iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75327

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Skill Swap - A Collaborative Learning Platform

Abinash K[1], Ajaykumar S[2], Arunachalam S[3], Mrs. M. Malathi[4]
[1, 2, 3]*Bachelor of Engineering in Computer Science and Engineering, Adhiyamaan College of Engineering, (An Autonomous Institution), ANNA University: Chennai*
[4]*Assistant Professor, Computer Science and Engineering, Adhiyamaan College Of Engineering, (An Autonomous Institution), ANNA University: Chennai*

Abstract: ***The Skill Swap Web Application is a platform designed to promote collaborative learning through skill exchange. It enables users to offer the skills they can teach and request the skills they wish to learn, creating a community-driven network of mutual knowledge sharing. Developed using HTML, Tailwind CSS, TypeScript, Node.js, Express.js and Supabase. The application provides secure user authentication with JWT tokens. Users can create personalized profiles, manage their offered and wanted skills, and connect with others through a smart matching algorithm that identifies suitable skill partners. A real-time chat system built with Socket.io allows matched users to communicate instantly. The responsive frontend ensures smooth accessibility across all devices, while the backend provides RESTful APIs for authentication, skill management, matching and messaging. Overall, the Skill Swap system bridges the gap between learners and teachers, fostering a collaborative environment that encourages continuous skill development and knowledge exchange in a digital community.***

I. INTRODUCTION

A. *Overview*

In today's rapidly evolving digital world, learning is no longer limited to formal education or institutional boundaries. People continuously seek opportunities to gain new skills, share their expertise, and grow together. The Skill Swap web application is designed to meet this need by providing an online platform where individuals can exchange skills and knowledge in a collaborative environment.

This system enables users to offer the skills they can teach and request the skills they wish to learn, forming a self-sustaining community of learners and educators. Unlike conventional e-learning platforms that rely on paid courses, Skill Swap promotes mutual learning without monetary exchange, making education more accessible, interactive and community-driven.

Built using HTML, Tailwind CSS, TypeScript, Node.js, Express.js and Supabase, the application ensures secure authentication, efficient data management and seamless real-time communication through Socket.io. Ultimately, Skill Swap serves as a bridge between learners and informal educators, encouraging peer-to-peer learning, knowledge sharing and personal growth in a modern digital ecosystem.

B. *Objectives*

The specific objectives are:

1) To develop a responsive and user-friendly web application that allows users to register, offer and request skills easily.
2) To implement a secure authentication mechanism using JWT and Supabase for protecting user data and ensuring reliable access control.
3) To design and integrate a smart matching algorithm that connects users based on their offered and desired skills.
4) To enable real-time communication between matched users through a built-in chat system powered by Socket.io.
5) To ensure scalability and efficient data management using Node.js, Express.js and Supabase for backend operations.
6) To promote community-based learning by encouraging collaboration, mutual teaching and knowledge exchange in a digital environment.

II. LITERATURE SURVEY

A review of existing literature reveals that the digitization of university administrative tasks has been a persistent trend aimed at improving operational efficiency.

Several studies, such as,

1) World Economic Forum (Future of Jobs Report) emphasizes that practical skills are becoming more valuable than formal degrees in the modern workforce. This shift has led to the development of platforms where individuals can exchange real-world skills, making education more flexible and accessible.
2) Simbi, an online skill-exchange platform, allows users to trade services using a non-monetary model. This real-world example demonstrates how a community-driven learning environment can function effectively without the involvement of money, promoting inclusivity and collaboration among participants.
3) Timebank, another peer-exchange system, uses time as a currency to facilitate skill sharing. Participants exchange one hour of their service for one hour of another's, showcasing the potential of reciprocal learning models in fostering social and educational growth.
4) Research by Patel and Kim (2022) on modern JavaScript frameworks highlights the effectiveness of technologies like Node.js, Express.js and React/TypeScript in building responsive, real-time web applications. These tools provide scalability and user interactivity essential for platforms like Skill Swap.
5) Garcia (2023) discusses the role of APIs and real-time communication technologies such as WebSockets and Socket.io in enabling seamless user interaction. This research supports the integration of real-time chat systems and dynamic updates in modern web platforms.
6) Studies on Supabase and serverless database solutions demonstrate how cloud-based storage and authentication systems can simplify backend development while maintaining data security and real-time synchronization.

III. SYSTEM ANALYSIS

A. *Existing System*

1) Most online platforms such as Coursera and Udemy follow a course-based approach where learners passively consume pre-recorded lessons instead of engaging in direct knowledge exchange.
2) There is very little opportunity for learners to connect and collaborate with others who share similar interests, as interaction is often limited to discussion forums.
3) Many systems operate on a paid subscription model, which restricts access for users who cannot afford premium content and reduces inclusivity.
4) The absence of personalized learning mechanisms makes it difficult for users to find others with matching skill interests or compatible learning goals.
5) Real-time communication features are rarely integrated, which limits direct interaction and immediate feedback between learners and instructors.
6) Existing platforms do not encourage community-based learning where users can play dual roles as both teachers and learners.
7) Accessibility is also limited, as many systems are not fully responsive across different devices, reducing flexibility for users who prefer mobile or low-end platforms.

Disadvantages

1) Limited interaction between learners, reducing opportunities for collaboration.
2) Paid learning models create financial barriers for many users.
3) Absence of real-time communication features for direct engagement.
4) No smart matching system to connect users based on skills and interests.
5) Lack of a community-based platform that supports mutual teaching and learning.

B. *Proposed System*

1) The system enables users to register, create profiles and list the skills they can offer and the skills they wish to learn.
2) A smart matching algorithm automatically connects users based on compatible skill interests, ensuring relevant and effective learning pairs.
3) Secure authentication is implemented using JWT and Supabase to protect user information and maintain data privacy.
4) Real-time chat functionality powered by Socket.io allows users to communicate instantly and share knowledge seamlessly.
5) A responsive web interface built with TypeScript ensures smooth usability across all devices.

6) The system eliminates the need for paid subscriptions, making learning completely free and accessible to everyone.
7) The platform encourages community-based interaction, where users can act as both learners and teachers, promoting collaboration and continuous skill development.

Advantages

- Collaborative: Promotes effective peer-to-peer learning by enabling users to both teach and learn skills in a shared, interactive environment.
- Cost-Free: Eliminates monetary barriers by providing a non-paid, community-based platform for knowledge exchange.
- Interactive: Enhances engagement through real-time chat communication and instant collaboration between matched users.
- Secure: Ensures data protection and user privacy with JWT authentication and Supabase database management.
- Responsive: Offers a modern and user-friendly interface built with Bootstrap and TypeScript, accessible on all devices.

C. *Proposed Solution*

1) The system provides a centralized web platform where users can register, create profiles and list the skills they wish to offer and learn.
2) A smart matching algorithm is implemented to automatically pair users based on their offered and desired skills, ensuring relevant and meaningful connections.
3) Secure authentication is established using JWT and Supabase to verify users and protect their personal information during login and data transactions.
4) A real-time communication module using Socket.io allows matched users to chat instantly, discuss lessons, and plan learning sessions effectively.
5) The frontend interface is designed with Bootstrap and TypeScript, ensuring a responsive, intuitive and device-friendly user experience.
6) The backend system, built with Node.js and Express.js, manages all API operations, data handling, and matching processes for smooth functionality.
7) The solution emphasizes community-driven learning, where users can act as both learners and instructors, creating an inclusive environment that promotes skill sharing and continuous growth.

D. *Problem Solution Fit*

1) Lack of Peer-to-Peer Collaboration:
   - Problem: Most existing online learning platforms follow a one-way teaching model where learners only consume information without active interaction or mutual skill exchange.
   - Solution: Skill Swap introduces a two-way learning approach that allows users to teach and learn from each other. This collaborative model promotes direct interaction and builds a strong community of shared learning.

2) Monetary Restrictions on Learning:
   - Problem: Many e-learning systems require paid memberships or course fees, which limits access for users who cannot afford premium content.
   - Solution: The Skill Swap platform provides a completely free learning environment by enabling non-monetary skill exchange. Users can gain knowledge or teach others without financial barriers, ensuring equal learning opportunities for all.

3) Absence of Real-Time Communication and Personalization:
   - Problem: Traditional learning systems often lack real-time interaction and personalized matching, making it difficult for learners to connect with suitable partners.
   - Solution: Skill Swap uses Socket.io for real-time chat and a smart matching algorithm to connect users based on their offered and desired skills, ensuring immediate communication and relevant learning experiences.

4) Limited Accessibility and Engagement:
- Problem: Some existing systems are not optimized for all devices, reducing user engagement and accessibility for mobile users.

- Solution: Skill Swap is designed with a responsive interface and Supabase backend, providing seamless usability across all devices and maintaining user engagement through an interactive, community-driven environment.

E. Architecture Design

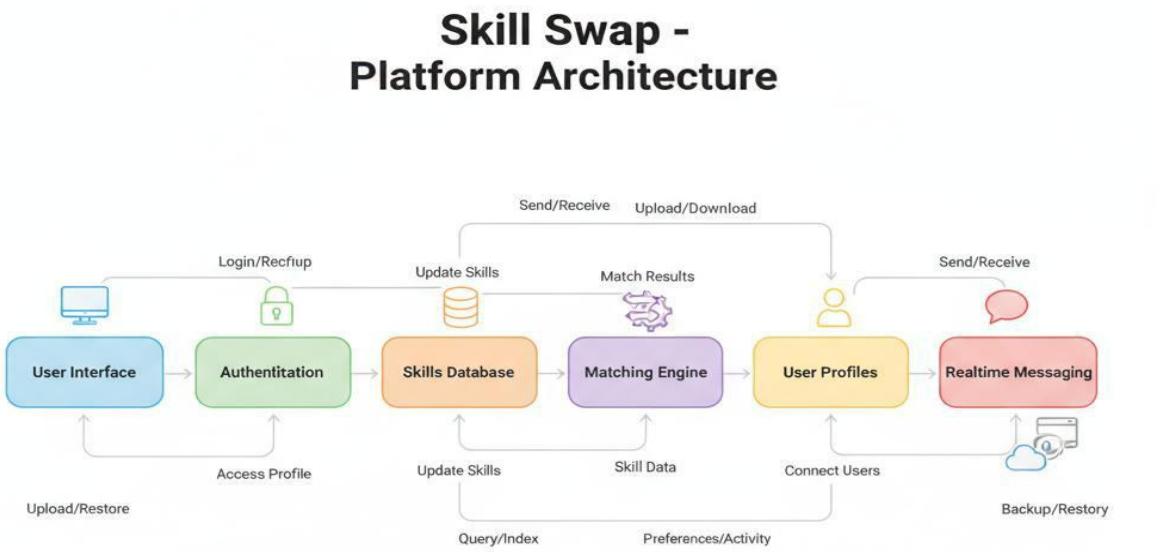


Figure 1: Architecture Diagram

The figure shown above represents the Solution Architecture that we made use of in our Project

- Frontend Layer: Developed using HTML, Tailwind CSS and TypeScript, this layer provides a responsive and interactive user interface. It allows users to register, log in, create profiles, list their skills and communicate easily with others.
- Backend Layer: The backend, built with Node.js and Express.js, handles all server-side logic such as authentication, skill matching, chat requests and data communication. It acts as the bridge between the user interface and the database.
- Database and Security Layer: The system uses Supabase for data storage and JWT authentication for secure login and data protection. This layer manages user credentials, skill data and chat history, ensuring reliability, privacy and scalability.
- Real-Time Interaction Module: Integrated with Socket.io, this module enables instant communication between matched users. It supports live chat, feedback sharing and continuous collaboration making learning interactive and engaging.

Data Flow Example (Application Submission):

1) A user fills out the registration or skill submission form on the Skill Swap frontend built with HTML, Tailwind CSS and TypeScript, then clicks “Add Skill”.
2) The frontend application packages the form data (user details, offered skills, and desired skills) into a POST request and sends it to the backend API endpoint, for example: http://localhost:5000/api/user/skillswap.
3) The Express.js backend server receives the request. The CORS middleware allows communication between the frontend (port 3000) and backend (port 5000), ensuring secure cross-origin access.
4) The backend validates the incoming data and checks the user’s authentication token using JWT (JSON Web Token) to verify identity and session validity.
5) The validated data (user info, skill sets, and preferences) is processed by the API route handler, which inserts it as a new record into the Supabase database under the corresponding user profile.
6) The smart matching algorithm is triggered to analyze the offered and desired skills, identifying potential matches and storing them for later retrieval.
7) The server sends a JSON response (e.g., { status: ‘success’, message: ‘Skills submitted successfully’ }) back to the frontend, which displays a confirmation message or next-step instruction to the user.

F. Description Modules

1) User Registration and Profile Management

The User Registration and Profile Management module serves as the foundation of the Skill Swap platform, providing users with a simple and secure onboarding experience. New users can register by entering their personal details, email and password, which are securely stored using JWT authentication and managed by Supabase. Once registered, users can create and update their profiles with information such as their offered skills, desired skills, experience level and personal bio.

This module allows users to personalize their learning experience and make their profiles discoverable for matching. It also includes features for password recovery, email verification and profile editing. The smooth and responsive interface, built with HTML and Tailwind CSS ensures an engaging and user-friendly experience across all devices.

2) Skill Listing and Smart Matching

The Skill Listing and Smart Matching module forms the core of the Skill Swap system, enabling users to list both the skills they can teach and the skills they wish to learn. Once users submit their skill data, the backend processes the information and applies a smart matching algorithm that identifies compatible partners based on their skill preferences.

The algorithm compares multiple parameters such as skill category, proficiency level and availability to generate accurate and relevant matches. Users are then notified about potential partners, allowing them to connect and collaborate easily. This automated system eliminates the need for manual searching and ensures efficient skill pairing for every user.

3) Real-Time Communication System

The Real-Time Communication System module enables instant interaction between matched users through a secure chat interface. Built with Socket.io, it supports real-time text communication, ensuring that learners and instructors can exchange messages, resources and learning materials instantly.

This module also allows users to schedule learning sessions, clarify doubts and share feedback during or after discussions. The real-time system enhances engagement, improves collaboration and makes the entire skill exchange process interactive, thereby replicating a face-to-face learning experience in a digital environment.

4) Feedback and Rating Management

The Feedback and Rating Management module allows users to provide feedback and ratings after completing a learning session. Each participant can evaluate their partner's teaching effectiveness, communication and overall experience. These ratings are stored in the database and help improve the reliability of future matches.

This system encourages accountability and ensures quality in peer-to-peer learning. The accumulated feedback also enables the platform to highlight active and well-rated users, fostering a healthy, trustworthy and motivating learning community.

5) Data Management

The Data Management module is responsible for securely handling and organizing all information within the platform. It ensures that user data, skill listings, chat records and feedback reports are efficiently stored, retrieved and maintained. Using the Supabase database, data is managed through structured tables with proper authentication and access control to ensure privacy and integrity. This module also supports data analytics and reporting, enabling the generation of insights such as user growth, active skill exchanges and feedback summaries. Administrators can export relevant datasets in Excel format for further analysis and record-keeping. Overall, this module ensures the accuracy, security and reliability of all data operations within the system

IV. SYSTEM REQUIREMENTS

A. Hardware Requirements

The Skill Swap – A Collaborative Learning Platform is a web-based full-stack application that operates efficiently on standard computing hardware. It does not require specialized equipment but performs best on modern systems with stable internet connectivity.

1) Server

- Processor: Minimum 1.8 GHz Dual-Core CPU or better.

- RAM: Minimum 4 GB (8 GB recommended for optimal performance).
- Storage: At least 40 GB of available disk space (SSD preferred for faster data access).

2) Client Devices

- Processor: 1 GHz or faster.
- RAM: Minimum 2 GB for smooth browser-based operation.

3) Network Infrastructure

- A stable internet connection such as Broadband, Wi-Fi or 4G/5G is required for real-time communication between clients and the server.
- Standard networking hardware such as routers and switches to ensure uninterrupted connectivity.

B. Software Requirements

The Skill Swap system is designed using modern web technologies for scalability, reliability and ease of deployment. The software requirements are as follows:

Operating System

- Server-Side: Any 64-bit modern OS such as Linux (Ubuntu 20.04+), Windows Server 2019+ or macOS Monterey+.
- Client-Side: Compatible with all major OS platforms — Windows 10+, macOS 10.15+, Android 8+, or iOS 14+.

Web Server

- Node.js runtime environment (includes an HTTP server module to handle client requests).
- For production, use a reverse proxy server such as Nginx or Apache to enhance security, optimize performance and manage traffic efficiently.

Database

- Supabase (PostgreSQL-based cloud database) is used for secure, scalable and real-time data management.

Core Frameworks and Languages

- Programming Language: JavaScript (ES6+).
- Backend Environment: Node.js (v18 or higher).
- Backend Framework: Express.js.
- Frontend Technologies: HTML5, Tailwind CSS and TypeScript for responsive design.
- Real-Time Communication: Socket.io for instant chat functionality.

Web Browser (Client-Side)

- Any modern browser supporting HTML5, Tailwind CSS and ES6 JavaScript — such as Google Chrome, Mozilla Firefox, Microsoft Edge or Safari.

Development and Deployment Tools

- Version Control: Git or GitHub for source code management.
- Code Editor: Visual Studio Code or any modern IDE.

V. IMPLEMENTATION

A. User Registration And Profile Creation

The User Registration and Profile Creation module is the first point of interaction for individuals using the Skill Swap platform. Users begin by signing up through a secure and responsive web interface, providing essential details such as their name, email and password. The authentication process is handled using JWT (JSON Web Tokens) to ensure that all sessions are secure and private. Once registered, users can log in to their personalized dashboard and create a detailed profile showcasing the skills they can teach and those they wish to learn. The system allows them to edit their bio, upload a profile picture and specify their skill categories. This

module provides a seamless onboarding experience that sets the foundation for effective peer-to-peer learning and personalized matching.

B. Skill Listing And Smart Matching

The Skill Listing and Smart Matching module represents the core logic of the Skill Swap system. After setting up their profiles, users can list the skills they are willing to teach and select those they are interested in learning. This information is stored securely in the Supabase database.

Once submitted, the smart matching algorithm runs on the backend (Node.js + Express.js) to identify compatible partners by comparing offered and desired skills. The system automatically suggests potential matches, allowing users to view recommended partners directly in their dashboard. This automated process removes the need for manual searching and ensures accurate, efficient and meaningful skill connections.

C. Real-Time Chat And Colloboration

The Real-Time Chat and Collaboration module enables instant communication between matched users. Built with Socket.io, this feature allows users to send and receive messages instantly without refreshing the page. The chat interface supports real-time discussions, session scheduling and sharing of learning resources. By facilitating direct interaction, this module transforms learning into a collaborative and interactive experience. It not only strengthens user engagement but also replicates the dynamics of in-person learning in a digital environment, encouraging trust, cooperation and continuous skill development.

D. Software Description

1) HTML (Hypertext Markup Language): HTML forms the structural foundation of the Digital Bus Pass Management System's web interface. It is used to create and organize the content on every page, from the student application form to the admin dashboard. Structural elements like <form>, <input> and <button> are used to build the interactive application form, while tags like , <header> and <footer> define the layout of the pages. Semantic HTML is used to improve accessibility and SEO, ensuring that elements like headings (<h1>, <h2>) and lists (<ul>, <li>) provide clear context to both users and search engines. This structured approach ensures that the application is logical, accessible and easy to navigate.

2) Tailwind CSS: Tailwind CSS is a modern utility-first CSS framework used to design responsive and visually appealing user interfaces efficiently. In this project, Tailwind CSS is implemented to streamline the styling process by applying predefined utility classes directly within the HTML structure. This approach eliminates the need to write custom CSS, allowing faster development and consistent design. Tailwind offers a wide range of utilities to control layout (using Flexbox and Grid), colors, typography, spacing, and animations. These responsive utilities enable developers to easily build adaptive layouts, ensuring the interface remains user-friendly and visually balanced on any screen size.

3) JavaScript: JavaScript is the core programming language that powers the dynamic and interactive features of the application on both the client and server sides.

- Client-Side (Frontend): In the browser, JavaScript is used with the React library to build a responsive and fast single-page application (SPA). It manages the application's state (e.g., form inputs), handles user events (like button clicks and form submissions) and makes asynchronous API calls to the backend to send and receive data without reloading the page. This creates a smooth and seamless user experience.
- Server-Side (Backend): On the server, JavaScript is run in the Node.js environment. It is used to build the entire backend API, handling incoming requests, interacting with the database, managing file uploads and implementing the core business logic of the application.

4) Node.js: Node.js is a server-side JavaScript runtime environment that serves as the backbone of the application's backend. It allows JavaScript to be used for server-side scripting, enabling the creation of a fast, scalable and efficient API.

- Asynchronous and Event-Driven: Node.js uses a non-blocking, event-driven architecture, which makes it highly efficient for handling many concurrent connections. This is ideal for a web application where multiple students and admins might be accessing the system simultaneously.

- Express.js Framework: The project utilizes Express.js, a minimal and flexible Node.js web application framework. Express simplifies the process of building the RESTful API by providing robust features for routing (defining API endpoints like /api/student/apply), middleware integration (for handling file uploads with Multer and enabling CORS) and managing HTTP requests and responses.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

- Database Interaction: Node.js, through the sqlite3 library, connects to and interacts with the SQLite database. It executes SQL queries to create, read, update and delete student application data, forming the crucial link between the application logic and the data store.

E. Code Implementation

Step 1: Set up the Profile of the User

The Skill Submission Form is the main interface where users enter their details, list the skills they can teach, and select the skills they wish to learn.

```
// src/pages/profile.tsx

import { useEffect, useState } from “react”;

import { supabase } from “@/integrations/supabase/client”; import { Button } from “@/components/ui/button”;

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from “@/components/ui/card”;

import { Input } from “@/components/ui/input”; import { Label } from “@/components/ui/label”; import { Textarea } from “@/components/ui/textarea”; import Navbar from “@/components/Navbar”; import { useNavigate } from “react-router-dom”; import { useToast } from “@/hooks/use-toast”; import { User } from “lucide-react”; const Profile = () => {

  const [loading, setLoading] = useState(true);

  const [saving, setSaving] = useState(false);

  const [profile, setProfile] = useState({

    full_name: “”,

    bio: “”,

  });

  const navigate = useNavigate();

  const { toast } = useToast();

  useEffect(() => {

    checkAuthAndLoadProfile();

  }, []);


  const checkAuthAndLoadProfile = async () => {

    const { data: { session } } = await supabase.auth.getSession();


    if (!session) {

      navigate(“/auth”);
```

```
    return;
  }

  loadProfile();
};

const loadProfile = async () => {
  try {
    const { data: { user } } = await supabase.auth.getUser(); if (!user) return;

    const { data, error } = await supabase
      .from("profiles")
      .select("*")
      .eq("id", user.id)
      .single();
    if (error) throw error;

    if (data) {
      setProfile({
        full_name: data.full_name || "",
        bio: data.bio || "",
      });
    }
  } catch (error) { toast({
      title: "Error",
      description: "Failed to load profile.",
```

```
      variant: “destructive”, });
  } finally {
    setLoading(false);
  }
};
const handleSubmit = async (e: React.FormEvent) => { e.preventDefault();
  setSaving(true);

  try {
    const { data: { user } } = await supabase.auth.getUser(); if (!user) return;
    const { error } = await supabase
      .from(“profiles”)
      .update({
        full_name: profile.full_name,
        bio: profile.bio,
      })
      .eq(“id”, user.id);

    if (error) throw error;

    toast({
      title: “Success”,
      description: “Profile updated successfully!”,
    });
  } catch (error) { toast({
      title: “Error”,
      description: “Failed to update profile.”,
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
      variant: “destructive”, });
  } finally {
    setSaving(false);
  }
};

if (loading) {
  return (
    <div className=”min-h-screen bg-background”> <Navbar />
      <div className=”container mx-auto px-4 py-8”>
        <p className=”text-center text-muted-foreground”>Loading…</p> </div>
    </div>
  );
}

return (
  <div className=”min-h-screen bg-background”> <Navbar />

    <div className=”container mx-auto px-4 py-8”> <div className=”max-w-2xl mx-auto”>
        <div className=”mb-8”>
          <h1 className=”text-4xl font-bold mb-2”>My Profile</h1>
          <p className=”text-muted-foreground”>Manage your account information</p> </div>

        <Card>
          <CardHeader>
            <div className=”flex items-center gap-4”>
```


ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
<div className="h-16 w-16 rounded-full bg-gradient-to-br from-primary to-secondary flex items-center justify-center">
<User className="h-8 w-8 text-primary-foreground" /> </div>
<div>
<CardTitle>Profile Information</CardTitle> <CardDescription>Update your personal details</CardDescription>
</div>
</div>
</CardHeader>
<CardContent>
<form onSubmit={handleSubmit} className="space-y-6"> <div className="space-y-2">
<Label htmlFor="full_name">Full Name</Label> <Input
id="full_name"
value={profile.full_name}
onChange={(e) => setProfile({ ...profile, full_name: e.target.value })}
placeholder="Your full name"
required
/>
</div>

<div className="space-y-2">
<Label htmlFor="bio">Bio</Label>
<Textarea
id="bio"
value={profile.bio}
onChange={(e) => setProfile({ ...profile, bio: e.target.value })} placeholder="Tell us about yourself..." rows={4}
/>
</div>
```


ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
<Button type="submit" variant="hero" className="w-full" disabled={saving}> {saving ? "Saving…" : "Save
Changes"}

</Button>

</form>

</CardContent>

</Card>

</div>

</div>

</div>

);

};

export default Profile;
```

Step 2: Search the Relevant Skill that matches the two User
The system automatically searches for relevant skills that match between two users based on their offered and desired skills.

```
//src/pages/Browse.tsx

import { useEffect, useState } from "react";

import { supabase } from "@/integrations/supabase/client";

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge"; import { Input } from "@/components/ui/input"; import { Button } from
"@/components/ui/button";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import Navbar from "@/components/Navbar"; import { useNavigate } from "react-router-dom"; import { Search, UserCircle } from
"lucide-react"; import { useToast } from "@/hooks/use-toast";


interface Skill {

  id: string;

  title: string;
```

```
  description: string;
  category: string;
  skill_level: string;
  time_commitment: string | null;
  looking_for: string[] | null;
  user_id: string;
  created_at: string | null;
  updated_at: string | null;
  profiles: {
    full_name: string;
  };
}
const Browse = () => {
  const [skills, setSkills] = useState<Skill[]>([]);
  const [filteredSkills, setFilteredSkills] = useState<Skill[]>([]); const [searchQuery, setSearchQuery] = useState("");
  const [categoryFilter, setCategoryFilter] = useState("all"); const [loading, setLoading] = useState(true); const { toast } =
  useToast();
  const [currentUserId, setCurrentUserId] = useState<string | null>(null);
   const [requestStatusBySkill, setRequestStatusBySkill] = useState<Record<string, { status:
string; id: string }>>({});
  const [connectedUsers, setConnectedUsers] = useState<Set<string>>(new Set()); const navigate = useNavigate();

  useEffect(() => {
    const init = async () => {
      const { data: authData } = await supabase.auth.getUser(); setCurrentUserId(authData.user?.id ?? null); await fetchSkills();
      if (authData.user?.id) {
```

```
    await fetchMyRequests(authData.user.id); await fetchConnectedUsers(authData.user.id);
    subscribeToMyRequestChanges(authData.user.id);
   }

  };

  init();

}, []);


useEffect(() => {

  filterSkills();

}, [searchQuery, categoryFilter, skills]);

const fetchSkills = async () => {

  try {

   console.log("Fetching all skills from database…");


   const { data, error } = await supabase

    .from("skills")

    .select(`

     *,

     profiles (

      full_name

     )

    `)

    .order("created_at", { ascending: false });


   if (error) {

    console.error("Error fetching skills:", error);
```

```
      throw error;
    }

    console.log("Skills fetched from database:", data);
    const list = (data || []) as unknown as Skill[];
    setSkills(list);
    setFilteredSkills(list);
  } catch (error) {
    console.error("Failed to load skills:", error);
    toast({
      title: "Error",
      description: "Failed to load skills. Please check the console for details.",
      variant: "destructive",
    });
  } finally { setLoading(false);
  }
};

const fetchMyRequests = async (userId: string) => { const { data, error } = await (supabase as any)
    .from("skill_exchanges")
    .select("id,skill_id,status")
    .eq("requester_id", userId);
  if (!error && data) {
    const map: Record<string, { status: string; id: string }> = {}; for (const row of data as any[]) {
      map[row.skill_id] = { status: row.status, id: row.id };
    }
```

```
    setRequestStatusBySkill(map);

  }

};
const fetchConnectedUsers = async (userId: string) => { try {

            // Fetch accepted exchanges where user is requester const { data: asRequester } = await (supabase as any)
    .from("skill_exchanges")
    .select("skills(user_id)")

    .eq("requester_id", userId)

    .eq("status", "accepted");



  // Fetch accepted exchanges where user is skill owner const { data: asOwner } = await (supabase as any)
    .from("skill_exchanges")

    .select("requester_id")

    .eq("status", "accepted")

    .in("skill_id", (

      await (supabase as any).from("skills").select("id").eq("user_id", userId) ).data?.map((s: any) => s.id) || []);


  const connectedUserIds = new Set<string>();


  // Add skill owners from requester exchanges if (asRequester) {

    asRequester.forEach((exchange: any) => { if (exchange.skills?.user_id) {

      connectedUserIds.add(exchange.skills.user_id);

    }

    });

  }


  // Add requesters from owner exchanges

  if (asOwner) {

    asOwner.forEach((exchange: any) => {
```

```
        if (exchange.requester_id) {
          connectedUserIds.add(exchange.requester_id);
                                                                                                    });
      }
      setConnectedUsers(connectedUserIds);
    } catch (error) {
      console.error("Error fetching connected users:", error);
    }
  };

  const subscribeToMyRequestChanges = (userId: string) => { supabase
      .channel("skill_requests_for_requester")
      .on(
        "postgres_changes",
                  { event: "INSERT", schema: "public", table: "skill_exchanges", filter: `requester_id=eq.${userId}` },
        (payload) => {
          const row = payload.new as any;
          setRequestStatusBySkill((prev) => ({ …prev, [row.skill_id]: { status: row.status, id: row.id
} })); },


      )
      .on(
        "postgres_changes",
                  { event: "UPDATE", schema: "public", table: "skill_exchanges", filter: `requester_id=eq.${userId}` },
        (payload) => {
```

```
          const row = payload.new as any;
          setRequestStatusBySkill((prev) => ({ …prev, [row.skill_id]: { status: row.status, id: row.id
} }));
          if (row.status === “accepted”) {
            toast({ title: “Connected!”, description: “You can now start chatting.” });
            // Refresh connected users when a new connection is made fetchConnectedUsers(userId);
          }
        },
      )
      .subscribe();
  };

  const filterSkills = () => {
    let filtered = skills;

    if (searchQuery) {
      filtered = filtered.filter(
        (skill) =>
          skill.title.toLowerCase().includes(searchQuery.toLowerCase()) ||
          skill.description.toLowerCase().includes(searchQuery.toLowerCase())
      );
    }
    if (categoryFilter !== “all”) {
      filtered = filtered.filter((skill) => skill.category === categoryFilter);
    }
```

```
  // Hide current user's own skills if (currentUserId) {
    filtered = filtered.filter((skill: any) => (skill as any).user_id !== currentUserId);
  }

  setFilteredSkills(filtered);
};

const categories = Array.from(new Set(skills.map((s) => s.category)));

return (
  <div className="min-h-screen bg-background"> <Navbar />
    <div className="container mx-auto px-4 py-8"> <div className="mb-8">
      <h1 className="text-4xl font-bold mb-2">Browse Skills</h1>
          <p className="text-muted-foreground">Discover amazing skills shared by our community</p>
    </div>
    {/* Filters */}
    <div className="flex flex-col md:flex-row gap-4 mb-8">
      <div className="relative flex-1">
          <Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-foreground h-4 w-4" />
        <Input
          placeholder="Search skills…"
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)} className="pl-10"
        />
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
  </div>

  <Select value={categoryFilter} onValueChange={setCategoryFilter}> <SelectTrigger className="w-full md:w-[200px]">
     <SelectValue placeholder="Category" />

   </SelectTrigger>

   <SelectContent>

     <SelectItem value="all">All Categories</SelectItem> {categories.map((category) => (
       <SelectItem key={category} value={category}>

         {category}

       </SelectItem>

     ))}

   </SelectContent>

  </Select>

</div>


{/* Skills Grid */}

{loading ? (

  <div className="text-center py-12">

    <p className="text-muted-foreground">Loading skills…</p> </div>

) : filteredSkills.length === 0 ? (

  <div className="text-center py-12">

    <p className="text-muted-foreground">No skills found. Try adjusting your filters.</p> </div>

) : (

  <div className="grid md:grid-cols-2 lg:grid-cols-3 gap-6"> {filteredSkills.map((skill) => (

      <Card key={skill.id} className="hover:shadow-lg transition-all duration-300 border-2 hover:border-primary/50">

       <CardHeader>
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
<div className="flex items-start justify-between mb-2"> <Badge variant="secondary">{skill.category}</Badge>
  <Badge variant="outline">{skill.skill_level}</Badge>

</div>

<CardTitle className="line-clamp-1">{skill.title}</CardTitle>

<CardDescription className="line-clamp-2">{skill.description}</CardDescription> </CardHeader>

<CardContent>

<div className="space-y-3">

  <div className="flex items-center gap-2 text-sm text-muted-foreground">

    <UserCircle className="h-4 w-4" />

    <span>{skill.profiles?.full_name}</span>

                    {currentUserId && connectedUsers.has((skill as any).user_id) && (

          <Badge variant="default" className="text-xs bg-green-100 text-green-800 hover:bg-green-100">

      Connected

    </Badge>

  )}

</div>


{skill.time_commitment && (

  <p className="text-sm text-muted-foreground"> {skill.time_commitment}

  </p>

)}


{skill.looking_for && skill.looking_for.length > 0 && ( <div>

    <p className="text-sm font-medium mb-1">Looking to learn:</p> <div className="flex flex-wrap gap-1">

      {skill.looking_for.map((item, idx) => (

        <Badge key={idx} variant="outline" className="text-xs"> {item}
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
        </Badge>
      ))}
    </div>
  </div>
)}

<Button
  className="w-full mt-4"
  variant={connectedUsers.has((skill as any).user_id) ? "secondary" : "default"} disabled={
    currentUserId === (skill as any).user_id || requestStatusBySkill[skill.id]?.status === "pending"
  }
  onClick={async () => {
    if (!currentUserId) {
      toast({
        title: "Sign in required",
        description: "Please sign in to connect with users.",
        variant: "destructive",
      });
      return;
    }
    if (currentUserId === (skill as any).user_id) {
      toast({
        title: "Action not allowed",
        description: "You cannot connect to your own skill.",
      });
```

```
  return;
 }
                                    if (requestStatusBySkill[skill.id]?.status === “accepted” &&
requestStatusBySkill[skill.id]?.id) {
  navigate(`/chat?exchangeId=${requestStatusBySkill[skill.id]!.id}`); return;
 }
 const { data, error } = await supabase
  .from(“skill_exchanges”)
  .insert({
   requester_id: currentUserId,
   skill_id: skill.id,
   status: “pending”,
  } as any)
  .select(“id,skill_id,status”)
                                                .single();
                                               if (error) {
                                                 toast({
   title: “Error”,
   description: “Could not send connect request. Please try again.”,
   variant: “destructive”,
  });
 } else {
  if (data) {
        setRequestStatusBySkill((prev) => ({ …prev, [skill.id]: { status: (data as
any).status, id: (data as any).id } }));
  }
```

```
                toast({
                  title: “Request sent”,
                  description: “The user has been notified of your interest.”, });
                }
              }}
            >
              {currentUserId === (skill as any).user_id
                ? “Your Skill”
                : connectedUsers.has((skill as any).user_id) ? “Chat (Connected)”
                : requestStatusBySkill[skill.id]?.status === “accepted” ? “Chat”
                : requestStatusBySkill[skill.id]?.status === “pending” ? “Pending”
                : “Connect”}
            </Button>
          </div>
        </CardContent>
      </Card>
    ))}
  </div>
 )}
 </div>
 </div>
 );
};
export default Browse;
```

F. Result

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

The result of the Skill Swap – A Collaborative Learning Platform has been a significant improvement in how individuals connect, share knowledge and learn from one another. The system has successfully created an interactive and community-driven environment that encourages peer-to-peer learning and collaboration. Key outcomes include enhanced user engagement, efficient skill matching and real-time communication between learners and instructors through the integrated chat system.

The automation of the skill matching and communication processes has streamlined the user experience, allowing participants to easily find suitable learning partners without manual searching.

This approach reduces time, improves accuracy and fosters active collaboration among users. The centralized database ensures secure data handling, real-time updates and reliable storage of user profiles, skill listings and feedback records.

The App Flow: After registering and creating their profile, users list the skills they can teach and the ones they wish to learn. The system automatically finds suitable matches through the smart matching algorithm and displays them on the dashboard. Once connected, users can start real-time chat sessions to exchange knowledge and skills. After completion, they can provide feedback and ratings to maintain quality and trust within the platform. The system ensures transparency, security, and efficiency throughout the entire skill exchange process.

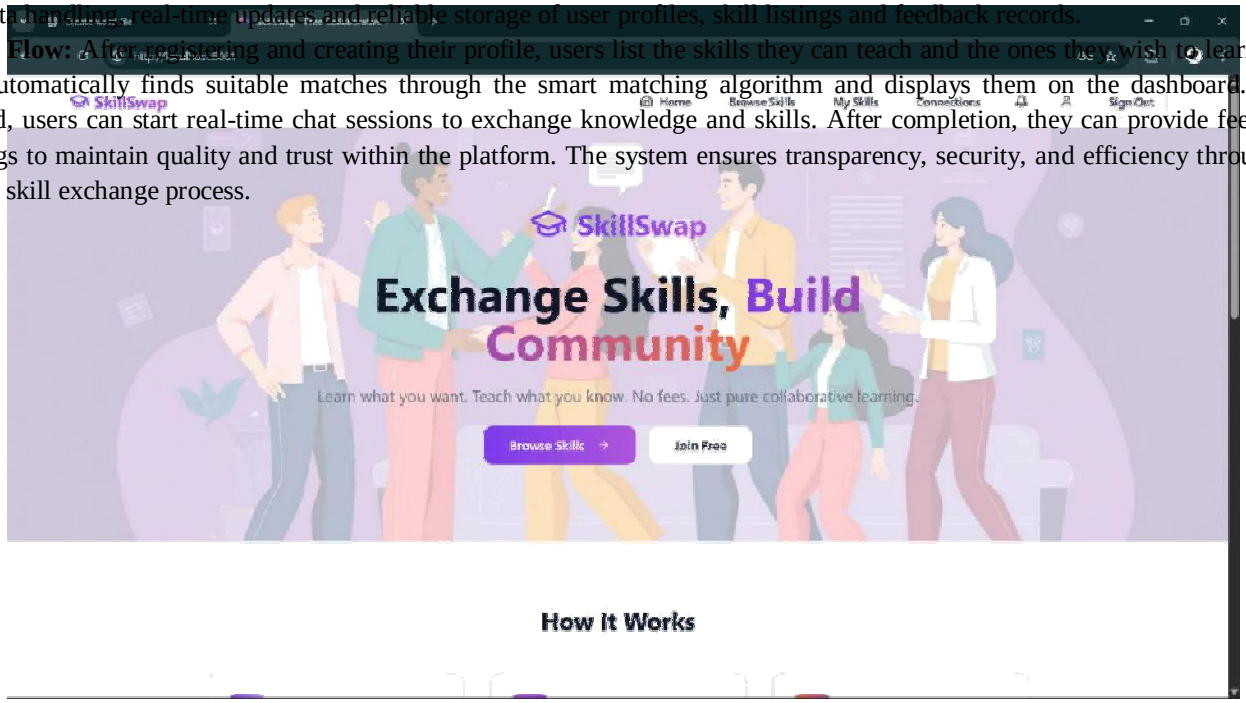


Figure 2: Home Page

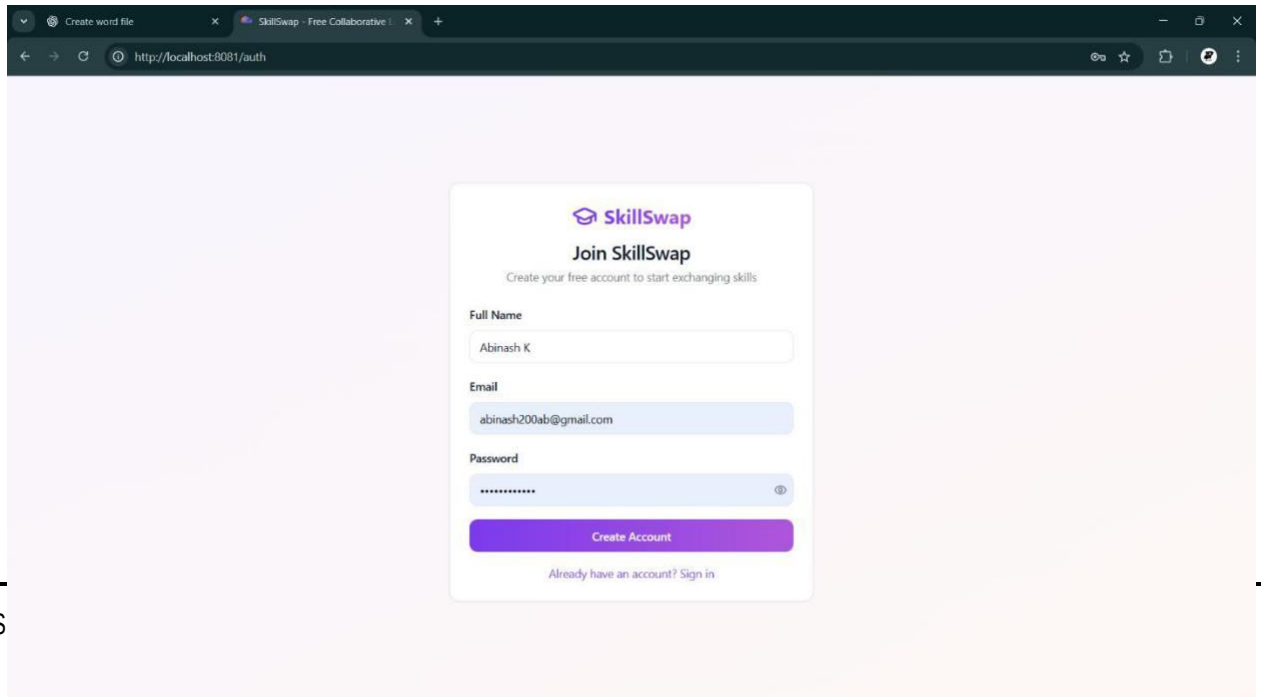


Figure 3: Sign Up Page

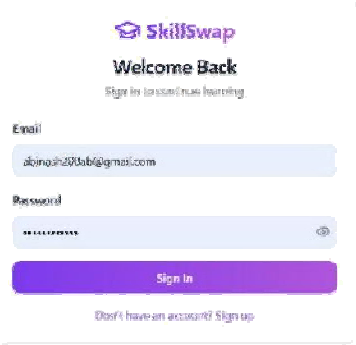
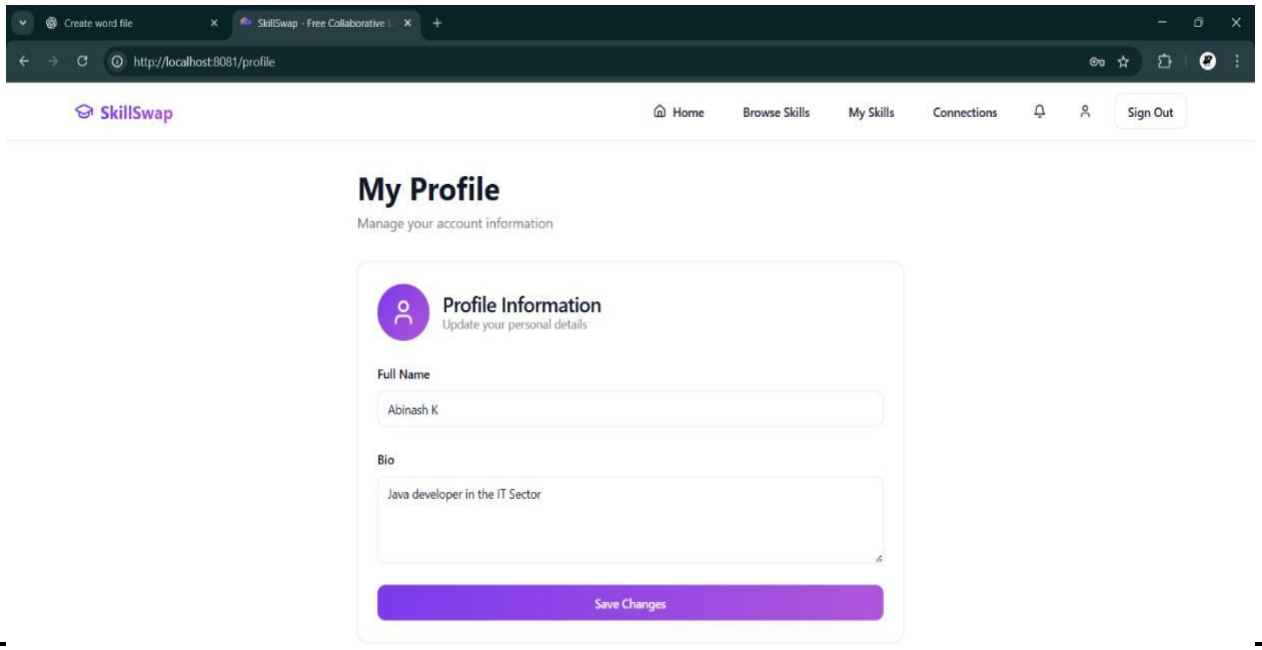


Figure 4: Sign in Page



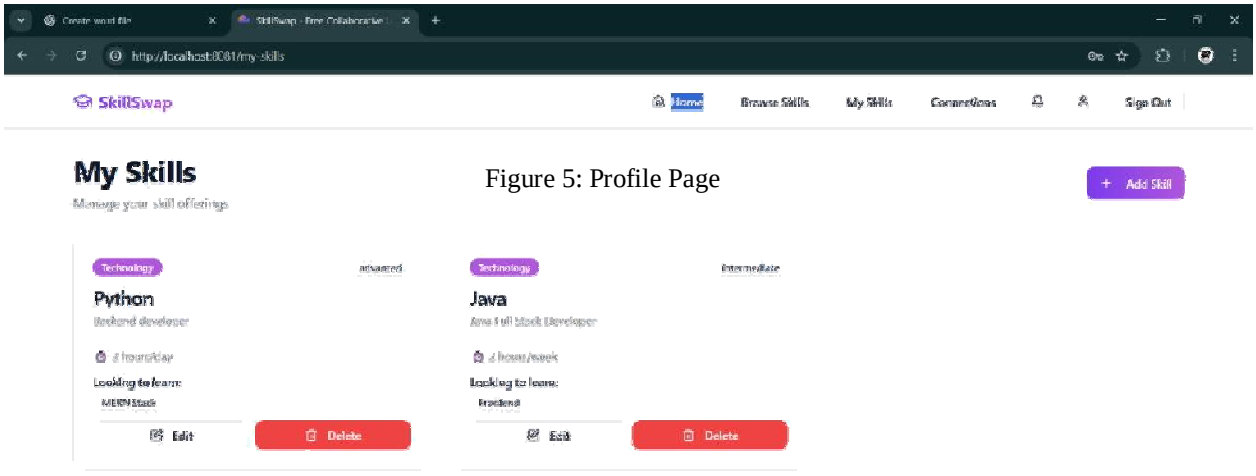
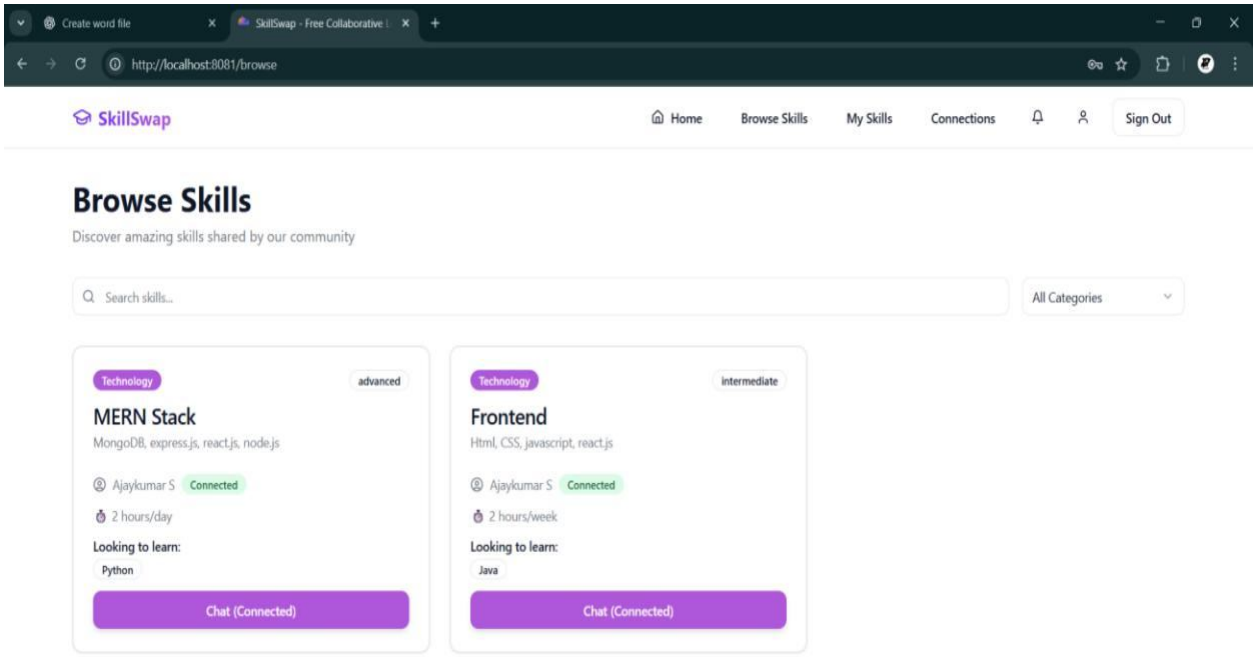


Figure 5: Profile Page

Figure 5: My Skills Page



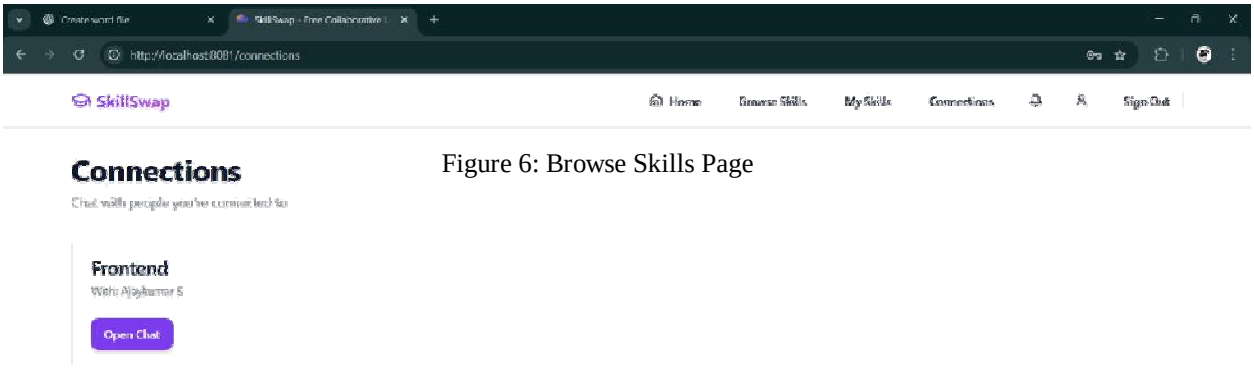


Figure 6: Browse Skills Page

Figure 7: Connections Page

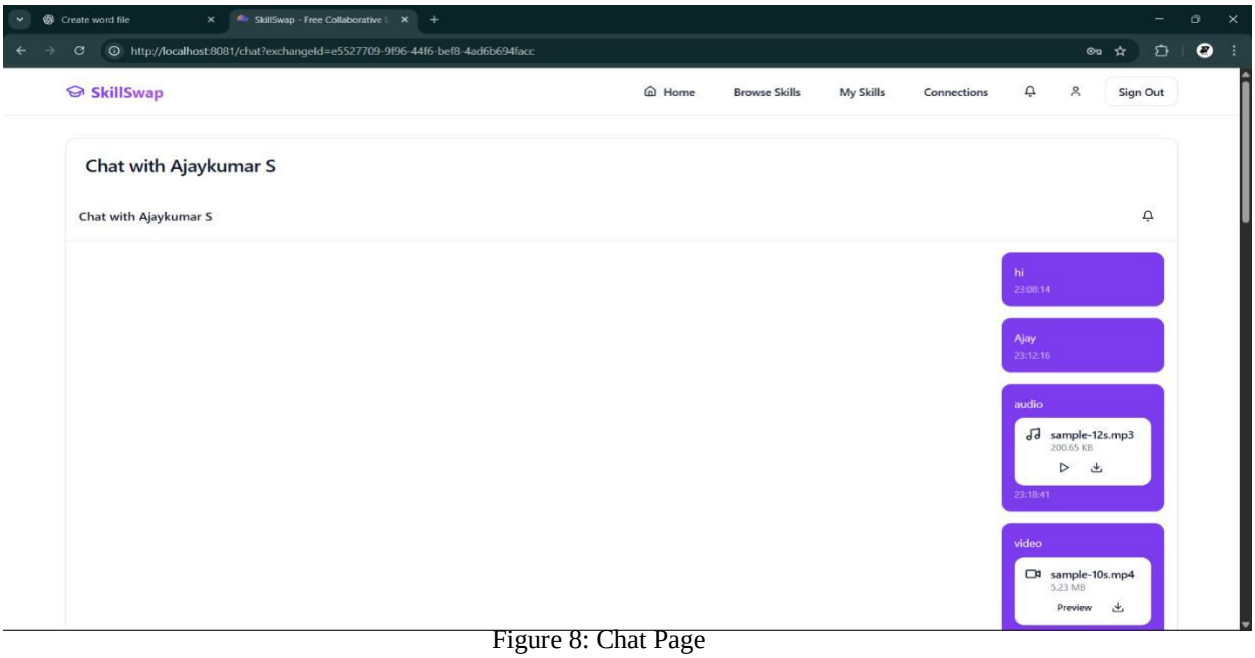


Figure 8: Chat Page

VI. CONCLUSION AND FUTURE ENHANCEMENT

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

A. Conclusion

In conclusion, the Skill Swap – A Collaborative Learning Platform provides an innovative and efficient solution for promoting peer-to-peer learning and knowledge sharing in a digital environment. By enabling users to both teach and learn skills, the system bridges the gap between learners and mentors, fostering a culture of collaboration and continuous personal development. The platform's user-friendly interface, coupled with smart matching and real-time communication, ensures a seamless and interactive experience for all participants.

The system's core features — including secure user authentication, intelligent skill matching, real-time chat functionality, and feedback management — work together to create a trusted, transparent, and engaging learning community. Built using modern technologies such as Node.js, Express.js, Bootstrap, TypeScript, and Supabase, the application guarantees scalability, reliability, and strong data security. Overall, the Skill Swap platform redefines the way individuals connect and learn by offering a flexible, cost-free, and community-driven approach to skill development. It serves as a foundation for building global learning networks that encourage mutual growth and digital empowerment.

B. Future Scope

1) Video and Voice Integration: Future versions can include video and voice calling features, allowing users to conduct live skill-sharing sessions for a more interactive learning experience.
2) AI-Based Skill Recommendation: Implementing artificial intelligence can help analyze user activity and preferences to automatically recommend relevant skills or potential learning partners.
3) Mobile Application: A dedicated mobile app for Android and iOS can be developed to allow users to manage profiles, chat, and learn on the go, improving accessibility and engagement.
4) Gamification Features: Introducing badges, rewards, or progress tracking can motivate users to stay active, share more skills, and improve learning consistency.
5) Advanced Analytics Dashboard: The admin dashboard can be enhanced with advanced analytics to monitor user engagement, popular skills, and learning trends, helping administrators optimize the platform for continuous growth.

APPENDICES

SOURCE CODE:

Chat.tsx:

```
import { useEffect, useMemo, useState } from "react"; import { useLocation } from "react-router-dom"; import Navbar from "@/components/Navbar";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"; import { useToast } from "@/hooks/use-toast";

import { supabase } from "@/integrations/supabase/client"; import ChatInterface from "@/components/ChatInterface";


const Chat = () => {

  const location = useLocation();

  const { toast } = useToast();

  const [displayTitle, setDisplayTitle] = useState<string>(""); const [loading, setLoading] = useState<boolean>(true);
  const [currentUserId, setCurrentUserId] = useState<string>("");

  const exchangeId = useMemo(() => new URLSearchParams(location.search).get("exchangeId"), [location.search]);
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
// Get current user
const getCurrentUser = async () => {
  const { data: { user } } = await supabase.auth.getUser(); if (user) {
    setCurrentUserId(user.id);
  }
};


useEffect(() => {
  const load = async () => {
    if (!exchangeId) {
      toast({ title: "No exchange selected", description: "Open chat from a connected request.",
variant: "destructive" });
      setLoading(false);
      return;
    }
    const [{ data, error }] = await Promise.all([
      supabase
      .from("skill_exchanges")
      .select(`
        id,
        status,
        requester_id,
        skills (
          title,
          looking_for,
```



```
      user_id,
      profiles!skills_user_id_fkey (full_name)
    )
  `)
  .eq("id", exchangeId)
  .single(),
]);
if (error) {
  toast({ title: "Error", description: "Failed to load chat.", variant: "destructive" });
} else {
  // Get current user to determine who we're chatting with const { data: { user } } = await supabase.auth.getUser(); const exchangeData = data as any;
  const skills = exchangeData?.skills;

  if (skills && user) {
    // Determine who we're chatting with based on current user let chatPartnerName = "";

    if (user.id === exchangeData.requester_id) {
      // Current user is the requester, so we're chatting with the skill owner chatPartnerName = skills.profiles?.full_name ||
      "Unknown User";
    } else if (user.id === skills.user_id) {
      // Current user is the skill owner, so we're chatting with the requester
      // We need to fetch the requester's profile
      const { data: requesterProfile } = await supabase
        .from("profiles")
        .select("full_name")
        .eq("id", exchangeData.requester_id)
        .single();
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
            chatPartnerName = requesterProfile?.full_name || "Unknown User";

          }



          setDisplayTitle(chatPartnerName);

        } else { setDisplayTitle("");
        }

      }



     // Get current user await getCurrentUser(); setLoading(false);
    };

    load();

  }, [exchangeId, toast]);



  return (

    <div className="h-screen bg-background flex flex-col"> <Navbar />

      <div className="flex-1 container mx-auto px-4 py-8"> <Card className="h-full flex flex-col">

          <CardHeader>

                <CardTitle>{displayTitle ? `Chat with ${displayTitle}` : `Chat ${exchangeId ?
`#${exchangeId}` : ""}`}</CardTitle>

          </CardHeader>

          <CardContent className="flex-1 flex flex-col p-0"> {loading ? (

             <div className="flex items-center justify-center h-full">

               <div className="text-muted-foreground">Loading chat...</div> </div>

           ) : exchangeId && currentUserId && displayTitle ? (

             <ChatInterface

               exchangeId={exchangeId}
```

```
            currentUserId={currentUserId}

            chatPartnerName={displayTitle}

                                                                                    />
                                                                                  ) : (
          <div className="flex items-center justify-center h-full"> <div className="text-muted-foreground text-center">
            <p>No exchange selected or user not found.</p>

            <p className="text-sm mt-2">Please open chat from a connected request.</p> </div>

          </div>

        )}

       </CardContent>

     </Card>

   </div>

 </div>

 );

};

export default Chat;
```

VII. ACKNOWLEDGEMENT

It is one of the most gratifying tasks in life to find the right words to express our sincere gratitude to those who have supported and guided us. We are deeply thankful to Almighty God for His blessings and guidance throughout the completion of our Skill Swap project, which has helped us learn, grow and achieve what we are today.

We are grateful to our beloved Principal Dr. R. RADHAKRISHNAN, M.E., Ph.D., Adhiyamaan College of Engineering (An Autonomous Institution), Hosur for providing the opportunity to do this work in premises.

We acknowledge our heartful gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur and the supervisor for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

We are highly indebted to Mrs. M. MALATHI, M.E., Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, whose immense support encouragement and valuable guidance were responsible to complete the project successfully.

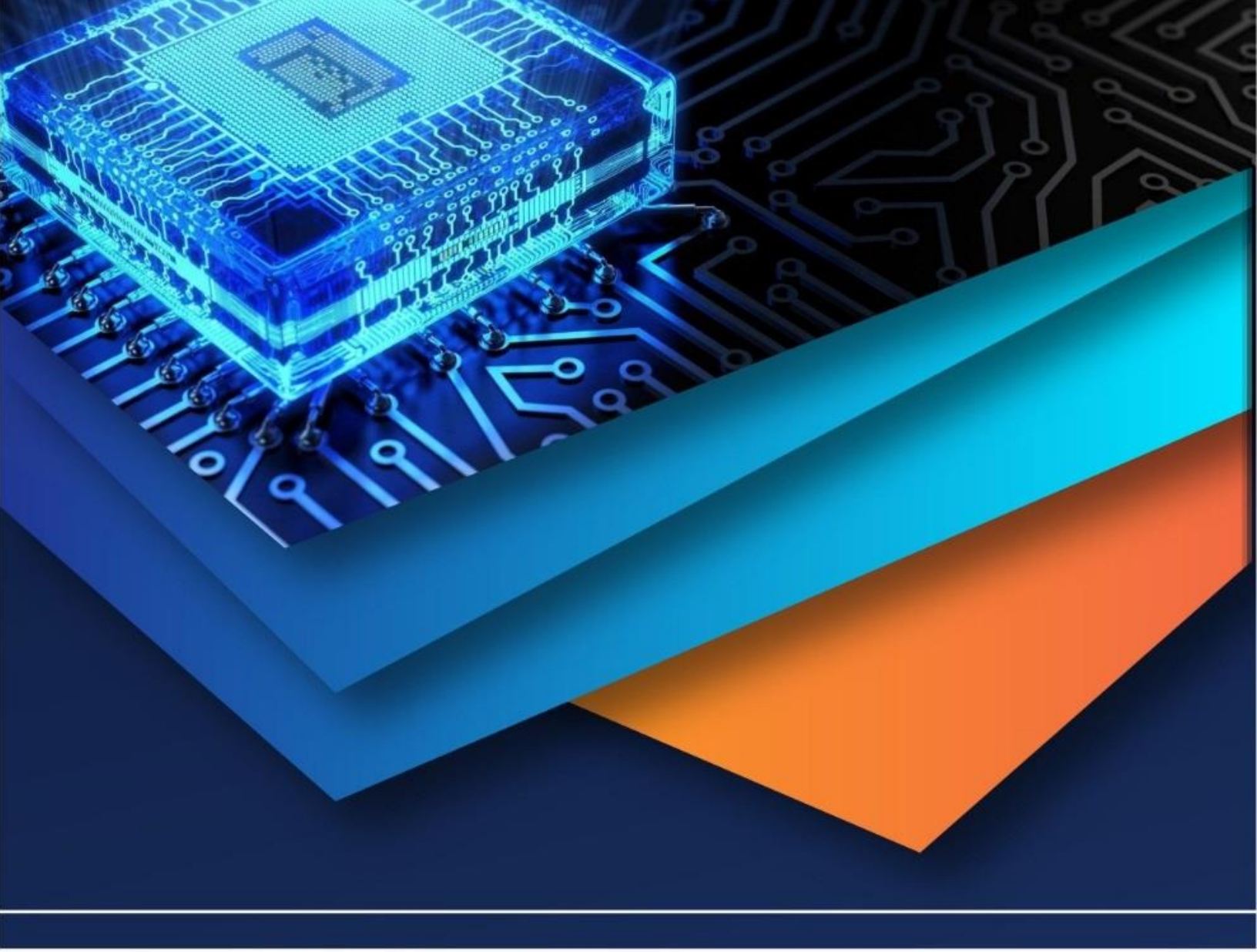
We also extent our thanks to Project Coordinator and all Staff Members for their support in complete this project successfully.

Finally, we would like to thank to our parents, without their motivational and support would not have been possible for us to complete this project successfully.

REFERENCES

[1] Skillshare: https://www.skillshare.com/ – Online learning platform that inspired many skill-exchange concepts.
[2] Swapaskill (Prototype idea): https://swapaskill.com/ – Community-based skill-swapping idea.
[3] Simbi: https://simbi.com/ – A marketplace for trading skills without money.
[4] LinkedIn Learning. (2025). Professional learning and development platform. Retrieved from https://www.linkedin.com/learning
[5] Coursera. (2025). Global online learning platform. Retrieved from https://www.coursera.org/

[6] Sharma, P. & Patel, R. (2021). Peer-to-Peer Learning through Digital Skill Exchange Platforms. International Journal of Computer Applications.
[7] Lee, J. (2020). Web-Based Platforms for Collaborative Skill Sharing and Community Development. Journal of Information Technology and Society.
[8] Gupta, S., & Kaur, D. (2022). Enhancing Employability through Online Skill Exchange Networks. IEEE Access, 10, 55412–55423.
[9] Brown, T., & Adams, M. (2019). The Growth of Skill-Sharing Economies: A Study of Peer Learning Models. Education and Information Technologies Journal
[10] Kumar, A., & Singh, P. (2023). Design and Development of Web-Based Collaborative Platforms Using MERN Stack. International Journal of Web Engineering.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY