Project Documentation

**Fit** **flex : Your personal Fitness Companion**.

# 1. Introduction

* **Project Title:** Fit flex
* **Team ID** :NM2025TMID47710
* **Team Leader: M.Keertha &** [**keerthukeerthi017@gmail.com**](mailto:keerthukeerthi017@gmail.com)
* **Team Members:**
* M.Kokila & madhappan753@gmail.com
* V.Lavanya & vadipushpa2004@gmail.com
* R.Madhumitha & kingrajamadhu@gmail.com
* M.Keertha & keerthukeerthi017@gmail.com

# 2. Project Overview

* **Purpose:** The purpose of the fit flex to provide a user friendly fitness platform that helps individuals achieve their health and wellness goals.
* **Features:**

Fit & Flex is a health-focused Indian brand offering baked oat-based breakfast cereals and snacks. Here are its standout features:

### Product Highlights

* **100% Baked Goodness**: All products are baked, not fried, preserving nutrients and reducing unhealthy fats.
* **Oat-Based Recipes**: Core ingredients include oats, seeds, and nuts, making them rich in fiber and protein.
* **No Added Sugar Options**: Many items, like their Power Oats, are free from added sugars.
* **Ready-to-Eat Convenience**: Designed for busy lifestyles—no cooking required for most products.
* **Flavor Variety**: Includes granola, muesli, multigrain mixtures, and mini bites in flavors like choco almond, cranberry, and peanut butter chocolate1.

### Brand & Reach

* **Founded in 2019** by Pathik Patel in Ahmedabad.
* **State-of-the-Art Facility**: A 4-acre production site with European machinery, capable of producing 375 metric tons.
* **Retail Presence**: Available in 5000+ stores across India and 1000+ international outlets in the UAE, Africa, and Maldives.
* **E-commerce Availability**: Sold on Amazon, Flipkart, Big Basket, and through vending machines.

### Unique Selling Proposition

* Combines **nutrition, taste, and convenience**—ideal for health-conscious consumers who want quick, wholesome meals.

# 3. Architecture

**Frontend:**

* **React.js**: Handles the UI, component logic, and state management.
* **Bootstrap & Material UI**: Used for responsive design and prebuilt UI components, enabling fast, visually appealing development.

**Backend:**

* **Node.js & Express.js**: Manages server-side logic, API routes, authentication, and business rules. Express.js provides a robust framework for building RESTful APIs.

**Database:**

* **MongoDB**: Stores and organizes all application data, including users, projects, applications, and chat messages. Its document-based structure is ideal for flexible, rapidly evolving schemas.

# 4. Setup Instructions

# Setup Instructions

## Prerequisites

Make sure the following are installed on your system:

- [Node.js](https://nodejs.org/)

- [MongoDB](https://www.mongodb.com/)

- [Git](https://git-scm.com/)

- [React.js](https://react.dev/)

- [Express.js](https://expressjs.com/)

- [Mongoose](https://mongoosejs.com/)

- [Visual Studio Code](https://code.visualstudio.com/)

## Installation Steps

1. \*\*Clone the Repository\*\*

```bash

git clone <repository-url>

```

2. \*\*Install Client Dependencies\*\*

```bash

cd client

npm install

```

3. \*\*Install Server Dependencies\*\*

```bash

cd ../server

npm install

```

4. \*\*Start MongoDB\*\*

Make sure your MongoDB server is running. You can typically start it with:

```bash

mongod

```

5. \*\*Start the Development Servers\*\*

- \*\*Server\*\* (Express.js):

```bash

npm start

```

(Run in the `server` directory)

- \*\*Client\*\* (React.js):

```bash

npm start

```

(Run in the `client` directory)

---

\*\*Tip:\*\*

Open the project in Visual Studio Code for the best development experience.

# 5. Folder Structure

**Explanation:**

* client/ contains the React app, with subfolders for reusable components and page views.
* server/ contains the Node.js backend, organized into routes (API endpoints), models (database schemas), and controllers (business logic).

# 6. Running the Application

**Frontend:**

1. Open a terminal and navigate to the client directory:

bash

cd client

1. Start the frontend server:

bash

npm start

**Backend:**

1. In a separate terminal window, navigate to the server directory:

bash

cd server

1. Start the backend server:

bash

npm start

**Access the Application:**

* Once both servers are running, open your browser and visit [http://localhost:3000](http://localhost:3000/) to access the application.

# 7. API Documentation

# API Documentation

## User Endpoints

- \*\*Register User\*\*

- `POST /api/user/register`

- Registers a new user.

- \*\*Login User\*\*

- `POST /api/user/login`

- Authenticates a user.

---

## Projects Endpoints

- \*\*Create Project\*\*

- `POST /api/projects/create`

- Creates a new project.

- \*\*Get Project by ID\*\*

- `GET /api/projects/:id`

- Retrieves a project by its ID.

---

## Applications Endpoint

- \*\*Apply for a Project\*\*

- `POST /api/apply`

- Submit an application for a project.

---

## Chats Endpoints

- \*\*Send Chat Message\*\*

- `POST /api/chat/send`

- Send a chat message.

- \*\*Get Chats with User\*\*

- `GET /api/chat/:userId`

- Retrieve chat messages with a specific user.

---

# 8. Authentication

## User Login and JWT Issuing

* When a user logs in, validate credentials.
* If valid, sign a JWT token (using a secret).
* Send the token to the client (usually in the response body or as an HTTP-only cookie).

## 2. Middleware for Route Protection

* Create a middleware that checks for a JWT token in the request (header/cookie).
* Verify the token using your secret.
* If valid, allow access to the route; if not, block access.

## Example Implementation

**auth.js**

const jwt = require('jsonwebtoken');

// Secret key for signing JWTs

const JWT\_SECRET = process.env.JWT\_SECRET || 'your\_jwt\_secret';

// Middleware to protect private routes

**routes.js**

const express = require('express');

const jwt = require('jsonwebtoken');

const { authenticateToken, JWT\_SECRET } = require('./auth');

const router = express.Router();

// Example login route

## Usage

* On login, client stores the JWT (localStorage, cookie, etc.)
* For private requests, client sends JWT in Authorization header:  
  Authorization: Bearer <token>
* Middleware authenticates each request.

# 9. User Interface

### Types of User Interfaces

* **Graphical User Interface (GUI):**
  + Uses visual elements like windows, icons, and buttons.
  + Common in desktops, smartphones, and web apps.
* **Command-Line Interface (CLI):**
  + Text-based interface where users type commands.
  + Used by developers and system administrators.
* **Voice User Interface (VUI):**
  + Interaction through voice commands (e.g., Siri, Alexa).
* **Touch User Interface:**
  + Found in smartphones and tablets; uses gestures like tapping and swiping.
* **Natural User Interface (NUI):**
  + Uses gestures, facial recognition, or motion (e.g., Kinect, AR/VR systems).

### Key Elements of a Good UI

* **Clarity:** Easy to understand and navigate.
* **Consistency:** Uniform design and behavior across the app.
* **Feedback:** Visual or audio cues to show actions were successful.
* **Efficiency:** Helps users complete tasks quickly.
* **Accessibility:** Usable by people with disabilities.

# 10. Testing

* Manual testing during milestones

## Landing Page

**Purpose**: Public homepage. Showcases features, login/signup options, benefits for freelancers & clients.

**Key Sections**:

* Hero section: app logo, tagline, call to action (CTA)
* Features overview
* Testimonials
* Login/Signup buttons or links

## **2. Freelancer Dashboard**

**Purpose**: Main hub for freelancers.

**Features**:

* Project list (active, pending, completed)
* Earnings summary
* Profile management (skills, bio, availability)
* Notifications/messages
* Quick access to apply for new projects

## **3. Admin Panel**

**Purpose**: Manage users, projects, payments, site settings.

**Features**:

* User management (view, block, promote)
* Project moderation
* Financial/reporting tools
* Dashboard statistics (signups, revenue, active users)
* Settings/configuration

## **4. Project Details Page**

**Purpose**: In-depth view for a single project.

**Features**:

* Project description, requirements
* Client info
* Application status (applied, accepted, rejected)
* Messaging/comments section
* Deliverables & milestones

### Example Directory Structure (React)

plaintext

src/

components/

LandingPage/

FreelancerDashboard/

AdminPanel/

ProjectDetailsPage/

App.js

routes.js

### **Example React Component Skeletons**

**LandingPage.js**

import React from 'react';

function LandingPage() {

return (

<div>

<header>

**FreelancerDashboard.js**

import React from 'react';

function FreelancerDashboard() {

return (

<div>

<h2>Freelancer Dashboard</h2>

**AdminPanel.js**

import React from 'react';

function AdminPanel() {

return (

<div>

<h2>Admin Panel</h2>

**ProjectDetailsPage.js**

import React from 'react';

function ProjectDetailsPage() {

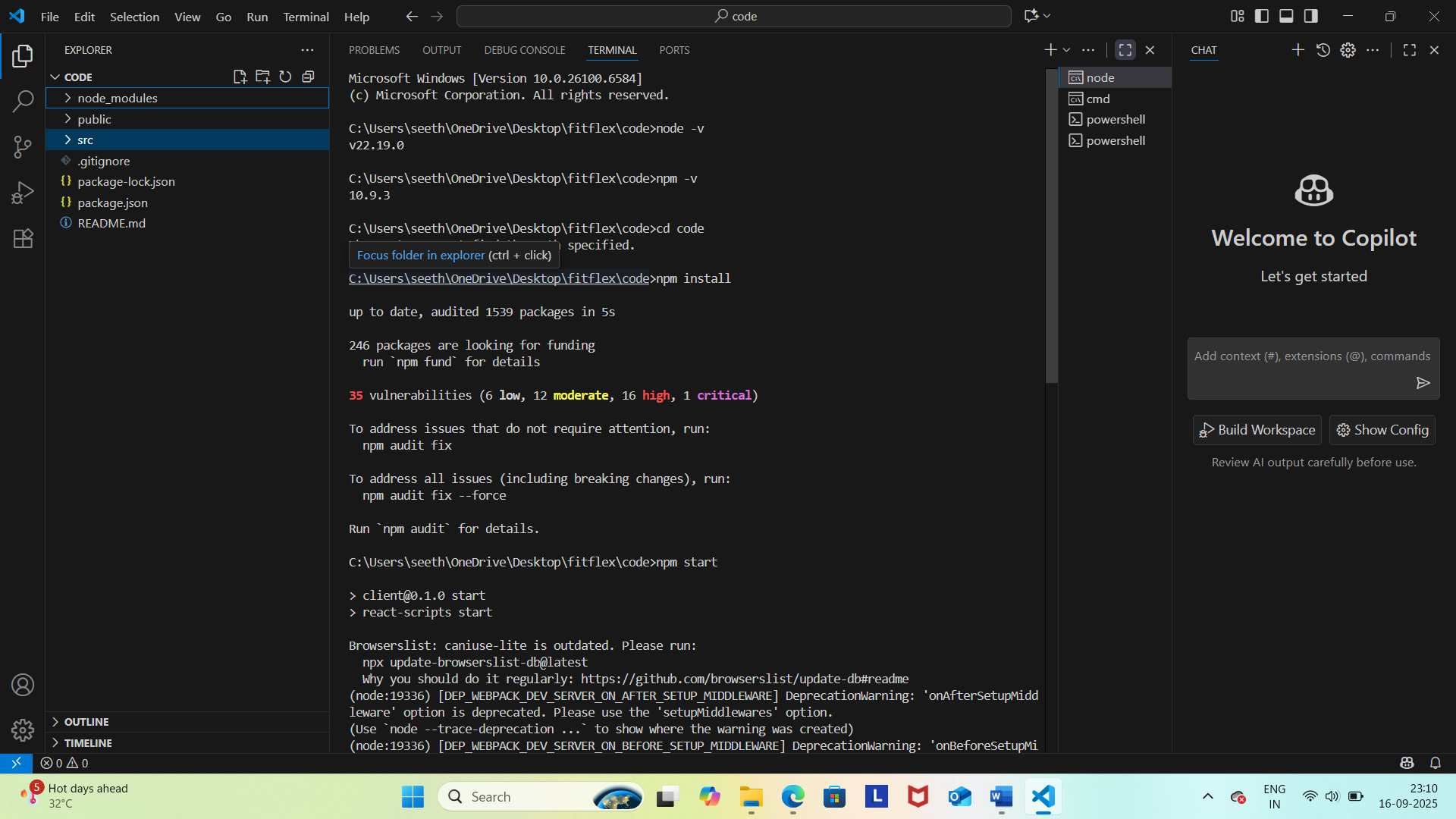
return (

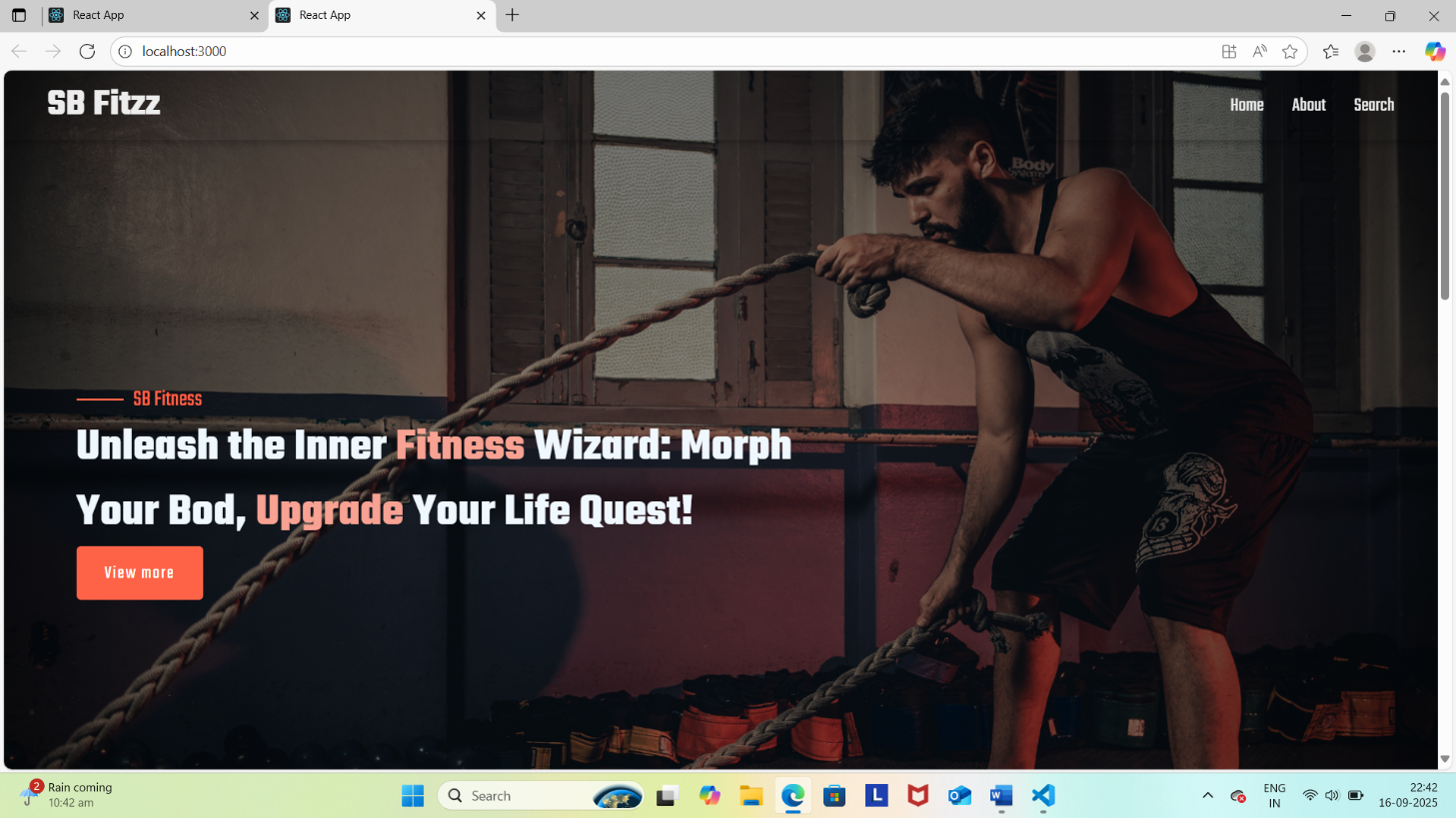
<div>

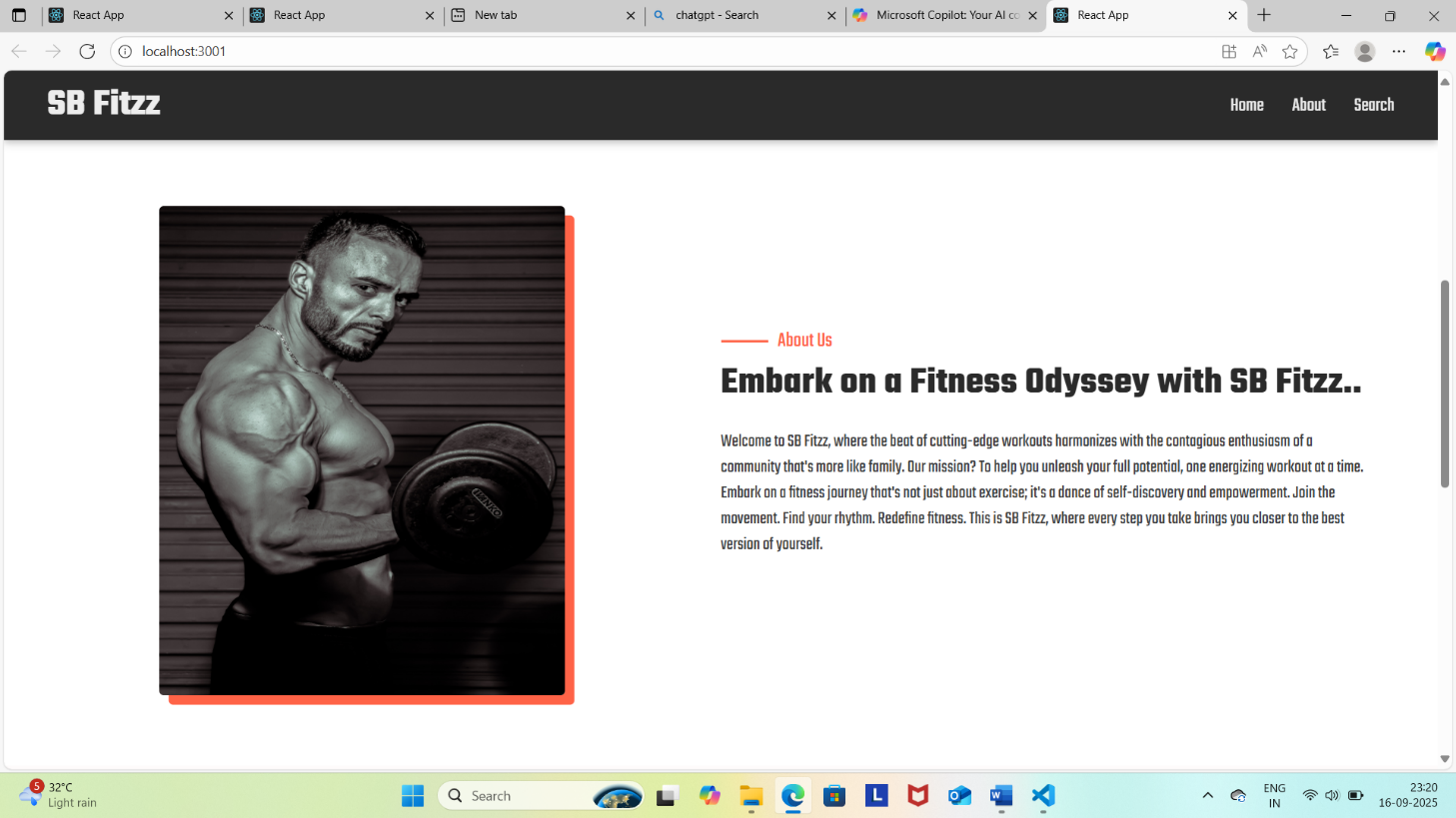
<h2>Project Details</h2>

# Existing:[ https://github.com/kingrajamadhu-hub/[fitflex](https://github.com/kingrajamadhu-hub/fitflex.git).git ]

1. **Screenshots or Demo**

1.





## 2. **Demo (Interactive or Video)**

* Embed a demo video (YouTube, Loom, etc.) on your landing page or README.
* Optionally, provide a live link to a deployed app for users to try.

**Example for README:**

**README.md**v2

**## Demo**

[Watch Demo Video](https://www.youtube.com/watch?v=your-demo-link)

[Try Live App](https://fitflex-demo.vercel.app)

## 3. **Best Practices**

* Store screenshots in a screenshots/ folder in your repo.
* Use clear, high-resolution images.
* Update screenshots if major UI changes are made.

1. **Known Issues**

Here’s a template and example you can use for your FitFlex project:

**README.md**v3

**## Known Issues**

- **\*\*Authentication:\*\***

- JWT tokens do not auto-refresh; users must log in again after expiry.

- No password reset functionality implemented.

**How to use:**

* Place this section in your README.md or a dedicated KNOWN\_ISSUES.md file.
* Update regularly as bugs are fixed or new ones are discovered.
* Encourage users to submit issues on GitHub for transparency and tracking.

**13 Future Enhancements**

1. **AI Integration & Automation**
   * Implement intelligent systems to streamline workflows, reduce manual effort, and improve decision-making.
2. **User Experience (UX) Improvements**
   * Enhance interfaces for better accessibility, personalization, and intuitive navigation.
3. **Sustainability Initiatives**
   * Adopt eco-friendly practices, materials, or energy sources to reduce environmental impact.
4. **Data Security & Privacy**
   * Strengthen encryption, compliance, and user control over data to build trust and resilience.
5. **Scalability & Performance Optimization**
   * Upgrade infrastructure to handle growth efficiently without compromising speed or reliability.
6. **Cross-Platform Compatibility**
   * Ensure seamless functionality across devices, operating systems, and browsers.
7. **Cloud Migration & Services**
   * Transition to cloud-based solutions for flexibility, collaboration, and cost-effectiveness.
8. **Advanced Analytics & Insights**
   * Use predictive analytics and real-time dashboards to uncover trends and guide strategy.
9. **Training & Skill Development**
   * Invest in upskilling teams or users to adapt to evolving tools and technologies.
10. **Global Expansion & Localization**
    * Tailor offerings to new markets with language, cultural, and regulatory adaptations.
11. **Community Engagement & Feedback Loops**
    * Create channels for user input and foster a loyal, active community.
12. **Innovation Labs or R&D**
    * Dedicate resources to exploring emerging technologies and testing bold ideas.
13. **Ethical & Inclusive Design**
    * Prioritize fairness, accessibility, and representation in all products and services.