res) => {

var name = req.body.name;

var phno = req.body.phno;

var data = {

name: name,

phno: phno

};

db.collection("customers").insertOne(data, (err, collection) => {

if (err) throw err;

console.log("Customer Inserted Successfully");

});

res.cookie('khatabookname', name);

return res.redirect("details.html");

})

app.post("/searchCustomer", async (req, res) => {

const customerCollection = db.collection("customers");

try {

customerCollection.findOne(

{ name: req.body.userName },

(err, user) => {

if (user) {

console.log("User found");

res.cookie('khatabookname', req.body.userName);

res.redirect("details.html")

} else {

res.status(404)

console.log("User Not Found")

res.redirect("home.html");

}})

} catch (e) {

document.getElementById('errorBox').style.display = "block";

console.log(e);

}

})

app.post('/api/add', async (req, res) => {

try {

const data = req.body;

const collection = db.collection('transactions');

await collection.insertOne(data);

res.status(201).send({ message: 'Data added successfully' });

} catch (error) {

res.status(500).send({ message: 'Error adding data to MongoDB', error: error.message });

}

});

app.get('/api/data', async (req, res) => {

try {

const collection = db.collection('transactions');

const data = await collection.find({}).toArray();

res.status(200).json(data);

} catch (error) {

res.status(500).send({ message: 'Error fetching data from MongoDB', error: error.message });

}});

app.delete('/api/delete/:id', async (req, res) => {

try {

const {id} = req.params;

const collection = db.collection('transactions');

const result = await collection.deleteOne({ \_id: new mongoose.Types.ObjectId(req.params.id) });

if (result.deletedCount === 1) {

res.status(200).send({ message: 'Transaction deleted successfully' });

} else {

res.status(404).send({ message: 'Transaction not found' });

}

} catch (error) {

res.status(500).send({ message: 'Error deleting transaction from MongoDB', error: error.message });

}

});

app.get("/", (req, res) => {

res.set({

"Allow-access-Allow-Origin": "\*",

});

return res.redirect("index.html");

}).listen(3000);

console.log("Listening on Port 3000");

**package.json**

{

"name": "khatabook",

"version": "1.0.0",

"description": "Money journal book app. Built to ease shopkeepers keep track of their money out in the market.",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"dependencies": {

"body-parser": "^1.20.2",

"cookie-parser": "^1.4.6",

"express": "^4.18.3",

"mongoose": "^8.2.1",

"nodemon": "^3.1.0"

}

}

**nodemon**

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

nodemon does not require any additional changes to your code or method of development. nodemon is a replacement wrapper for node. To use nodemon, replace the word node on the command line when executing your script.

nodemon wraps your application, so you can pass all the arguments you would normally pass to your app:

nodemon [your node app]

For CLI options, use the -h (or --help) argument:

nodemon -h

Using nodemon is simple, if my application accepted a host and port as the arguments, I would start it as so:

nodemon ./server.js localhost 8080

Any output from this script is prefixed with [nodemon], otherwise all output from your application, errors included, will be echoed out as expected.

**Server startup string:**

nodemon index.js

**Success Criteria:**

The success of the KhataBook app can be evaluated based on various metrics and indicators that reflect its impact on Indian shop vendors and their businesses. These metrics encompass both quantitative measurements, such as user adoption rates and financial performance indicators, as well as qualitative assessments, including user satisfaction feedback and business growth outcomes.

The adoption rate of the KhataBook app among Indian shop vendors serves as a key indicator of its success. High levels of user registration, and active usage demonstrate the app's relevance and value to its target audience. Moreover, metrics such as session duration, frequency of app usage, and user engagement with different features provide insights into the app's usability and effectiveness in meeting users' needs.

One of the primary objectives of the KhataBook app is to minimize errors and inefficiencies associated with manual record-keeping methods. Monitoring the incidence of errors, discrepancies, and data inconsistencies in financial records before and after adopting the app can quantify the app's impact on improving data accuracy and reliability.

The app's ability to streamline financial management processes and automate repetitive tasks can result in significant time savings for shop vendors. Measuring the time spent on manual accounting tasks, such as recording transactions, tallying balances, and generating reports, before and after using the app provides insights into efficiency gains and productivity improvements achieved through digitalization.

**Conclusion:**

In conclusion, the KhataBook app emerges as a transformative solution poised to revolutionize financial management practices for Indian shop vendors. By understanding the unique challenges and requirements of this demographic, KhataBook is designed to serve as more than just a digital ledger; it's a catalyst for empowerment and growth in the small business sector.

With its user-friendly interface, advanced features, and commitment to data security and privacy, KhataBook addresses the pain points faced by Indian shop vendors in managing their finances. Through simplified transaction recording, real-time financial insights, and automated calculations, the app streamlines operations and enhances decision-making capabilities.

Moreover, KhataBook's integration with inventory management systems and payment tracking functionalities further optimize business processes, enabling vendors to maximize efficiency and capitalize on growth opportunities. The app's multi-platform accessibility ensures that shop vendors can access critical financial data anytime, anywhere, empowering them to stay in control of their businesses on the go.

Ultimately, KhataBook represents more than just a software solution; it's a testament to the potential of technology to empower small businesses and drive economic prosperity at the grassroots level. By embracing digital innovation and leveraging the power of data-driven insights, Indian shop vendors can unlock new pathways to success and chart a course towards sustainable growth and resilience in an ever-evolving marketplace.

As KhataBook continues to evolve and expand its offerings, it stands poised to become an indispensable partner for Indian shop vendors, supporting their journey towards financial stability, operational excellence, and long-term success in the dynamic and competitive business landscape of India.