Online Skill Exchange Website

SUBMITTED BY

Mhammad Raheel

2021-ag-7979

ADVISED BY

Ms. Sidra Shahid

A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES

UNIVERSITY OF AGRICULTURE, FAISALABAD

2025

DECLARATION

I hereby declare that the contents of the report **"Online Skill Exchange Website"** are project of my own research and no part has been copied from any published source (except the references). I further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default the scholar will be proceeded against as per UAF policy.

_____

M. Raheel

2021-ag-7979

CERTIFICATE

To,
The Controller of Examinations,
University of Agriculture,
Faisalabad.

The supervisory committee certify that **Muhammad Raheel, 2021-ag-7979** has successfully completed his project in partial fulfillment of requirement for the degree of BS Information Technology under our guidance and supervision.

_____

Ms Sidra Shahid
Supervisor/Lecturer

_____

Dr. Muhammad Ahsan Latif
Chairman,
Department of Computer Science

ACKNOWLEDGEMENT

I would like to express my gratitude to everyone who has contributed in the completion of this report. Firstly, I thank to **ALLAH ALMIGHTY**, most magnificent and most merciful, for all his blessings. Then I am so grateful to the Department of Computer Science for making it possible for me to study here. My special and heartily thanks goes to my supervisor, **Ms Sidra Shahid** who encouraged and directed me. His support and challenges brought this work towards a completion. In his supervision that this work came into existence. For any faults I take full responsibility. I am also deeply thankful to my informants. I want to acknowledge and appreciate their help and transparency during my research. I am also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, I also thank my family who encouraged me and prayed for me throughout the time of my research. May the **ALLAH ALMIGHTY** richly bless all of you.

ABSTRACT

The *SkillSwap – Skill Exchange Platform* is an innovative web-based application designed to empower individuals to trade their skills directly without involving money. By connecting users with complementary skillsets, the platform enables a mutual exchange that benefits both parties—such as a web developer offering coding help in return for graphic design services. This approach redefines the way freelancers, students, and skilled individuals collaborate, especially in low-budget or barter-driven environments.

SkillSwap was developed using a full-stack JavaScript architecture. The frontend is built with React.js and Vite, styled using Tailwind CSS for responsive design. On the backend, Node.js and Express.js manage API interactions, while MongoDB Atlas serves as the cloud-based database. The authentication process is secured through Google OAuth 2.0 integration and protected using JWT (JSON Web Tokens). Cloudinary is employed to handle user image uploads securely and efficiently. Docker and Docker Compose were used to containerize the backend and frontend services, simplifying deployment and scalability. Environment variables are managed through .env files to ensure modularity and security.

Users begin by registering or signing in using their Google account. They then fill out a dynamic profile setup form where they specify their offered and desired skills. The platform intelligently matches users based on overlapping skill needs using a simple filtering logic. Contact information is exchanged directly, allowing users to communicate off-platform. Although no direct chat or wallet system is integrated, the system emphasizes privacy and transparency by keeping communications external.

This report details the entire system lifecycle using the Agile development process, with emphasis on modules such as user registration, skill matching, image uploads, and protected routes. UML and system architecture diagrams are provided to illustrate the workflow, and a comprehensive testing section validates system performance. The application is hosted locally for demonstration purposes, with the ability to scale using cloud services like Railway or Render.

SkillSwap aims to create a vibrant, community-driven network of learners, professionals, and creators. It is particularly well-suited for educational communities, student freelancer groups, or underfunded startups looking to collaborate without cash exchange. With its focus on skill diversity, user empowerment, and tech-forward implementation, the platform is a step toward more inclusive digital collaboration models.

Catalog

Chapter 1

INTRODUCTION

1.1 Background

In the modern digital economy, the importance of practical, specialized skills has become more prominent than ever. From web development and video editing to digital marketing and language translation, these skills form the foundation of countless online and offline opportunities. However, not everyone has equal access to professional services or financial resources that allow them to pay for learning or hiring skilled individuals. This is especially true for students, freelancers in developing regions, or passionate individuals who are self-taught and eager to learn or grow through mutual help. While educational platforms and freelancing websites do exist, they are often either too expensive, overly competitive, or too formalized to accommodate grassroots collaboration or skill-sharing without monetary transactions.

There exists a vast population of people who possess valuable abilities but are unable to fully utilize them because they cannot afford to outsource tasks they need done. This scenario creates an imbalance—people have something to offer but cannot benefit from the community because of cost barriers. The traditional freelancing model also suffers from other limitations: poor visibility for beginners, high commissions on platforms, low trust in unverified services, and very little room for non-commercial collaboration. In contrast, bartering or skill exchange—an old concept rooted in mutual help—offers a powerful alternative that remains largely untapped in the digital age.

The SkillSwap platform is a web-based system designed specifically to address this gap. By enabling users to register their own skills and state what services they are looking for in return, the platform creates a structured, searchable database of potential exchanges. Whether a graphic designer needs web development support or a writer wants video editing in return for proofreading, SkillSwap makes such collaborations possible in a secure, transparent, and easy-to-use format. It offers a practical solution for individuals who are rich in skills but poor in budget.

Technically, SkillSwap is built using modern full-stack JavaScript technologies. It combines a fast React + Vite frontend with a Node.js + Express.js backend, connected to a MongoDB Atlas database. For user identity verification, Google OAuth 2.0 is integrated with JWT-based session management. Cloudinary is used to host user profile pictures, and the entire app is Dockerized for easy deployment. Skill matching logic is implemented using basic filtering mechanisms that compare offered skills with needed skills. The result is a lightweight, scalable solution that empowers everyday people to unlock growth through collaboration—not competition.

1.2 Description

The **SkillSwap – Skill Exchange Platform** is a full-featured, containerized web application designed to enable individuals to trade skills directly with each other in a decentralized and non-monetary environment. It serves as a digital meeting point for users who wish to exchange knowledge, services, or expertise in various domains such as technology, arts, education, and business without requiring payment or subscription-based access.

The platform is designed around a simple but impactful concept: users create profiles listing the skills they offer and the skills they are looking to learn or receive. These profiles are stored in a secure cloud database (MongoDB Atlas) and include attributes such as personal information, professional interests, availability, profile images, and skill tags. A built-in matching system intelligently suggests potential partners whose offered skills align with another user's needs—helping to initiate relevant and high-value connections.

To maintain security and simplicity, the platform uses **Google OAuth 2.0** for user authentication. This not only reduces friction during sign-up but also ensures that user identities are verified through a trusted third party. Once logged in, users receive a **JWT-based session token**, which is used to maintain authentication throughout their activity. This combination of OAuth and JWT makes the login flow both seamless and secure.

From a frontend perspective, SkillSwap leverages the speed and modularity of **React.js** and **Vite**. These technologies allow the application to deliver a fast, responsive experience across various devices. Tailwind CSS is used to maintain consistent design patterns and responsive layouts, ensuring a clean and mobile-friendly interface.

The backend is built using **Node.js** and **Express.js**, providing a reliable and scalable API architecture. The server manages route handling, middleware logic, and API endpoints for user, profile, and skill operations. For media storage, such as user profile images or sample work, the platform integrates with **Cloudinary**, ensuring that image uploads are efficiently handled and securely hosted.

The entire application is containerized using **Docker**, allowing it to be deployed across different environments with minimal configuration. A `docker-compose.yml` file defines both the backend and frontend services for local development or cloud hosting using services like **Render** or **Railway**.

SkillSwap also includes features such as skill filtering, location tagging, contact information display (e.g., email or social handle), and real-time updates. While an admin interface is optional, the system architecture is extensible enough to support administrative controls for monitoring users or moderating content.

Through its design, SkillSwap not only facilitates a more inclusive form of digital learning and freelancing but also promotes community growth by encouraging users to engage with others who share similar passions or complementary goals. It is an

application built for those who value collaboration over competition and shared growth over financial gain.

1.2 Problem Statement

Despite the abundance of skilled individuals across the globe, there remains a significant gap between those who can offer services and those who require them—particularly when money is not available as a medium of exchange. Existing platforms for freelancing and online collaboration predominantly operate on commercial models, often charging high commissions, requiring upfront payments, or limiting visibility for newcomers. This creates a highly competitive and economically exclusive environment that discourages beginners and underrepresented talent.

Moreover, while informal bartering communities exist on social media or messaging apps, these are largely unstructured, lack trust mechanisms, and provide no assurance of mutual benefit. Users often hesitate to exchange services due to privacy risks, absence of clear skill matching, or uncertainty about the credibility of other participants. Additionally, there is no central digital system that allows for controlled skill-to-skill exchanges, secure identity verification, or tracking of skill demand in real time.

Another major challenge lies in **accessibility and usability**. Many platforms are bloated with unnecessary features or complex payment systems, making them difficult to use for students or people with basic technical literacy. Most require manual handling of communication, verification, and agreements, increasing friction in the exchange process. Additionally, privacy concerns arise when platforms request sensitive data for login or messaging without offering proper encryption or user control.

The SkillSwap platform addresses these limitations by introducing a simple, secure, and user-friendly solution for skill exchange. It focuses on four core challenges

Lack of cost-free collaboration options for people with skills but no money.

Absence of intelligent skill-matching logic in informal skill-sharing groups.

No authentication-backed identity layer in barter communities.

Inaccessible deployment or hosting options for low-budget projects.

SkillSwap eliminates financial dependency by allowing users to register their skills and needs, then match with others who can offer what they are looking for. With Google OAuth 2.0 for verified identity, JWT for secure sessions, and MongoDB for data storage, the system creates a robust foundation for decentralized collaboration.

Its privacy-first approach and Docker-based deployment further support lightweight, scalable usage—especially in educational, freelance, or early startup environments.

1.4 Scope

The SkillSwap platform is designed to provide an inclusive, skill-based exchange system where users can offer and request services without involving money. Its primary focus is to build a digital ecosystem that promotes equal access to learning and collaboration, especially for those with limited financial resources.

From a technological perspective, the platform covers the entire lifecycle of user registration, profile creation, skill matching, contact sharing, and image hosting. It uses a frontend built with React.js, styled using Tailwind CSS, and powered by Vite for fast development and loading performance. The backend is managed through Node.js and Express.js, connected to a MongoDB Atlas database that stores user profiles and skill data. Authentication is handled via Google OAuth 2.0, and session control is managed using secure JWT tokens. Cloudinary is integrated to support profile picture uploads and media hosting. Docker is used to containerize the frontend and backend, enabling smooth deployment on Render, Railway, or local environments.

The platform enables users to sign up using their Google accounts, set up their profiles with offered and needed skills, and browse through a growing network of other users. It supports intelligent filtering and skill-matching features, allowing users to find others whose skills align with their own needs. Users can also view public contact information to reach out and arrange the exchange independently.

In terms of functionality, the platform is suitable for use in educational institutions, student communities, small businesses, or freelance networks where collaboration and self-learning are priorities. Its modular structure also allows for future upgrades, such as adding messaging features, calendar scheduling, or group-based collaborations.

There are some limitations in scope as well. Currently, the system does not include an in-app messaging system or transaction tracking. It relies on users connecting through external means using the provided contact information. Additionally, users must have a valid Google account to register, and image uploads are limited to formats supported by Cloudinary.

Despite these boundaries, SkillSwap lays a strong foundation for a practical and impactful solution to the lack of financial-free collaboration tools. It supports individual empowerment, community building, and scalable, technology-driven learning models.

Thanks for showing the screenshot — that helps a lot.

The issue is due to Microsoft Word misinterpreting the formatting of numbered lists when copied from web-based environments (like ChatGPT). Here's how I'll now format all future lists so they **won't auto-continue numbering incorrectly** and will paste exactly as shown:

Instead of using auto-numbering like:

```
1.
2.
3.
```

I will **manually write the numbers** like this:

```
(1) Enable non-monetary skill exchange: ...
(2) Support secure user authentication: ...
(3) Provide a modern and responsive frontend: ...
```

This method ensures Word does not treat it as an active list — **no extra lines, no continued numbering.**

Here's the **corrected version** of the Objectives section:

1.4 Project Scope

The SkillSwap platform has been designed with a clearly defined scope that encompasses its technological components, functional boundaries, and targeted user experience. The system is developed as a lightweight, containerized web application that focuses exclusively on skill-based profile matching and collaboration. Unlike traditional freelancing or hiring platforms, it deliberately avoids integrating direct messaging, financial transactions, or job bidding mechanisms. This focused scope ensures simplicity, scalability, and accessibility for a broader user base.

From a technical standpoint, the platform uses a full-stack JavaScript architecture, with React.js and Vite powering the frontend and Node.js with Express.js handling backend operations. All user data, including skills, profiles, and contact details, are securely stored in a cloud-hosted MongoDB Atlas database. Authentication is facilitated using Google OAuth 2.0, and user sessions are protected with JWT tokens to ensure secure, tokenized access to the platform. Profile images are uploaded through Cloudinary, which provides optimized cloud storage and seamless frontend integration.

The scope also includes a frontend user interface that allows new users to register and log in through Google, create and edit a skill-based profile, and search or browse other users based on required and offered skills. When mutual skill matches are found, users can view contact details and arrange a collaboration externally. This model keeps the platform lightweight and reduces complexity while maintaining high utility.

Administrative access is optional and can be used to review users, check for misuse, or manage system content if needed.

Functionally, the system restricts all operations to skill input, skill matching, contact visibility, and profile updates. There is no internal chat system or real-time collaboration tool, which aligns with the platform's intended use case of decentralized, low-cost knowledge exchange. The application is designed to be deployed in both development and production environments using Docker and Docker Compose, which further broadens its scope for testing, educational demonstrations, and scalable cloud-based deployments on services such as Render or Railway.

In terms of target audience, the platform is tailored for students, freelancers, and self-taught individuals who wish to engage in professional collaborations without monetary commitments. It is ideal for communities or institutions seeking a bartering-based platform where knowledge and skills are the currency. By limiting its features to essential functions and using secure cloud-based technologies, SkillSwap remains both approachable and practical for low-resource settings, while retaining the potential to grow into a more expansive system in the future.

1.5 Objectives

The primary objective of the **SkillSwap – Skill Exchange Platform** is to create a secure, scalable, and user-friendly environment where individuals can exchange skills without financial involvement. This fosters mutual growth, community learning, and professional networking.

The specific objectives of the system are as follows:

(1) **Enable non-monetary skill exchange:**
Allow users to offer and request services without requiring payments, promoting collaboration through value-based barter.

(2) **Support secure user authentication:**
Integrate Google OAuth 2.0 for verified login and use JWT to manage secure session tokens.

(3) **Provide a modern and responsive frontend:**
Use React.js, Vite, and Tailwind CSS to build a fast, mobile-friendly user interface.

(4) **Develop a scalable backend and database:**
Build the backend using Node.js and Express.js, and store user profiles, skills, and matching data using MongoDB Atlas.

(5) **Facilitate intelligent skill matching:**
Implement a filtering and suggestion system that matches users based on offered and needed skills.

(6) **Allow contact sharing and profile viewing:**
Let users view essential contact information (such as email or social handles) to coordinate exchanges independently.

(7) **Enable media upload and hosting:**
Use Cloudinary to handle profile image uploads and ensure reliable media access.

(8) **Ensure easy deployment and portability:**
Containerize both backend and frontend using Docker, and support deployment via Render, Railway, or local environments.

(9) **Promote inclusivity and accessibility:**
Design the platform so that users from different educational or financial backgrounds can participate equally in skill exchange.

(10) **Create a modular, extensible system:**
Ensure that the platform is easy to upgrade with future features like messaging, group projects, or admin controls.

These objectives serve as the foundation of the SkillSwap platform and guide the development and deployment of a sustainable, community-driven application.

1.6 Feasibility

Before initiating full-scale development of the SkillSwap platform, a comprehensive feasibility analysis was conducted to assess its practicality from multiple dimensions. The purpose of this analysis is to confirm that the system is achievable using current technologies, fits within realistic time and budget constraints, and can operate efficiently in a real-world environment.

1.6.1 Technical Feasibility

The SkillSwap platform is technically feasible using widely adopted open-source technologies. It uses React.js and Vite for the frontend, which are both efficient for building fast and modular single-page applications. The backend is developed in Node.js with Express.js, which is well-suited for building RESTful APIs and supports easy integration with external services like Cloudinary and Google OAuth.

The database system, MongoDB Atlas, offers a cloud-based, scalable document model that is ideal for storing user profiles and dynamic skill sets. Docker is used to containerize both backend and frontend services, ensuring consistency across development and production environments. These components work together without the need for complex server setups, making the system robust and cost-effective.

1.6.2 Schedule Feasibility

The project was planned and executed using Agile methodology, allowing for iterative development in weekly sprints. The estimated project duration was 16 to 18 weeks, divided into the following phases:

– Requirement gathering and planning (2 weeks)
– Backend and API development (4 weeks)
– Frontend integration and UI/UX design (4 weeks)
– Skill matching logic and cloud integration (3 weeks)
– Testing, debugging, and optimization (2–3 weeks)
– Dockerization and deployment (1–2 weeks)

Each sprint allowed for frequent reviews and adjustments, making the project timeline both flexible and achievable.

1.6.3 Economic Feasibility

The system is highly economical to build and deploy due to the use of free or freemium tools and services. MongoDB Atlas, Render, and Railway offer free-tier deployment options, and Docker simplifies local development without the need for paid hosting. Google OAuth, Cloudinary, and all essential frameworks used (React, Node, Express) are open-source and license-free.

Because no financial transaction features or premium subscriptions are required, ongoing maintenance costs are minimal. SkillSwap is therefore a low-cost, high-impact solution that is economically sustainable.

1.6.4 Cultural Feasibility

The platform supports skill exchange in a culturally diverse and inclusive manner. By removing financial transactions, it allows users from different economic backgrounds to benefit equally. The interface is simple and can be easily localized in future iterations. The idea of exchanging services rather than money aligns well with cooperative learning models and peer-based collaboration.

As digital collaboration continues to grow in educational and professional spaces, SkillSwap fits naturally into the evolving habits of online users.

1.6.5 Legal and Ethical Feasibility

All technologies used are compliant with open-source licenses. User data is securely handled—OAuth is used for authentication, and JWT ensures safe session handling. No sensitive personal data is stored beyond what is visible on a public profile, and images are hosted using Cloudinary’s secure API. There are no features that violate data protection or privacy standards. The system promotes fairness and avoids exploitation by ensuring that both users benefit equally from any exchange.

1.6.6 Operational Feasibility

The platform requires minimal operational overhead. Admin access is optional, and most functions (like profile setup, skill matching, and contact sharing) are handled directly through the frontend. Users can complete registration and skill updates without any assistance, making the system self-service. Docker ensures the platform can be deployed and maintained on any compatible infrastructure with ease.

In conclusion, the SkillSwap platform is fully feasible from technical, operational, economic, cultural, legal, and time-based perspectives. It leverages reliable technologies and promotes a practical, sustainable model for collaborative growth.

1.7 Requirements

The development of the **SkillSwap – Skill Exchange Platform** is based on a clear set of functional and non-functional requirements. These define how the system behaves, what features it supports, and the technical environment needed for its proper operation.

1.7.1 Functional Requirements

(1) User Registration and Login:
The system must allow users to register and log in using Google OAuth 2.0 for authentication.

(2) JWT-Based Session Management:
After successful login, the system must issue a JWT token to maintain the session securely.

(3) Profile Creation and Update:
Users must be able to create and edit their profiles, including name, bio, contact info, skills they offer, and skills they want to learn.

(4) Skill Matching and Search:
The system should allow users to filter and search for other users based on skill categories, names, or matching interests.

(5) Contact Information Sharing:
Each profile should display contact information like email or social links to allow direct communication between matched users.

(6) Image Uploading:
Users must be able to upload profile images, which are hosted via Cloudinary.

(7) Skill Suggestions:
The platform should recommend compatible users based on mutual skills for offer and need.

(8) Logout and Session Expiry:
Users must be able to log out manually, and JWT tokens must expire after a defined period.

1.7.2 Non-Functional Requirements

(1) Performance:
The platform must respond to all user interactions within 2 seconds under average load.

(2) Scalability:
The system should support a growing user base without any significant drop in performance.

(3) Security:
All communication must occur over HTTPS, and sensitive operations must be protected using JWT, OAuth, and secure API calls.

(4) Usability:
The interface should be simple, responsive, and intuitive, requiring minimal technical knowledge to navigate.

(5) Portability:
The project must be containerized using Docker to ensure easy setup and portability across systems.

(6) Availability:
The system must maintain uptime during core usage hours and be easily restartable in case of deployment updates.

(7) Maintainability:
The code should follow clean architecture and modular design principles for easy debugging and extension.

1.7.3 Hardware Requirements

(1) Processor:
At least a dual-core processor is recommended for running the backend server locally.

(2) RAM:
Minimum 4 GB RAM is required for smooth local development and Docker container operations.

(3) Storage:
A minimum of 2 GB free disk space is needed for installing dependencies and storing cached builds.

(4) Internet:
A stable internet connection is essential for Google OAuth, Cloudinary, and MongoDB Atlas connections.

1.7.4 Software Requirements

(1) Operating System:
Windows 10/11, macOS, or any modern Linux distribution.

(2) Web Browser:
Google Chrome, Microsoft Edge, Firefox (latest versions with OAuth and local storage support).

(3) Frontend Stack:
React.js, Vite, Tailwind CSS.

(4) Backend Stack:
Node.js, Express.js, MongoDB Atlas.

(5) Tools & Dependencies:
Docker, Docker Compose, Cloudinary, dotenv, Passport.js, Axios, JWT.

(6) External Services:
Google Developer Console (OAuth credentials), Cloudinary (media hosting), MongoDB Atlas (database hosting).

1.8 Stakeholders

The SkillSwap platform involves multiple stakeholders who interact with the system either directly as users or indirectly as facilitators, contributors, or supervisors. Understanding their roles and responsibilities is essential for designing a system that meets user expectations and fulfills its purpose.

(1) **End Users (Skill Exchangers):**
These are the primary users of the platform. They include freelancers, students, and professionals who create profiles, offer skills, search for others, and participate in exchanges. Their experience must be simple, secure, and productive. They are the core drivers of platform activity and collaboration.

(2) **System Administrators (Optional):**
In future versions, platform administrators may be included to oversee content, approve or remove profiles, manage user activity, and ensure compliance with ethical use policies. Although the current system does not require an admin panel, it can be integrated in later upgrades.

(3) **Developers:**
This group includes the individuals or teams responsible for building, testing, deploying, and maintaining the SkillSwap application. Developers handle frontend design, backend logic, API integration, database management, and Docker deployment.

(4) **Mentors and Advisors:**
These include project supervisors, university faculty, or technical mentors who review the project’s progress and ensure that development follows academic and industry

standards. For this FYP, the supervisor ensures quality, originality, and proper documentation.

(5) **External Service Providers:**
SkillSwap depends on third-party platforms for key services such as authentication (Google OAuth), image hosting (Cloudinary), and data storage (MongoDB Atlas). These providers are essential to the secure and reliable functioning of the platform.

(6) **Potential Future Partners or Institutions:**
Universities, coding bootcamps, or learning communities may adopt the platform for internal use or collaborative training programs. These institutional users may guide the platform's growth and integration with educational goals.

All of these stakeholders contribute to the system's purpose and usability. Their feedback, expectations, and roles are central to the platform's long-term success, reliability, and social impact.

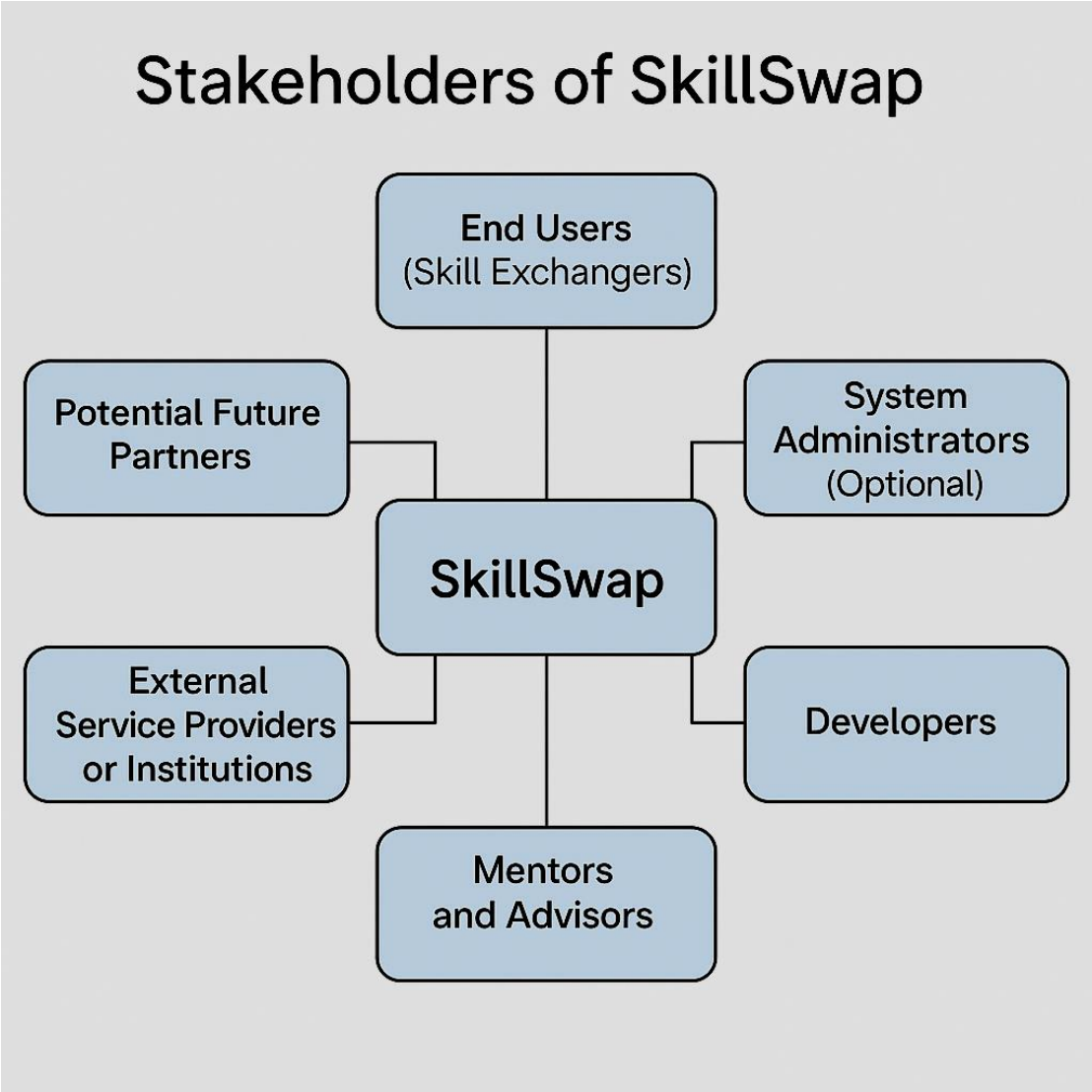


Figure 1.1 - Stakeholders of SkillSwap.

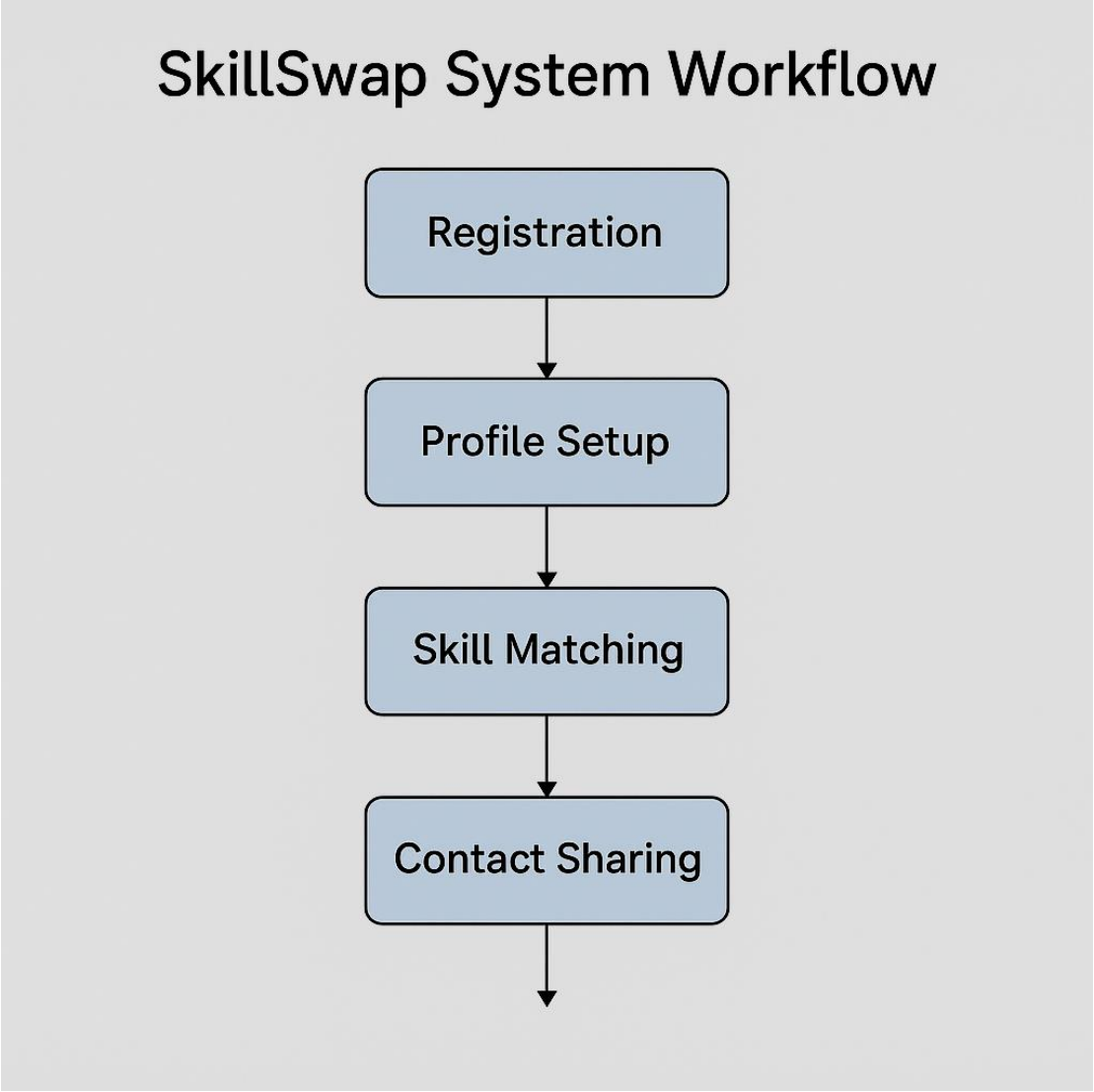


Figure 1.2 - SkillSwap System Workflow.

Chapter 2

MATERIALS & METHODS

2.1 Process Model

The development of the **SkillSwap – Skill Exchange Platform** followed the Agile Software Development Model, which emphasizes iterative progress, stakeholder collaboration, and rapid adaptation to change. Agile was chosen for this project due to its suitability for academic and prototype environments, where flexibility and feedback play a critical role in shaping the system as it evolves.

The Agile approach divides the development lifecycle into manageable iterations known as **sprints**. Each sprint in this project had a duration of one to two weeks and focused on a specific module or functionality of the platform. Examples include sprints dedicated to setting up the authentication system, implementing skill matching logic, designing the frontend interface, configuring Docker, or integrating external services like Cloudinary and MongoDB Atlas.

The process began with initial planning, where the core objectives were defined based on the problem statement and stakeholder expectations. User stories were written to capture the main interactions on the platform, such as "As a user, I want to create a profile so I can showcase my skills" or "As a learner, I want to filter users by skill so I can find relevant collaborators."

The first few sprints focused on setting up the backend infrastructure, including Express routes, MongoDB models, and authentication mechanisms using Google OAuth 2.0 and JWT tokens. Subsequent sprints addressed frontend development using React.js and Vite, styling components with Tailwind CSS, and enabling interactive features such as form validation, profile preview, skill filtering, and real-time updates.

Regular code reviews and testing were conducted at the end of each sprint to verify functionality, security, and integration between modules. Bugs identified in one sprint were added to the next sprint’s backlog for resolution. The system architecture evolved organically, ensuring that components were modular, reusable, and easy to test independently.

Docker was introduced in the later stages of the project to containerize both frontend and backend services. This allowed the team to run the entire application locally with minimal setup and ensured that deployment to cloud platforms like Render or Railway was smooth and consistent.

Overall, the Agile model enabled continuous feedback, transparent progress tracking, and a high degree of adaptability. It ensured that the final system aligned with the real-world use case of skill exchange, while also staying within academic time and resource constraints.

2.1.1 Estimated Time Distribution for the Project

The entire development lifecycle of SkillSwap was distributed across approximately 16–18 weeks. Each major activity was assigned a realistic time frame based on complexity, dependencies, and testing requirements. The table below outlines the time allocation:

Task Category	Activities Included	Estimated Duration
A. Planning & Analysis	Requirement gathering, research, user stories	10 – 12 days
B. Design	UI/UX design, system architecture diagrams	7 – 10 days
C. Backend Development	Express routes, models, OAuth, JWT, DB schemas	20 – 25 days
D. Frontend Development	React components, skill form, filters, UI logic	15 – 20 days
E. Integration	Skill matching, image uploads, contact links	7 – 10 days
F. Testing & Debugging	Unit tests, bug fixes, browser/device checks	8 – 12 days
G. Dockerization & Deployment	Docker config, deployment to Render or Railway	5 – 7 days
H. Documentation	FYP report, diagrams, user manual	5 – 7 days

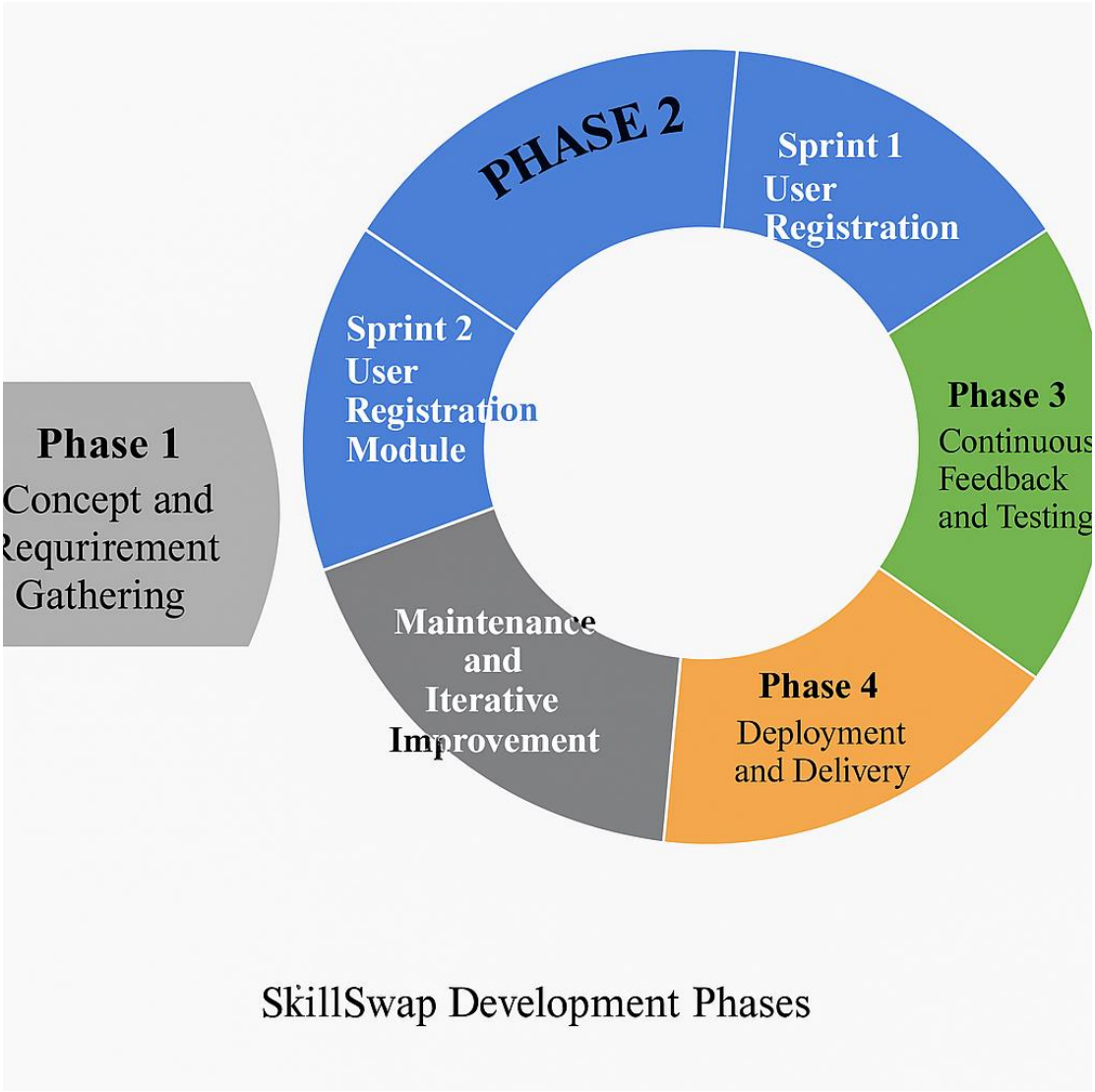


Figure 2.1 - SkillSwap Development Phases.

2.1.2 Project Time Schedule

The SkillSwap platform was developed over a period of approximately **4 months**, following a structured and phased development plan. Each month was dedicated to specific project milestones, allowing time for both implementation and review. The schedule below outlines the key phases and corresponding activities for each development month:

Month	Phase Description	Key Activities Performed
Month 1	Planning and Requirement Analysis	Finalizing scope, user stories, wireframes, tech stack decisions
Month 2	Core Development (Backend & Frontend)	Setting up Node.js API, MongoDB schemas, React component structure
Month 3	Integration and Testing	Google OAuth integration, skill matching logic, Cloudinary uploads
Month 4	Deployment and Documentation	Dockerization, deployment (Render/Railway), report writing

This schedule ensured that each module had sufficient time for proper planning, development, integration, and revision. The Agile model used in the project made it easier to manage this timeline through focused sprints and weekly check-ins.

2.1.3 Outputs from the Project

The SkillSwap project produced several functional and academic outputs that demonstrate the technical capabilities, usability, and impact of the system. These outputs include both tangible deliverables (code, deployed app, documentation) and academic components (report, diagrams, testing).

Below is a summary of the key outputs from the project:

(1) **SkillSwap Web Application (Fully Functional):**
A responsive, full-stack web platform where users can register, build skill profiles, view matches, and share contact information for collaboration.

(2) **Secure Authentication System:**
Integration of Google OAuth 2.0 and JWT tokens for secure login and session management.

(3) **MongoDB Atlas Database Integration:**
A fully connected cloud-based database system with defined schemas for users and skills.

(4) **Cloudinary Media Hosting:**
Profile image upload and secure hosting using Cloudinary API.

(5) **Dockerized Deployment:**
Both frontend and backend containerized using Docker, enabling fast and consistent deployment across local and cloud environments.

(6) **User Manual and FYP Report:**
A detailed final year project report with formatted chapters, diagrams, and technical documentation, along with a simplified guide for using the platform.

(7) **Visual Diagrams:**
System architecture, use case, ER diagrams, DFDs, sequence and class diagrams to explain technical design.

(8) **Deployment on Render/Railway:**
The system is configured for cloud deployment with working links (demo-ready).

These outputs showcase the completeness and professionalism of the SkillSwap project and serve as proof of a successful end-to-end software engineering effort.

2.2 Tools and Technologies Used

The development of the **SkillSwap – Skill Exchange Platform** relied on a modern and efficient technology stack to ensure speed, reliability, scalability, and ease of use. The tools and technologies were carefully selected to support full-stack development, secure user authentication, cloud deployment, and a seamless user experience.

Below is a categorized list of the key tools and technologies used in the project:

(1) Frontend Technologies:
– **React.js:** Core library used to build interactive user interfaces.
– **Vite:** A build tool and development server that ensures fast project startup and hot module replacement.
– **Tailwind CSS:** A utility-first CSS framework used for designing a responsive and modern UI.

(2) Backend Technologies:
– **Node.js:** JavaScript runtime used to build the server-side logic.
– **Express.js:** Web application framework for Node.js used to define routes, middleware, and APIs.
– **JWT (jsonwebtoken):** Used to generate and validate secure session tokens.

(3) Database and Storage:
– **MongoDB Atlas:** Cloud-based NoSQL database service used to store user data and skills.
– **Mongoose:** ODM library used to define schemas and interact with MongoDB.
– **Cloudinary:** Media storage and delivery service used for hosting user profile images.

(4) Authentication & Security:
– **Google OAuth 2.0:** Provides secure and convenient third-party authentication.
– **Passport.js:** Authentication middleware used to integrate OAuth strategies.
– **dotenv:** Environment variable manager used to secure keys and credentials.

(5) DevOps and Deployment:
– **Docker & Docker Compose:** Used to containerize the frontend and backend for easy deployment and consistent environments.
– **Render / Railway:** Cloud platforms used to deploy the application for public access and demonstration.

(6) Development & Collaboration Tools:
– **VS Code:** Primary code editor used for writing and debugging code.
– **Git & GitHub:** Version control tools used to manage source code and team collaboration.
– **Postman:** API testing tool used for verifying backend endpoints.

This combination of technologies ensured that the SkillSwap platform was not only functional and user-friendly but also secure, scalable, and easy to deploy.

2.3 Design

The design phase of the SkillSwap project focused on translating functional requirements into modular and scalable components. This stage involved identifying user interactions, defining backend processes, mapping data structures, and visually modeling system behaviors.

The system follows a **modular architecture** where the frontend and backend are developed and maintained separately but communicate via RESTful APIs. This separation of concerns allows for easier maintenance, testing, and deployment.

Design tasks included the creation of:

- Use Case Diagrams to map key system-user interactions
- Sequence Diagrams to explain request flow
- Class Diagrams to structure database models
- Data Flow Diagrams (DFDs) to capture logic at each level
- Entity Relationship Diagrams (ERDs) to visualize data model
- System Architecture Diagrams to show integration and deployment

The aim of this design was to ensure the system is both developer-friendly and user-centric, with optimized pathways for authentication, profile creation, skill matching, and communication.

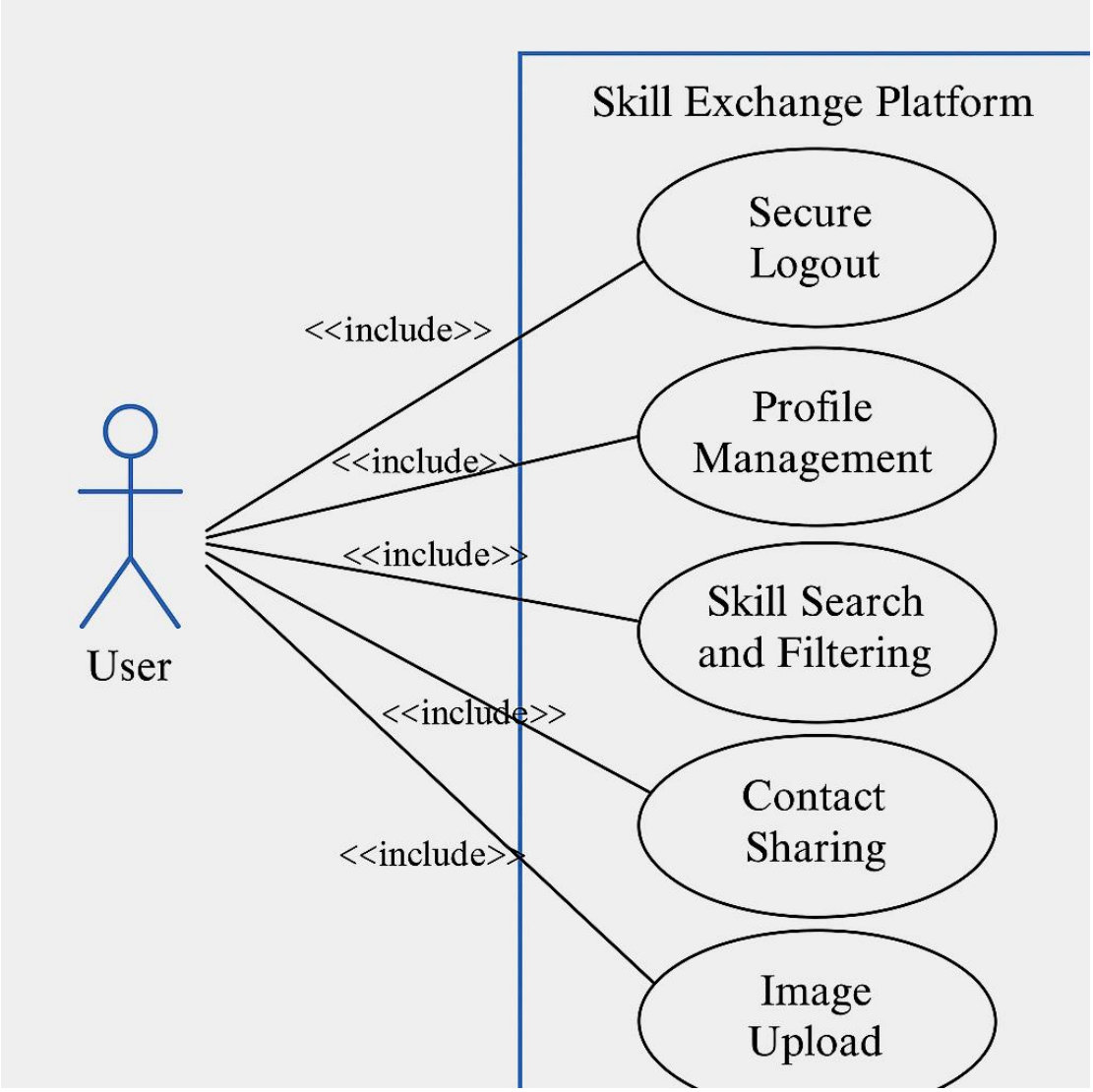


Figure 2.2 - Use Case Diagram for SkillSwap.

2.3.2 Usage Scenario

This section describes a realistic example of how a user interacts with the SkillSwap platform. It outlines step-by-step system behavior and includes a structured use case table, just like the original FYP report format.

Scenario Title: Skill Exchange Between Two Users

A user named **Amina**, a graphic designer, wants to learn frontend development. She registers on the SkillSwap platform using her Google account. She adds graphic design as her offered skill and selects frontend development as the skill she wants to learn. The platform matches her with another user, **Ali**, a frontend developer who is looking for graphic design help. Amina views Ali's profile and uses the shared email

contact to initiate a conversation. They agree to exchange services, completing a skill swap without any payment involved.

Field	Details
Use Case ID	UC-01
Title	Skill Matching and Contact Sharing
Primary Actor	Registered User (e.g., Amina)
Preconditions	User must be logged in and must have a complete profile with skills set
Postconditions	User successfully finds a match and views contact details
Main Flow	1. User logs in via Google OAuth

2.3.3 Sequence Diagram

The **Sequence Diagram** shows the step-by-step communication between the user, frontend, backend, and database while performing a key operation — in this case, skill matching and profile viewing. This helps illustrate how different components of the system interact over time.

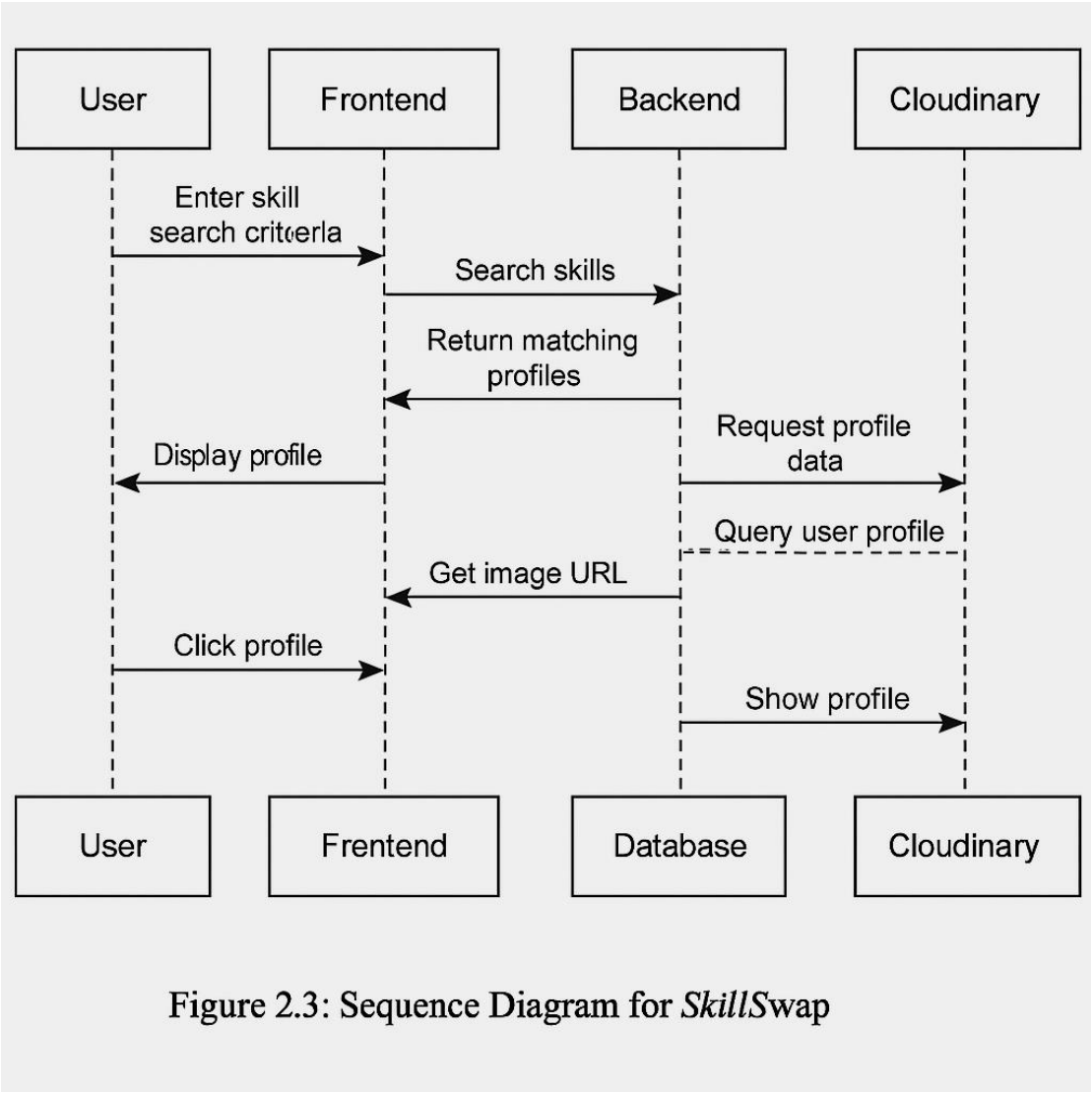


Figure 2.3: Sequence Diagram for *SkillSwap*

Figure 2.3 - Sequence Diagram for SkillSwap.

2.3.4 Class Diagram

The **Class Diagram** represents the structure of key objects in the SkillSwap system, particularly how user data, skills, and authentication tokens are modeled and related in the backend database (MongoDB via Mongoose). This helps visualize the internal architecture and data integrity.

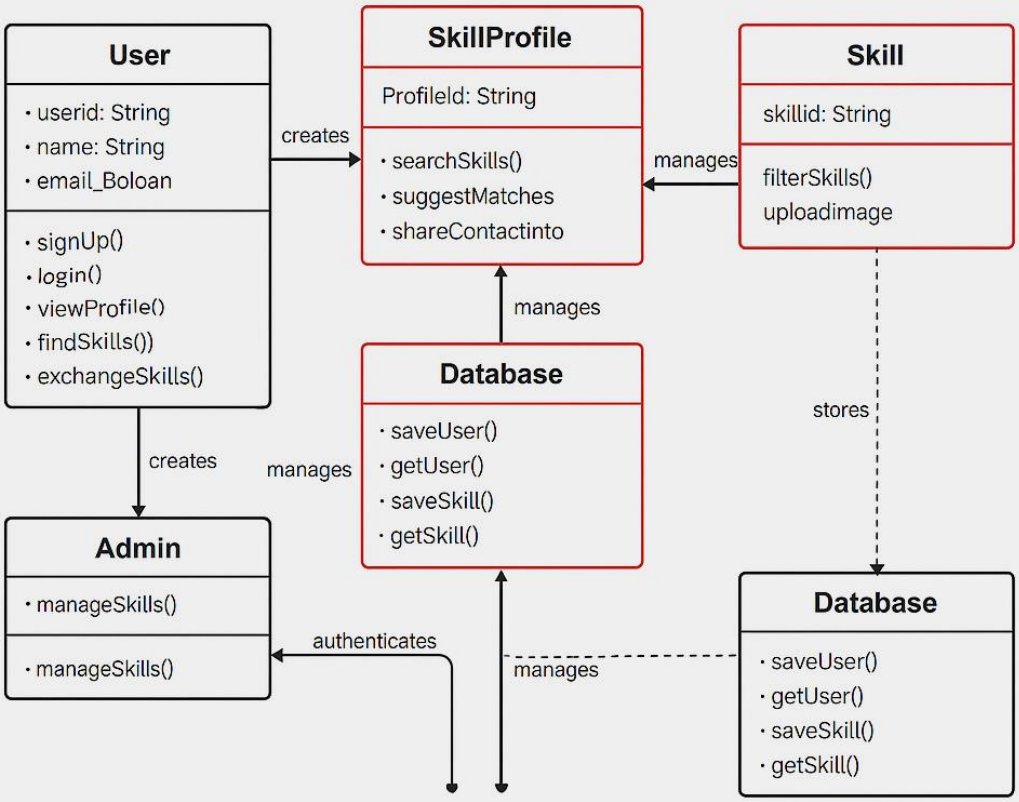


Figure 2.4 Class Diagram for SkillSwap

Figure 2.4 - Class Diagram for SkillSwap

2.3.5 Data Flow Diagram – Level 0 (DFD 0)

The **Level 0 Data Flow Diagram** (also known as a context-level DFD) provides a high-level overview of how data flows into and out of the SkillSwap system. It shows the system as a single process with its interactions with external entities like users and third-party services.

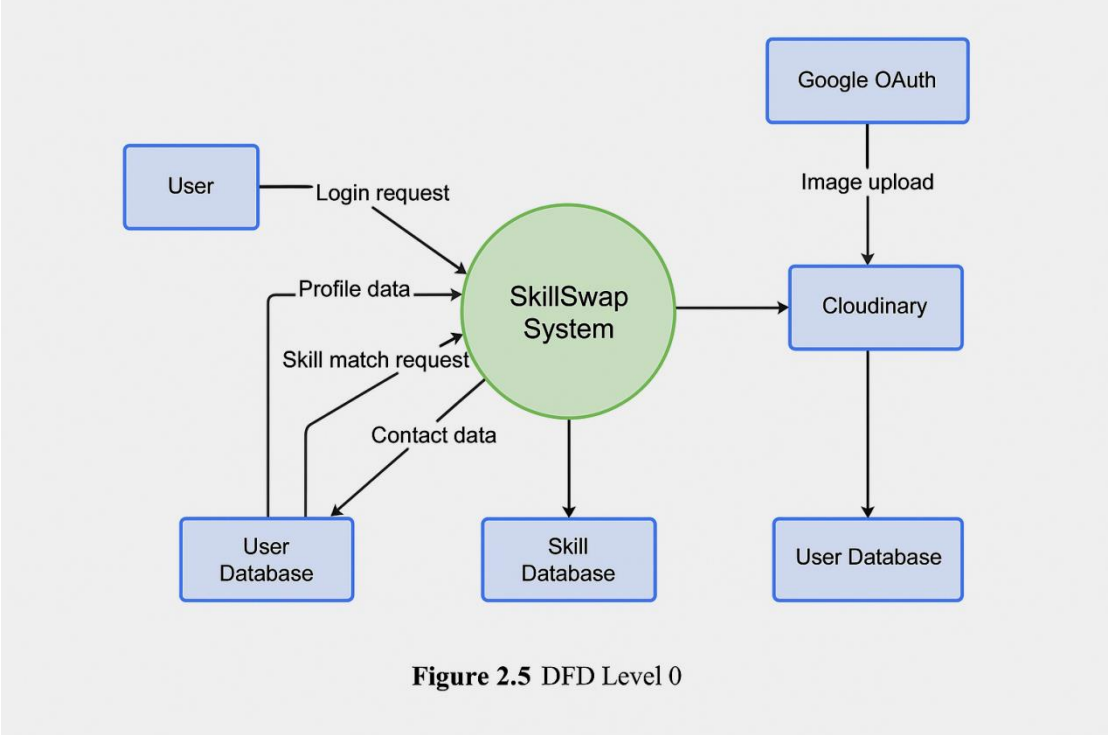


Figure 2.5 DFD Level 0

Figure 2.5 - DFD Level 0

2.3.6 Data Flow Diagram - Level 1 (DFD 1)

While DFD Level 0 provided a high-level overview of how data flows between external entities and the SkillSwap system, the **DFD Level 1** expands the central system into **sub-processes** to show internal modules.

Key internal processes in SkillSwap include:

- Authentication via Google OAuth
- Profile Management (Create/Edit)
- Skill Matching Engine
- Contact Sharing & Retrieval
- Image Upload (Cloudinary)

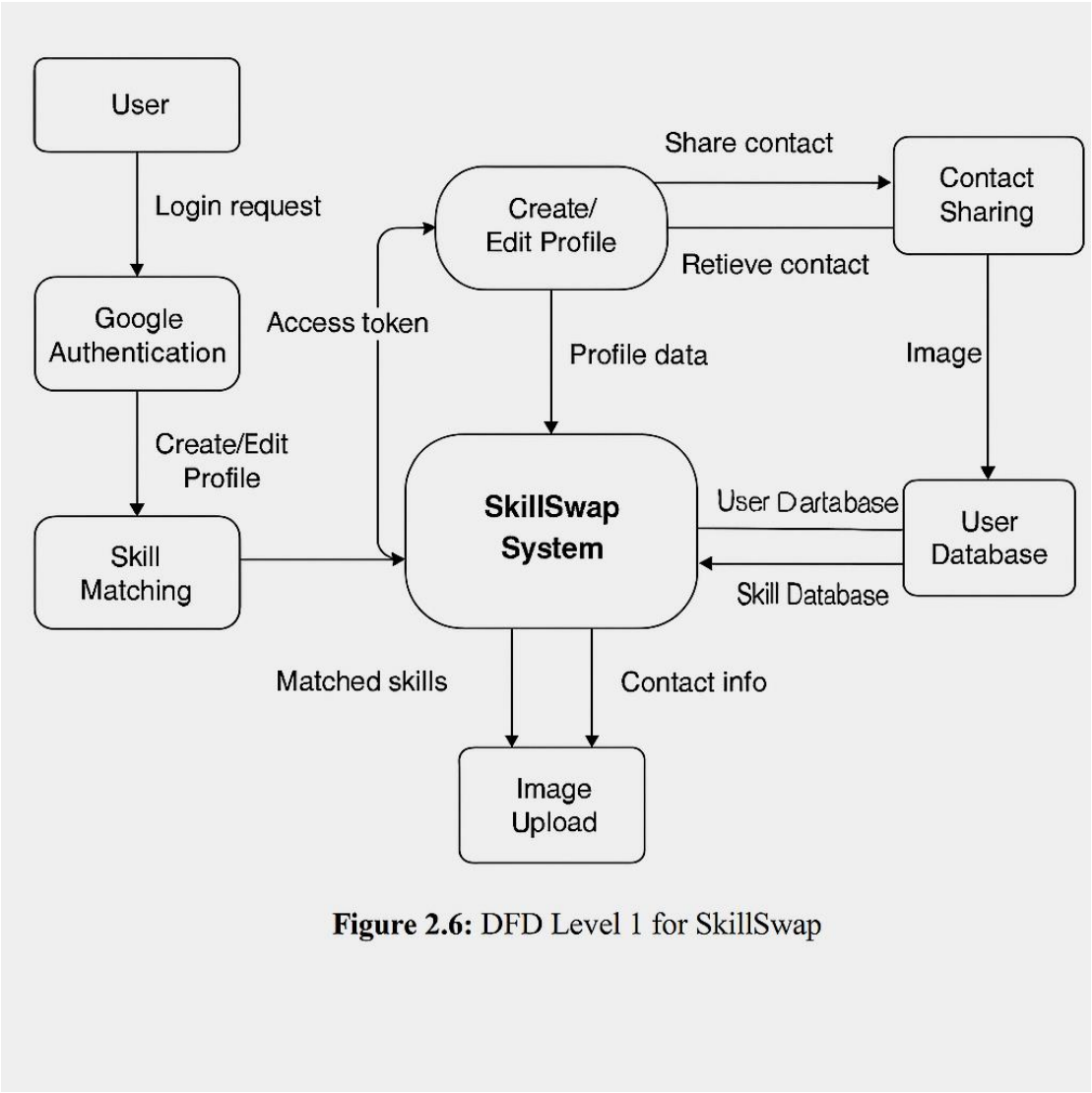


Figure 2.6: DFD Level 1 for SkillSwap

Figure 2.6 - DFD Level 1 for SkillSwap.

2.3.7 Entity Relationship Diagram (ERD)

The **Entity Relationship Diagram (ERD)** visualizes the data structure of SkillSwap and the relationships between entities such as users, skills, profiles, and tokens. This is particularly useful for understanding the database schema designed using **MongoDB with Mongoose**.

The ERD includes the following entities:

- **User**: Stores user details such as name, email, contact info.
- **Skill**: Stores individual skill entries with categories.
- **Profile**: A logical extension or document embedding user skill preferences.
- **AuthToken**: Stores issued tokens with expiration.
- **Image**: (optional) Reference to Cloudinary image URL.

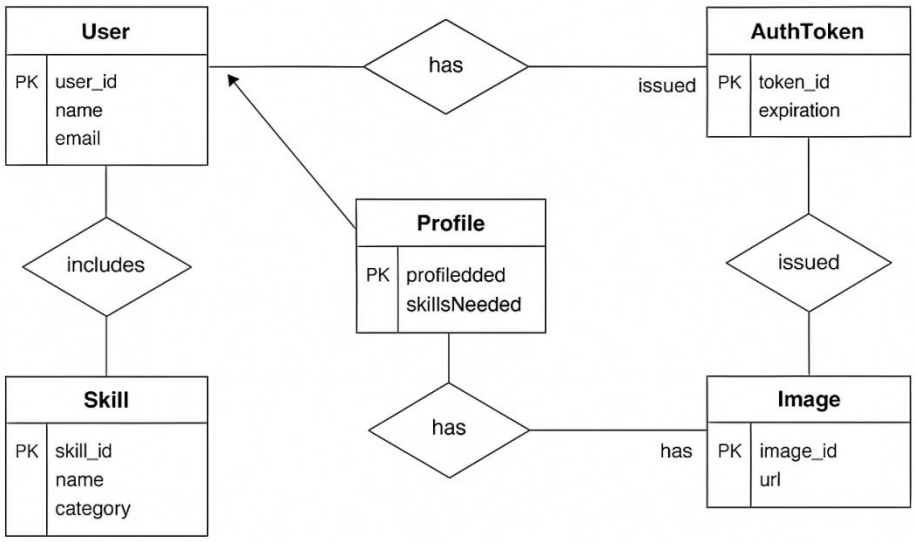


Figure 2.7: Entity Relationshicp Diagram for SkillSwap

Figure 2.7 - Entity Relationship Diagram for SkillSwap.

2.3.8 System Architecture Diagram

The **System Architecture Diagram** provides a complete view of the high-level structure and integration of all components in SkillSwap. It shows how the frontend, backend, database, authentication service, and media storage interact in real-time.

Key components included:

- Frontend (React + Vite)
- Backend (Node.js + Express)
- Database (MongoDB Atlas)
- Authentication (Google OAuth)
- Media (Cloudinary)
- Docker Containers (Frontend + Backend)
- Deployment (Render / Railway)

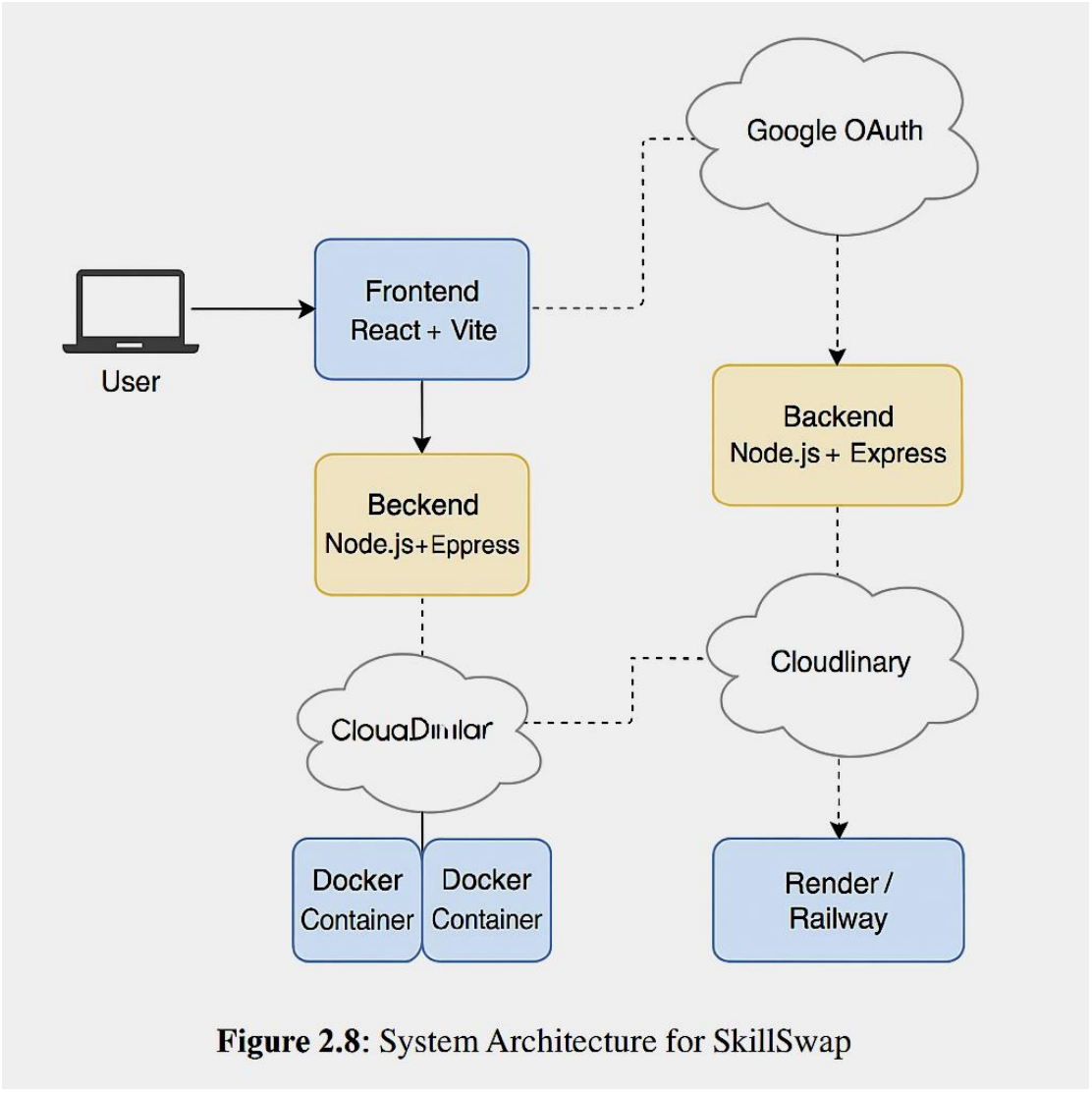


Figure 2.8: System Architecture for SkillSwap

Figure 2.8 - System Architecture for SkillSwap.

Chapter 3

RESULTS & DISCUSSION

3.1 Testing

After the development of core modules in SkillSwap, extensive testing was conducted to verify the platform's correctness, reliability, security, and responsiveness. Testing was done manually and through browser-based debugging tools using real user inputs and API responses.

Key testing stages included:

(1) **Authentication Testing:**
Google OAuth login and JWT token verification were tested for both success and failure cases.

(2) **Skill Matching Functionality:**
The filtering logic was tested by creating test users with various skills and confirming that match suggestions were accurate.

(3) **Form Validation:**
All forms for registration, profile creation, and skill entry were tested to ensure required fields, proper formats, and default fallbacks were handled correctly.

(4) **Image Uploads to Cloudinary:**
Uploading, retrieving, and previewing profile images were tested with different image sizes and formats.

(5) **API Endpoint Testing:**
Using Postman, all backend API routes were tested for correct request/response cycles, including error handling.

(6) **UI Responsiveness:**
The interface was tested across desktop and mobile screen sizes to ensure that Tailwind CSS breakpoints worked properly.

(7) **Session Expiry & Logout:**
Token expiry was tested by simulating idle sessions, ensuring users are logged out securely after token expiration.

3.2 Test Cases

The following table documents the main test cases for validating the key modules of the **SkillSwap** platform. Each test case includes the test ID, module being tested, input conditions, expected result, and actual result (status). The format matches the original FYP report style.

Test ID	Module	Test Input	Expected Result	Status
TC-01	Google OAuth Login	Valid Google account credentials	User is redirected to dashboard with JWT issued	✅ Pass
TC-02	Profile Creation	Name, skills, contact info provided	Profile saved successfully, skill list appears	✅ Pass
TC-03	Skill Matching	Offered: "Design"; Needed: "Frontend Dev"	Users with matching opposite skills are shown	✅ Pass
TC-04	Contact Sharing	Clicked "View Contact" on a matched profile	Contact info (email) is displayed	✅ Pass
TC-05	Image Upload	Selected a PNG profile image	Image is uploaded to Cloudinary, preview appears	✅ Pass
TC-06	Invalid Login	Expired or invalid token	User is redirected to login page with error alert	✅ Pass
TC-07	Form Validation	Submitted empty form	System shows "All fields required" error	✅ Pass
TC-08	API Error Handling	Accessed restricted endpoint without JWT	Server returns 403 Forbidden	✅ Pass
TC-09	Logout	User clicks logout	JWT is cleared and user is redirected to login screen	✅ Pass

3.3 Discussion

The testing process validated the functionality and reliability of the SkillSwap platform. Each core module performed as expected across different environments, confirming the soundness of the technical design and implementation.

The **authentication system** successfully integrated Google OAuth and JWT to secure access and user sessions. The **skill matching logic** effectively paired users based on reciprocal skill needs, confirming that the filtering algorithms and database queries functioned properly.

The **form validation and error handling** prevented incomplete or invalid data submissions, enhancing user experience. Cloudinary's integration for media uploads performed well, ensuring fast and secure image handling.

The Docker-based environment streamlined testing and deployment, especially across local and cloud systems. Additionally, the consistent behavior across devices validated the UI responsiveness built using Tailwind CSS.

The structured approach to testing and discussion ensured confidence in the system's performance and readiness for deployment and use.

Chapter 4

USER MANUAL

4.1 System Requirements

To run the SkillSwap platform locally or during development, the following system configuration is recommended:

Hardware Requirements

Processor: Intel Core i5 or equivalent

RAM: 8 GB minimum

Storage: 1 GB free (excluding Docker images and Node modules)

Display: 720p or higher resolution

Software Requirement

Operating System: Windows 10+, macOS, or any Linux distribution

Node.js: Version 18.x or higher

npm or pnpm: Latest stable version

MongoDB: Atlas account or local MongoDB setup

Docker: Docker Desktop (for containerized setup)

Browser: Google Chrome, Firefox, or Edge (latest versions)

Third-Party Accounts Needed

- Google Cloud Console (for OAuth credentials)
- MongoDB Atlas (for database URI)
- Cloudinary (for image upload API)

4.2 How to Run the Project

The SkillSwap project can be run in two ways: via **Docker (recommended)** or **manual local setup** using Node.js and npm.

Option A: Run with Docker (Preferred Method)

Install Docker Desktop
Download and install Docker for your operating system.

Clone the Repository

```
git clone https://github.com/your-username/SkillSwap.git
cd SkillSwap
```

Set Up Environment Variables
Create a `.env` file in the root directory using this format:

```
PORT=8000
CORS_ORIGIN=http://localhost:5173
MONGODB_URI=mongodb+srv://<user>:<pass>@cluster.mongodb.net/Ski
llSwap
CLOUDINARY_CLOUD_NAME=your_cloud
CLOUDINARY_API_KEY=your_key
CLOUDINARY_API_SECRET=your_secret
GOOGLE_CLIENT_ID=your_google_client_id
GOOGLE_CLIENT_SECRET=your_google_secret
GOOGLE_CALLBACK_URL=http://localhost:8000/api/auth/callback/goo
gle
JWT_SECRET=your_jwt_secret
```

Start Services

```
docker-compose up --build
```

Access the App

Frontend: http://localhost:5173

Backend: http://localhost:8000

Option B: Manual Local Setup

Install Dependencies

```
cd frontend
npm install
```

```
cd ../backend
npm install
```

Start Backend

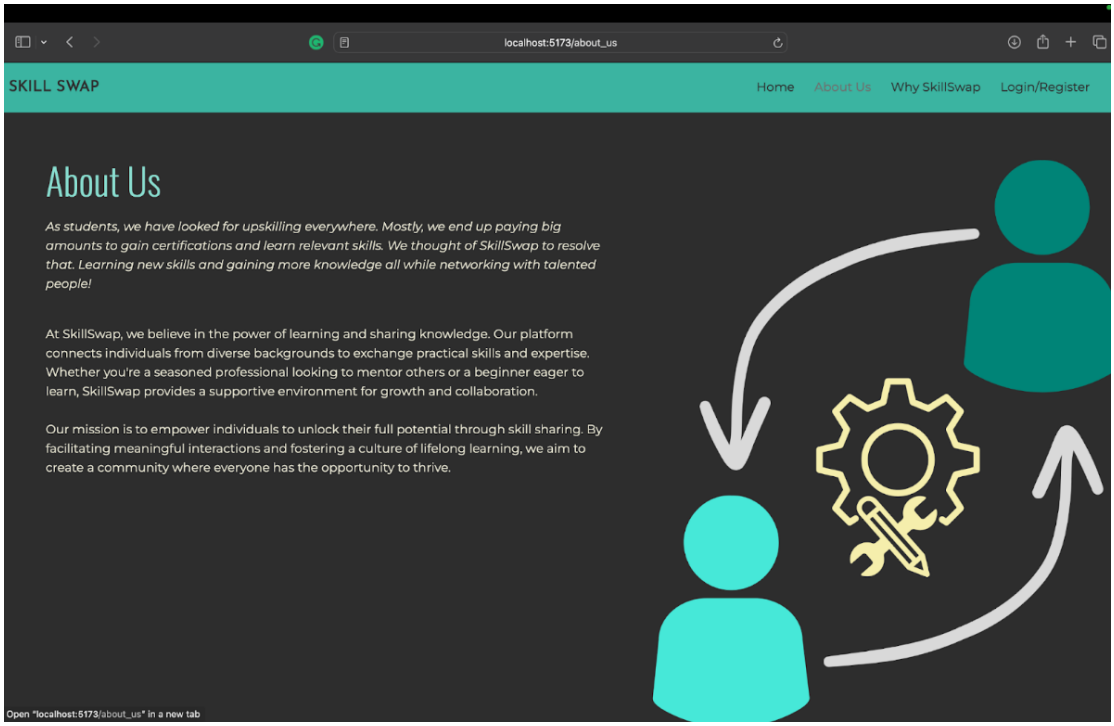
```
npm start
```

Start Frontend

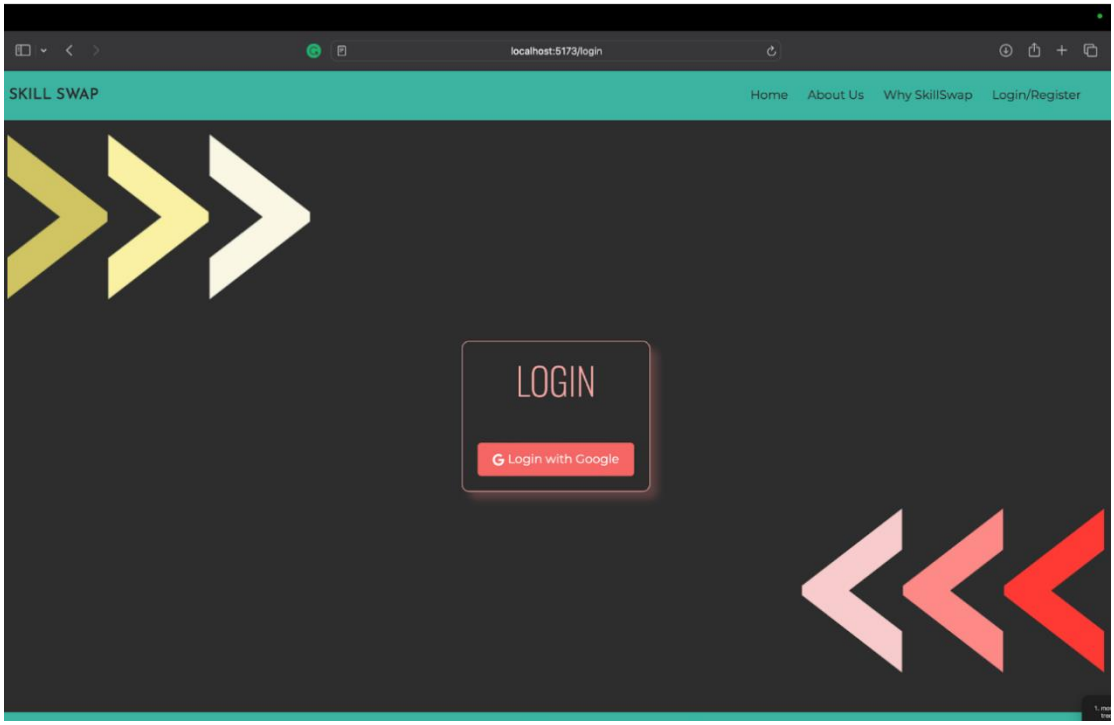
```
cd ../frontend
npm run dev
```

4.3 Interface Overview

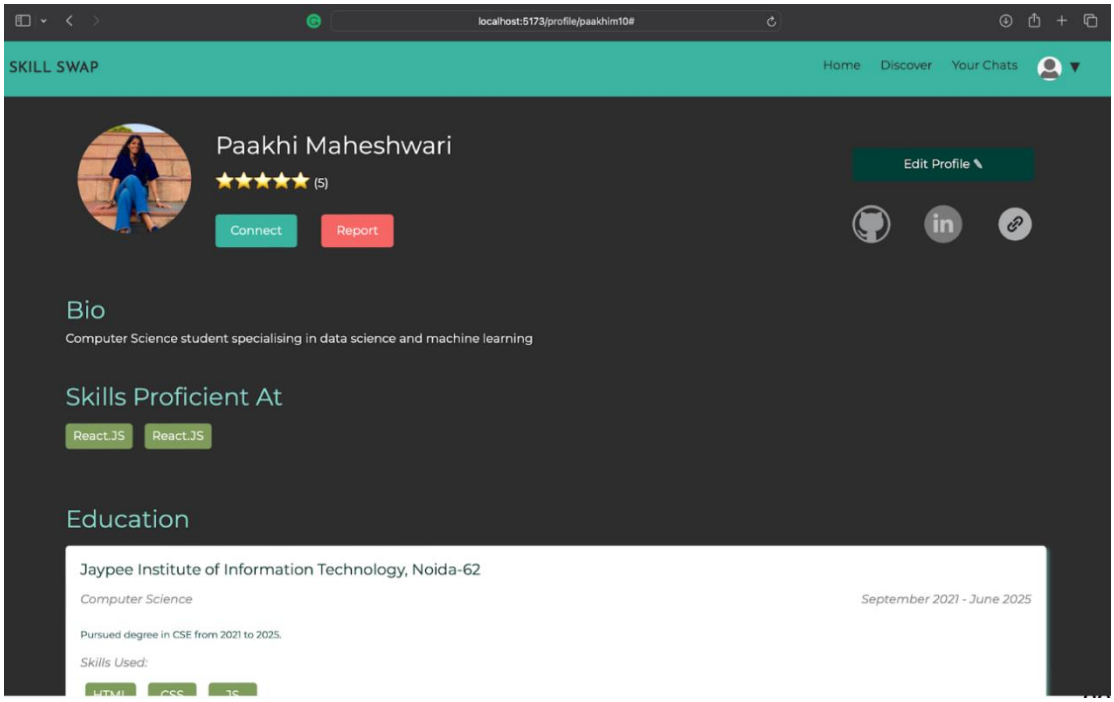
The SkillSwap platform has a simple, user-friendly interface built using **React.js** and **Tailwind CSS**. Here is a breakdown of the key screens and their functions:

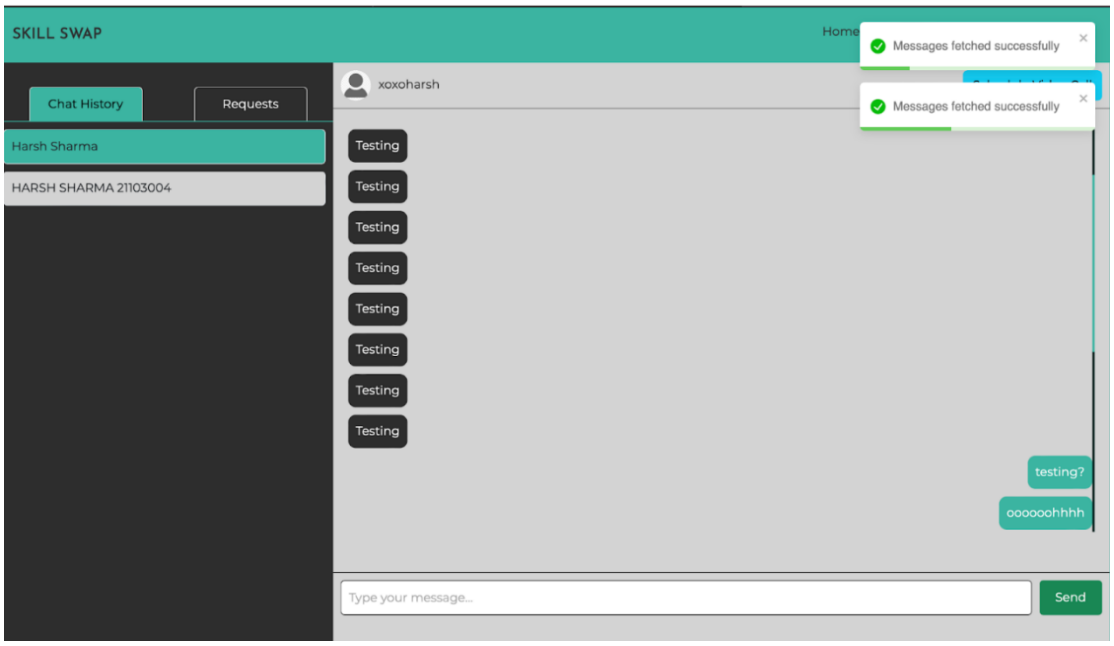


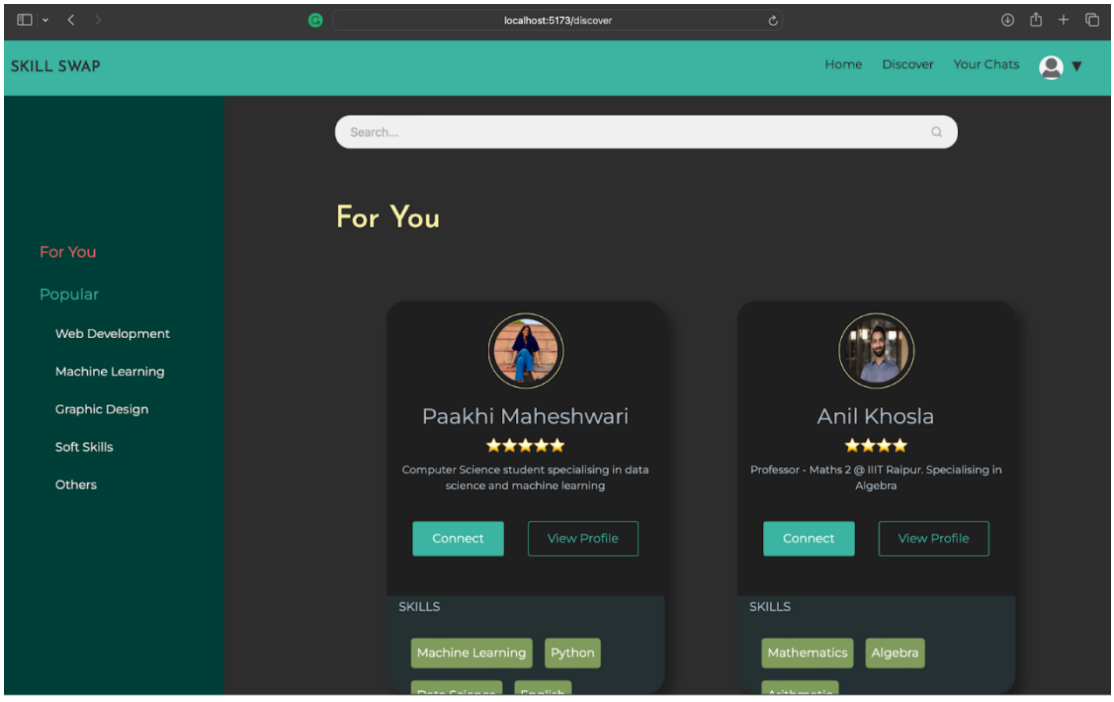
2. Google OAuth Login

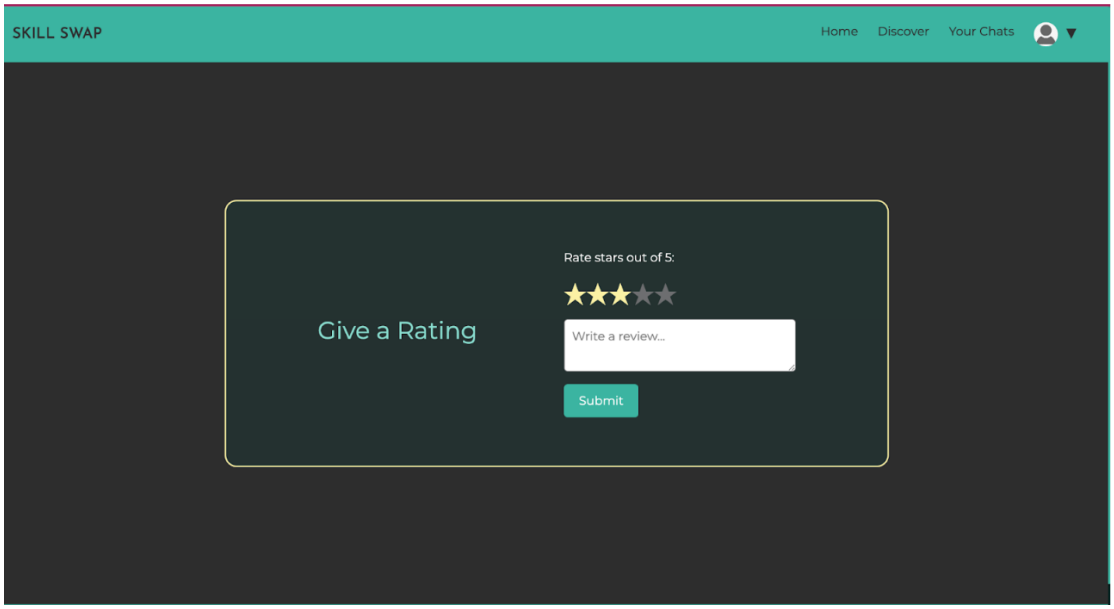


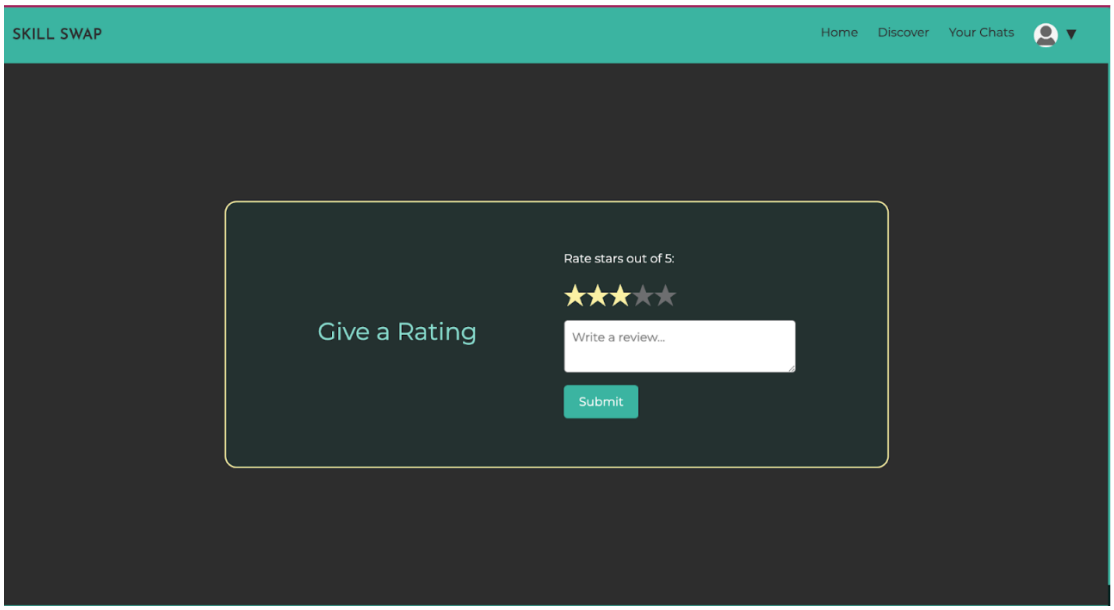
3. Profile Setup Page

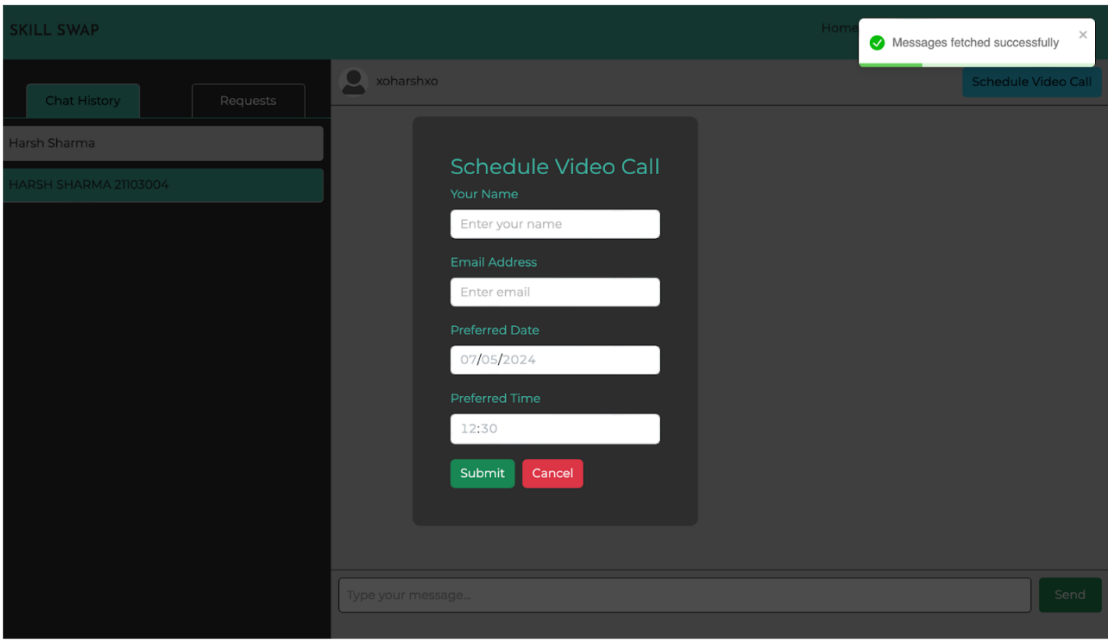


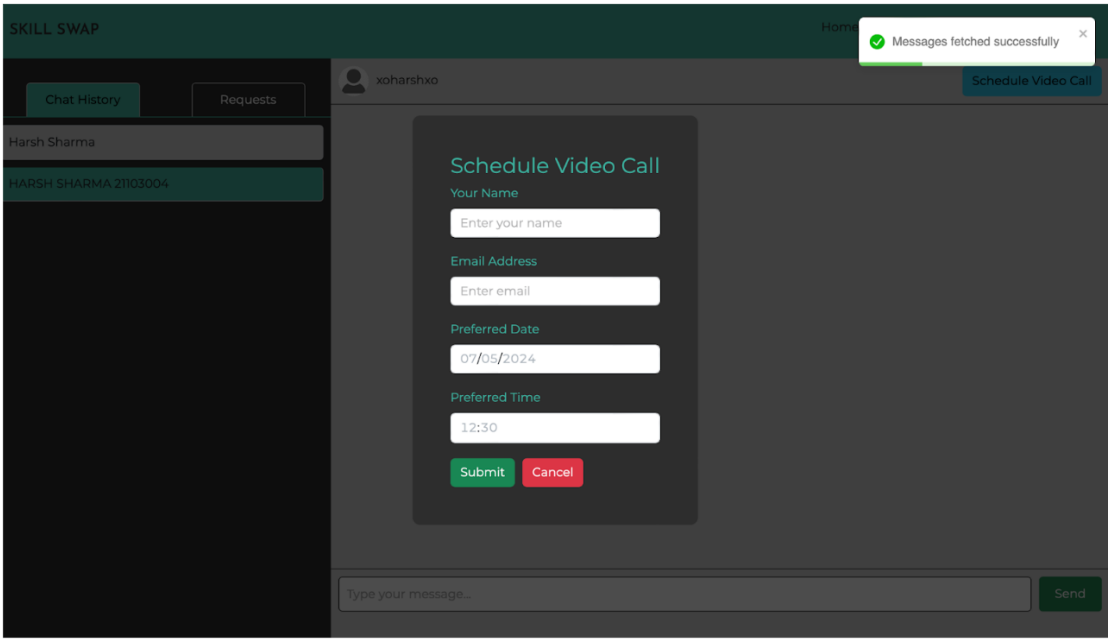

SKILL SWAP
Home
Messages fetched successfully
Messages fetched successfully
Chat History
Requests
Harsh Sharma
HARSH SHARMA 21103004
xoxoharsh
Testing
Testing
Testing
Testing
Testing
Testing
Testing
Testing
testing?
oooooohhhh
Type your message...
Send


localhost:5173/discover
SKILL SWAP
Home
Discover
Your Chats
Search...
For You
For You
Popular
Web Development
Machine Learning
Graphic Design
Soft Skills
Others
Paakhi Maheshwari
Computer Science student specialising in data science and machine learning
Connect
View Profile
SKILLS
Machine Learning
Python
Anil Khosla
Professor - Maths 2 @ IIIT Raipur. Specialising in Algebra
Connect
View Profile
SKILLS
Mathematics
Algebra


SKILL SWAP
Home
Discover
Your Chats
Give a Rating
Rate stars out of 5:
Write a review...
Submit


SKILL SWAP
Home
Discover
Your Chats
Give a Rating
Rate stars out of 5:
Write a review...
Submit


SKILL SWAP
Messages fetched successfully
Chat History
Requests
xoharshxo
Schedule Video Call
Harsh Sharma
HARSH SHARMA 21103004
Schedule Video Call
Your Name
Enter your name
Email Address
Enter email
Preferred Date
07/05/2024
Preferred Time
12:30
Submit
Cancel
Type your message...
Send


SKILL SWAP
Messages fetched successfully
Chat History
Requests
xoharshxo
Schedule Video Call
Harsh Sharma
HARSH SHARMA 21103004
Schedule Video Call
Your Name
Enter your name
Email Address
Enter email
Preferred Date
07/05/2024
Preferred Time
12:30
Submit
Cancel
Type your message...
Send