Princess Sumaya University for Technology
King Hussein School for Computing Sciences

DevHive
“From Developers To Developers”

By
Saeed Haddad - 20191095
Alaa Al-Taher - 20200298

Supervised by:
Dr. Ahmad Al-Tamimi

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Software Engineering

2024

Acknowledgment

Words alone can never fully convey our heartfelt thanks and gratitude to our parents, friends, and everyone who supported us throughout our journey to become who we are today.

We also extend our most profound appreciation to our supervisor, Dr. Ahmad Al-Tamimi, for his unwavering support and encouragement over the past four years.

We are deeply grateful for the dedication, hard work, and time of all our professors, especially those in our fantastic department, for helping us excel in our endeavors.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

Introduction

1.1 Abstract

In today's digital age, freelancing has become one of the backbones of the tech industry, providing developers with opportunities for flexible work on diverse projects at any time and place. However, navigating the world of freelancing can be tedious, with challenges such as finding reliable clients, negotiating fair rates, and managing projects efficiently. To overcome these challenges, we developed a comprehensive, easy-to-use freelancing website explicitly tailored for developers in Jordan.

Our methodology focused on an in-depth analysis of existing freelancing platforms and inducted surveys targeting Jordanian developers interested in the freelancing business to understand what they expect and desire. Leveraging modern web development technologies, we designed a user-friendly platform prioritizing developer empowerment and user satisfaction.

The key features found in "DevHive" are advanced search and filtering options to aid the developers in finding projects that align with their skills and preferences, a feature called "Negotiation Block" where developers and clients can negotiate fair rates and come to an agreement where both parties are satisfied, a bug hunting section that individuals or companies can publish, a rating system that goes both ways for developers and clients, a milestone system to keep track of projects, profiles to showcase users abilities/ interests, and ratings that can attract clients and potentially companies who are looking for developers devoted to their craft.

1.2 Problem Statement

In the age of the digital era, people assume that being involved in the tech industry means having a stable job with high salaries. Unfortunately, this is not the case anymore; layoffs in tech companies are taking a toll on developers worldwide and it's only a matter of time until it hits the Jordanian market; due to the layoffs, some developers choose to join startups compromising their payments, others start their businesses which prove to be quite tricky, and some might even change career paths.

However, many of these developers gravitate towards freelancing since it is flexible and they can make a living. Finding a freelancing website with reliable clients, respectable fair rates, and a professional atmosphere would be impossible because such a platform does not exist in the Jordanian market. We must recognize that, shockingly, a website that focuses on freelancing only for developers has yet to be applied, yet greatly demanded. This is a huge gap in the market that flies over the heads of stakeholders and investors.

"DevHive" offers developers in Jordan a platform that focuses on the IT industry alone, it is a haven for all developers who are keen to showcase their skills and find the work-life balance they need while being provided with the necessary tools to compete in this highly competitive industry.

1.3 Problem Motivation

We are passionate about this because we offer new opportunities to developers in Jordan, expand the market, and defy the ordinary. Every developer dreams of having a platform that brings people together to work on projects that align with their interests while making a living.

One of the main challenges is gaining people's acceptance of this idea, as most are accustomed to certain lifestyles. However, we are optimistic that people will pick up on it as soon as they hear about DevHive.

The impact of building this project will be significant for all involved parties, whether it is time or convenience-related.

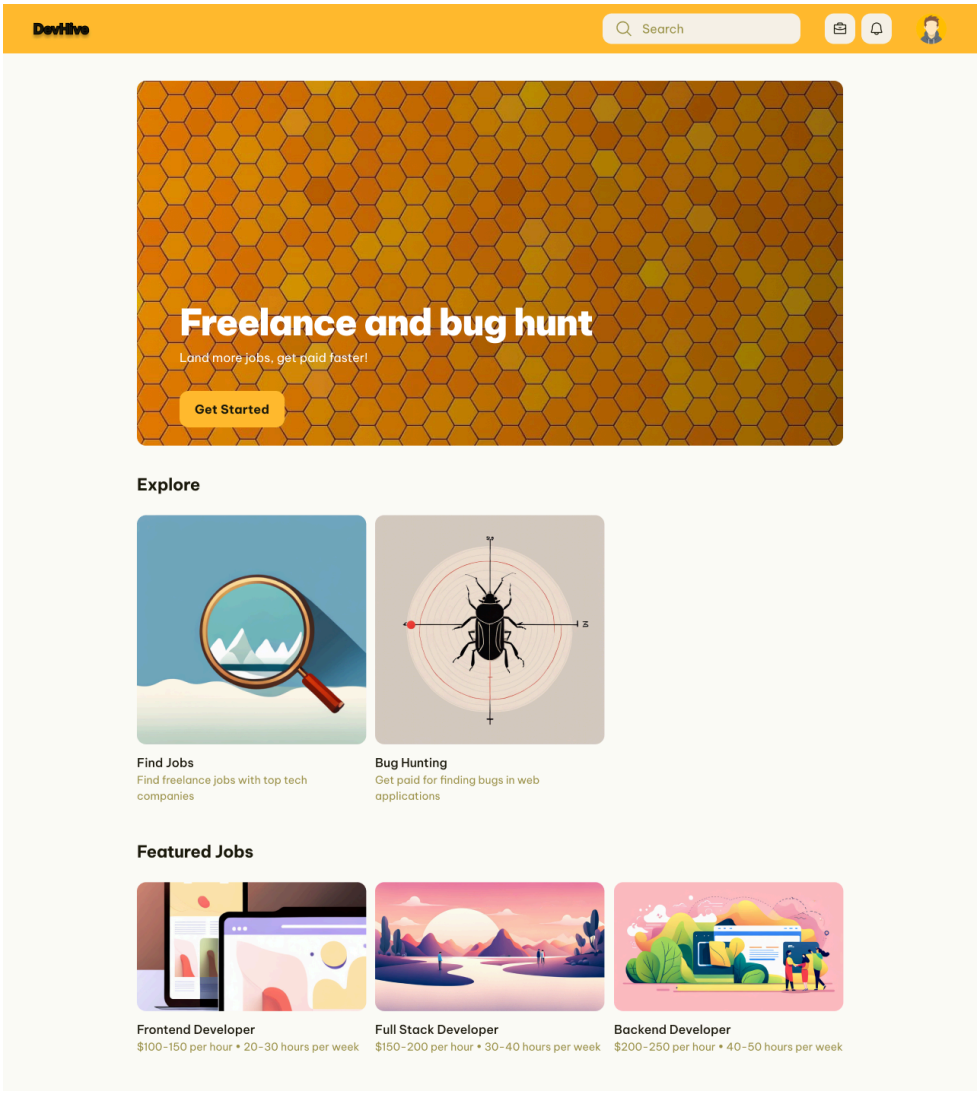
1.4 Related Work

Based on the market research, multiple solutions exist related to the problem that DevHive aims to solve, but each one has a downside that makes DevHive stand out.

Most solutions focus on something other than the Jordanian market and workforce, while other businesses have a more general user base. Some companies provide solutions at a high fee rate without room for negotiation. DevHive aims to close the gap between both parties and offer a professional way to ensure satisfaction. Other competitors lack the features DevHive will include, making it a unique solution.

1.5 Scenario

1. The users will navigate to our website.



2. They will be welcomed with a well-designed interface that is practical and straightforward. To use our services, they must create and verify their account.

3. The users will choose their skills and interests on separate occasions.
    a. They will be met with a bubble selection process to choose their skills such as the programming languages they know best.

    b. A mandatory section where they provide the education they had, whether it's their universities, companies they've worked with, or even past projects.

4. Now they can navigate our website freely to check what we have to offer.

5. The users can choose between two main options

    a. **Freelance:** where users can find job posts for projects that meet the skills and interests they chose when they created their accounts, alongside some recommendations based on what they browse.

    b. **BugHunting** is a creative model in which users post bug-infested code, and the post owner monetarily rewards the user whose solution is correct.

6. In Freelance, the price is negotiable, this is where the “Negotiation Block” feature comes in. An initial price will be set by the client, stating the job, what resources it would take, and the deadline for the project/task. The client and freelancer can negotiate the payment and resources, where they can exchange ideas on the said project, some details may go over the heads of clients, so the developer can state what it actually would take to complete the task based on their experience. An auction-like window will be displayed in the “Negotiation Block” at all times so both of them can see the currently set price. If they agree, the two parties click on the accept button to lock in and proceed with the project.

7. In BugHunting, the price is fixed. A client or company posts their bug-infected code in this section and waits until freelancers provide solutions; then, freelancers will be rewarded accordingly. This idea keeps freelancers on the lookout, always checking for new posts, which is beneficial for “DevHive” as it will have people always coming back for more since it is very competitive.

1.6 Literature Review

The flourishing temporary market has demonstrably fueled the rise of freelance developers in the tech industry. Upwork's 2023 Freelance Economy Report [Upwork's 2023 Freelance Economy Report] underscores this trend, highlighting a surge in freelance workers and a growing demand for flexible work arrangements and project variety. However, research also identifies persistent challenges faced by this expanding workforce.

Challenges for Freelance Developers in Jordan

- **Finding Reliable Clients:** A significant hurdle for Jordanian freelance developers is securing consistent and dependable clientele.

- **Project Management and Communication:** Effective project management, especially when working with multiple clients simultaneously, presents complexities for freelance developers.

- **Fair Pay and Negotiation:** Negotiating fair rates and securing competitive pay is another major challenge for Jordanian freelancers.

Limitations of Existing Freelancing Platforms

While established platforms like Upwork, Freelancer, and Fiverr offer some solutions for connecting developers with clients, there are documented limitations. Research on developer experiences with these platforms reveals dissatisfaction with:

- **Restricted Search Functionality:** Due to limitations in search functionalities, finding projects that align with specific skills and experience can take time and effort.

- **Lack of Built-in Negotiation Tools:** The absence of integrated negotiation tools within these platforms can lead to power imbalances, unfair compensation for developers, and high service fees.

DevHive: A Tailored Solution

The following are studies about most challenges and limits of the platforms that exist now under the need for having more complete solutions, especially for the Jordanian freelancing driving developers. DevHive aims to fix and bridge this gap in a good way by providing a user-friendly platform that gives developers the power and encourages a more balanced interesting helpful successful freelancing experience.

Improvements Addressed by DevHive

- **Advanced Search and Filtering:** DevHive's search features will enable developers to efficiently find projects that align with their specific skills and preferences.

- **Negotiation Block:** This unique feature facilitates fair and transparent negotiation of rates between developers and clients.

- **Bug Hunting Section:** This innovative feature provides developers with the opportunity to showcase their skills and earn rewards by fixing bugs in client-submitted code.

- **Two-Way Rating System:** A robust rating system evaluates developers and clients, fostering trust and accountability within the platform.

By doing that, DevHive is aiming to revolutionize the freelancing experience for Jordanian developers.

Chapter 2

Software Project Management Plan

2.1 Project Overview

2.1.1 Project Purpose And Objectives

DevHive main purpose is to become the best place for developers to find opportunities in the region. The project intends to put into people's hands a website that will reward you for your skills whenever and wherever making every moment valuable.

As DevHive thrives on originality, dedicated to quality, the project objectives are simple. Easy, direct, and satisfying website to use which is an important aspect. To provide the main functionalities to and from the people who will use it.

The completion date for DevHive is set to be August 2024. In addition, the team should adhere to the quality standards needed to create the website, and most importantly to achieve customer satisfaction.

2.1.2 Project Charter

The Project Charter is a vital piece of the project because it frames the entire project. It describes the outline of the objectives of the project, milestones, & other important related titles.

Project Charter

Project Title: DevHive Date of Authorization: March 26, 2024 Project Start Date: March 26, 2024 Project Finish Date: August 15, 2024
Key Schedule Milestones: ● Accomplish an initial version of the software requirements specifications (SRS) by May 2024. ● Accomplish an initial version of high-level design documentation by May 2024. ● Accomplish a fully functional version of the project by August 2024.
Budget Information: There will be no allocated budget, for now, all the labor is internal and divided among both team members.
Project Objectives: DevHive, a unique platform in the saturated market, aims to help developers in Jordan find a work-life balance and earn money on demand. Unlike other platforms, DevHive offers a meeting ground where users and clients can find mutual satisfaction. The goal is to deliver an initial version of the software requirements specifications (SRS) and high-level design documentation within 3-4 months. The final fully functional and tested version of the product will be delivered within 5-6 months.
Main Project Success Criteria: The platform meets the requirements within the SRS with no contradiction or ambiguity. It finishes on time and within budget. Clients receive what they ask for, and developers are compensated for their work as negotiated.
Approach: ● Creating an initial version of the SRS within the first 4 months of building the project. ● Keep track of tasks using Jira Software. ● Apply an iterative approach to develop the project. ● Apply rigorous testing on the project to ensure functionality. ● Document all work and findings using Google Docs, Jira Software, and GitHub to store multiple versions/backups of the project. ● Meet with supervisor weekly for reviews and status reports.

Roles and Responsibilities

Name	Role	Position	Contact Info
Dr. Hazem Qattous	Supervisor	Supervise	h.qattous@psut.edu.jo
Alaa Al-Taher	Development & Project Manager, Team Member	Developer, Documentation, UI/UX, Front-End/Back-End, Code Optimization	ala20200298@std.psut.edu.jo
Saeed Haddad	Development & Project Manager, Team Member	Developer, Documentation, UI/UX, Front-End/Back-End, Code Optimization	sae20191095@std.psut.edu.jo

2.1.3 Project Scope Statement

The scope, a crucial aspect, defines the items delivered throughout the project's lifetime. This project scope, encompassing the delivery of complete system documentation, a production version of the software with tested main functionalities, and the rollout of the production version to users, is a roadmap to our success. Defining a project's scope is not just necessary, but it's a shared responsibility to ensure adherence to the project's needs.

Project Title: DevHive Date: April 2nd, 2024 Prepared by: Alaa Al-Taher
Project Justification: DevHive is a website that will help users in Jordan to work and fill their free time by being productive and getting paid at the same time. It is a freelancing website that will allow some users to show their skills, use them, and apply them to their projects in exchange for money.
Product Characteristics and Requirements: ● The website shall provide a good search mechanism for the users and projects ● The website shall have a "Negotiation Block" and allow developers and clients to negotiate fair rates and come to an agreement where both parties are satisfied ● The website should have a bug-hunting section that individuals or companies can publish ● The website should have a rating system ● The website should be simple to learn & to use
Summary of Project Deliverables: Deliverables related to project management include the project charter, work breakdown structure, scope statement, project schedule and timeline, and risk analysis. **Product-related deliverables:** 1. System Requirements Specification (SRS) 2. Software Project Management Plan (SPMP) 3. System Design Document (SDD) 4. Source code for backend predictor, skill tree builder, recommender, crawler, and frontend website where users can sign up, enter links they have learned from, and have new skills recommended to them 5. System Test Document (STD) 6. Website domain
Project Success Criteria: To submit a functional website version within 6 months, without exceeding the budget and meeting requirements set in the SRS.

2.1.4 Assumptions And Constraints

No matter how big the project is, identifying the constraints will help define the project boundaries, which are achievable and not. The project manager will measure the effects of the constraints over the four project factors, time, scope, cost, and quality, to ensure the right balance among these factors.

Assumptions:

1. **Availability of Skilled Developers:** DevHive assumes a sufficient pool of skilled developers in Jordan who are interested in freelance opportunities and bug-hunting events.

2. **Access to Technology:** The platform assumes developers can access necessary technology and infrastructure, such as internet connectivity and computing devices, to participate in freelance projects and bug-hunting activities.

3. **Regulatory Compliance:** DevHive assumes it will comply with relevant laws and regulations governing freelancing, data privacy, and intellectual property rights in Jordan.

4. **User Adoption:** The platform assumes that developers and clients will adopt and actively engage with the platform, contributing to its growth and success.

Constraints:

1. **Budget Constraints:** DevHive operates within a predefined budget, limiting the resources available for platform development, marketing, and operations.

2. **Time Constraints:** There are time constraints for project completion and feature rollout, influenced by market demand, competitor activity, and investor expectations.

3. **Technical Limitations:** The platform development may face technical constraints, such as limitations in scalability, security, or compatibility with existing systems and tools.

4. **Market Competition:** DevHive operates in a competitive market landscape, facing competition from existing freelancing platforms and other technology solutions targeting developers in Jordan.

2.1.5 Tools And Techniques

Tools are crucial in project development, supporting various activities and phases throughout the project lifecycle. However, while tools are essential, techniques and approaches are equally significant. The project utilizes a range of strategies to aid in the execution and evaluation of each stage, complemented by modern methods to ensure efficiency and effectiveness.

Tools:

1. **Project Management**

    - **Jira Software:** is a project management tool designed to track tasks and their progress. It offers customizable workflows and comprehensive reporting capabilities to enhance productivity.

    - **Discord:** is a real-time messaging, voice calls, and video conferencing platform that facilitates seamless collaboration among individuals and communities.

    - **GanttProject:** is a project scheduling and management software that facilitates efficient planning, tracking, and visualization of tasks and resources.

2. **Design and Prototyping:**

    - **Figma:** is a collaborative tool used for interface design that enables teams to create, prototype, and iterate on digital designs in real time.

    - **LucidChart:** a cloud-based diagramming and visualization tool that empowers teams to create, edit, and share various diagrams and charts.

3. **Documentation:**

    - **Google Docs:** is a word processing platform that enables users to create, edit, and collaborate on documents in real time. Its intuitive interface and robust sharing features allow seamless collaboration among individuals and teams.

    - **Grammarly:** is a writing assistant that helps users improve their writing quality by providing real-time grammar, spelling, and punctuation suggestions.

Techniques:

The blend of techniques and tools utilized shapes the project's management approach. A combination of methods is employed to oversee and assess progress effectively.

Monitoring team progress and deliverables involves regular meetings to discuss achievements, challenges, and project-related issues. Additionally, Jira, a project management tool, tracks progress in implementation and documentation tasks.

An iterative software development life cycle (SDLC) approach caters to various project factors. This approach aligns with the team's objectives, facilitating continual progress and achievement within the progress.

2.2 Project Plan

2.2.1 Software Development Life Cycle

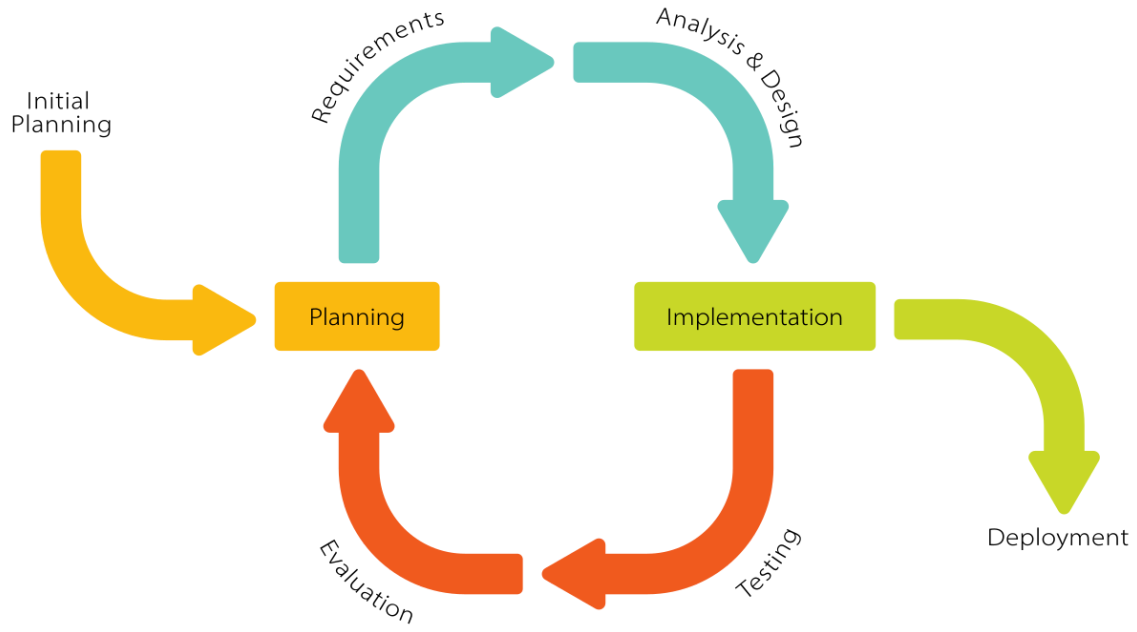


Figure 2.1: Iterative Development

We decided to use the Iterative approach carefully but surely considering our limited time and number, with high expectations and ambition relative to our resources.

Knowing that the iterative approach depends on the cycle of planning, building, testing, and refinement repeated over and over is called iterative till deployment. It's like a learn-by-doing method that keeps improving things in several loops. We are using this approach because it gives us a good amount of adaptability while reducing risk compared to other approaches.

It allows us to add new ideas to the project while improving continuously. In addition to the benefits of adding and improving the original ideas, it makes them, the latest ideas, and the whole project more compatible in each iteration.

2.2.2 Work Breakdown Structure

The work breakdown structure describes the task units to be performed by the project team.

Name	Begin date	End date	Duration
Gathering Information & initiation	3/1/24	3/6/24	6
BrainStorming	3/1/24	3/6/24	6
Professors & Students consultation	3/1/24	3/6/24	6
Internet Research	3/1/24	3/6/24	6
Planning phase	3/7/24	3/31/24	25
Develop Project Charter	3/7/24	3/10/24	4
Develop Scope Statement	3/14/24	3/15/24	2
Develop Work Breakdown Structure	3/16/24	3/19/24	4
Making Gantt Chart	3/20/24	3/21/24	2
Develop Risk Managment Plan	3/26/24	3/31/24	6
Analyses Phase	4/1/24	5/9/24	39
Requirments Elecitation	4/1/24	4/7/24	7
Determine Requirments Spacification	4/25/24	5/1/24	7
Manage Requirments	5/2/24	5/7/24	6
Analyses Documentation	5/8/24	5/9/24	2
Design Phase	6/6/24	6/18/24	13
High level architecture design	6/7/24	6/9/24	3
Create User interface flow diagram	6/10/24	6/12/24	3
Create graphical user interface	6/13/24	6/17/24	5
Other needed Diagrams	6/6/24	6/18/24	13
Implrementation Phase	6/24/24	7/12/24	19
Develop The Website	6/24/24	7/12/24	19
Publishing needed actions	6/24/24	7/12/24	19
Testing Phase	7/19/24	8/5/24	18
Establish Test Plan	7/19/24	7/22/24	4
Analyze Requirments and Test Cases	7/25/24	7/27/24	3
Design Tests	7/30/24	8/1/24	3
Implement Tests	7/30/24	8/2/24	4
Excute Tests	7/30/24	8/2/24	4
Complete Tests	8/3/24	8/5/24	3

Figure 2.2: Work Breakdown Structure

2.2.3 Gantt Chart

The Gantt chart demonstrates the flow of tasks over the project timeline and represents the dependencies and milestones in the project schedule.

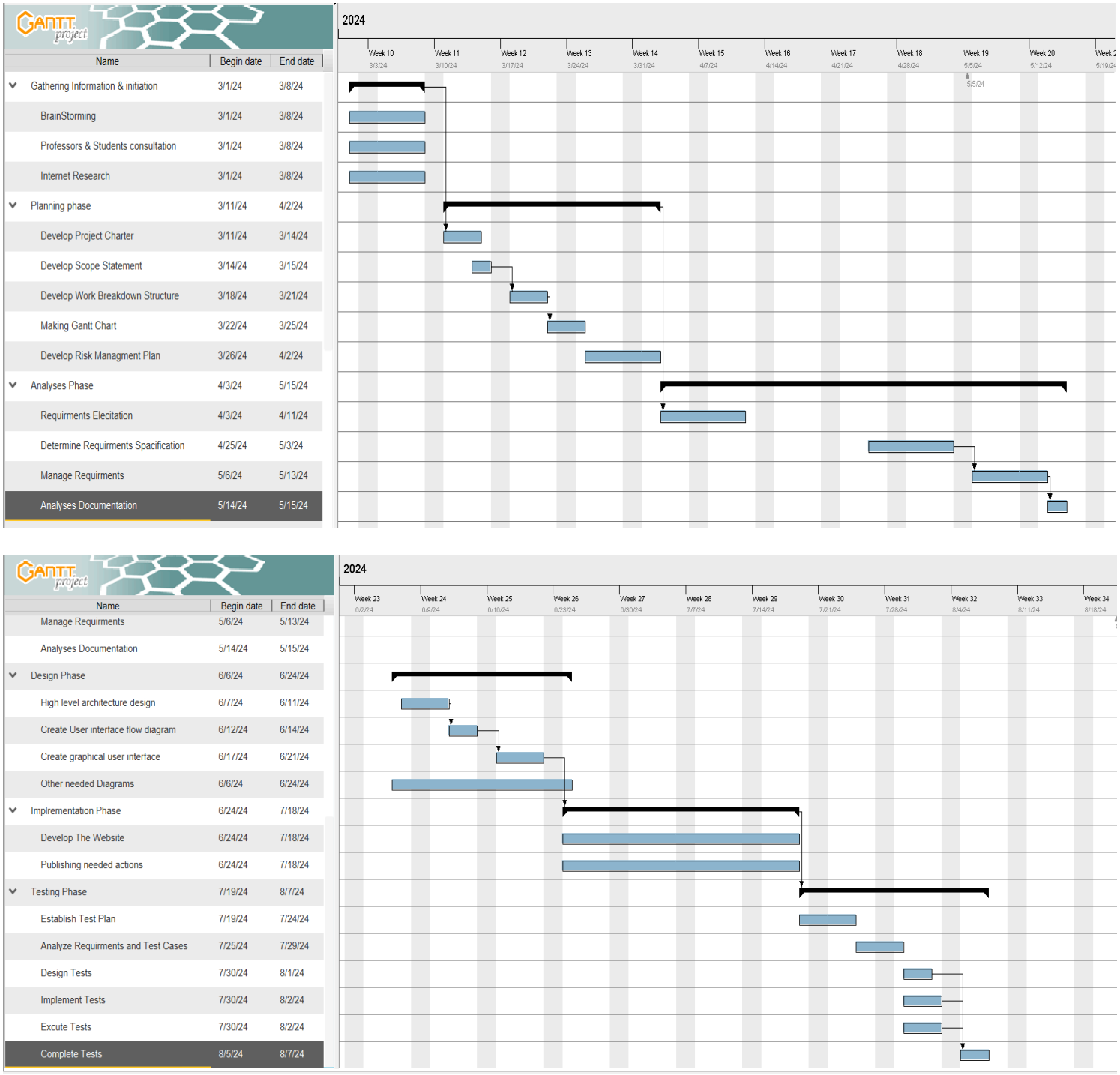


Figure 2.3: Gantt Chart

2.2.4 Tasks Allocation

Through work allocation, we identified each team member's responsibilities and level of involvement in the project's tasks. The project tasks have been allocated to the team members.

Both of us worked collaboratively on the project. We worked in person in the same place most of the time, so the effort was distributed equally between team members.

2.3 Risk Management Plan

2.3.1 Risk Source

The project may encounter risks identified by the project team and categorized as project-related or product-related. Project risks are further classified into organizational, technical, or business categories.

2.3.2 Risk Identification

Table 2.1: Risk Identification

Risk ID	Risk Description	Risk Category	Subcategory
R1	The project deadline was not met	Project	Organizational
R2	Not all team members have the needed skills in modern web technology	Project	Organizational
R3	Lack of skilled backend developers	Project	Organizational
R4	Poor documentation for certain web development tools	Project	Technical
R5	Not finding tool support for specific web technologies	Project	Technical
R6	Unable to validate new customer requirements due to external factors	Project	Business
R7	The possibility of not achieving customer satisfaction	Product	-
R8	The possibility of poor maintainability for the system	Product	-

R9	The possibility of not meeting feature requirements	Product	-
R10	Cloud server response issues	Project	Technical
R11	The project team was inexperienced with proper documentation tools	Project	Organizational
R12	Lack of plugins for specific web technologies	Project	Technical
R13	Lack of familiarity with testing tools	Project	Organizational
R14	Lack of skills for writing a proper test plan	Project	Organizational
R15	Project manager underestimation of the project cost	Project	Organizational
R16	Lack of security and security testing	Project	Technical
R17	Difficulty in attracting initial users	Project	Business
R18	Unreliable third-party integrations (payment gateways)	Project	Technical
R19	Negative feedback or reviews impacting platform reputation	Product	-
R20	Scalability issues as user base growth	Project	Technical
R21	Changes in freelancing market regulations	Project	Business
R22	Unclear project scope or objectives	Project	Organizational
R23	UI design not meeting user expectations	Product	-
R24	Potential data privacy issues impacting user trust	Project	Technical
R25	Challenges in maintaining consistent user experience	Product	-

2.3.3 Risk Analysis

Following identifying all the risks, the project team examined each from the angles of impact and probability to determine the risk exposure and priority. This information will help select the most appropriate risk response plan to address the identified risks.

Priority	Probability	Impact
High	It has a high probability of occurrence and might impact the whole project.	I cannot proceed with the project activity if it is not solved immediately.
Medium	50% probability of occurrence.	Cannot proceed with project activity if it's not solved.
Low	Low occurrence rate.	It needs to be solved, or it can be resolved later in the project or by taking another alternative.

Table 2.2: Risk Analysis

Risk ID	Probability	Impact	Priority
R1	High	High	High
R2	High	Medium	High
R3	High	Medium	High
R4	High	Medium	High
R5	Medium	Medium	Medium
R6	Medium	Medium	Medium
R7	Medium	High	High
R8	Low	High	Medium
R9	Medium	High	High
R10	Low	Medium	Low
R11	Medium	Low	Low

R12	Low	Medium	Low
R13	Low	Low	Low
R14	Low	High	High
R15	Medium	Medium	Medium
R16	High	Medium	High
R17	High	High	High
R18	High	Medium	High
R19	Medium	High	High
R20	Medium	High	High
R21	Medium	High	High
R22	High	High	High
R23	Medium	High	High
R24	High	High	High
R25	Medium	High	High

2.3.4 Risk Metric

Table 2.3: Risk Metric

Impact / Probability	Low	Medium	High
Low	R13	R11	
Medium	R12	R5 R6 R15	R2 R3 R4 R16 R18
High	R8 R14	R7 R9 R19 R20 R21 R23 R25	R1 R17 R22 R24

2.3.5 Risk Response

In this final part of the risk management plan, the team has developed countermeasures and response plans for each identified risk. These response plans provide a clear vision of the actions that need to be taken to address these risks effectively and prepare for any potential consequences. By proactively implementing these response plans, DevHive aims to mitigate the impact of risks to ensure the successful execution of the project.

Table 2.4: Risk Response

Risk ID	Mitigation Action	Person In Charge (PIC)
R1	Develop an efficient project schedule to ensure timely delivery.	Project Manager (Mitigation)
R2	Conduct online courses and self-study to enhance web development skills.	Project Team (Mitigation)
R3	Recruit additional backend developers or provide training to existing team members.	Project Manager (Mitigation)
R4	Establish clear documentation standards and ensure documentation is regularly updated.	Project Manager/ Technical Lead (Mitigation)
R5	Research alternative tools or develop custom solutions to support specific technologies.	Development Team/ Technical Lead (Mitigation)
R6	Implement iterative methodologies to adapt to changing customer requirements quickly.	Project Manager (Mitigation)
R7	Implement a comprehensive quality assurance process and gather regular feedback from clients.	Project Manager/ QA Team (Mitigation)
R8	Prioritize maintainability in the development process and conduct regular code reviews.	Development Team/ Technical Lead (Mitigation)
R9	Conduct regular feature requirement reviews and ensure alignment with client expectations.	Project Manager (Mitigation)
R10	Monitor cloud server performance and optimize configurations as needed.	Project Manager (Mitigation)
R11	Provide training on documentation tools and establish documentation best practices.	Project Manager (Mitigation)

R12	Explore available plugins or develop custom solutions to address technology gaps.	Development Team/ Technical Lead (Mitigation)
R13	Conduct training sessions on testing tools and methodologies.	Project Manager/ QA Team (Mitigation)
R14	Develop a comprehensive test plan and ensure its implementation throughout the project lifecycle.	Project Manager/ QA Team (Mitigation)
R15	Conduct a thorough project cost estimation and ensure accurate budget allocation.	Project Manager (Mitigation)
R16	Conduct security assessments and implement necessary security measures.	Technical Lead (Mitigation)
R17	Implement targeted marketing strategies to attract initial users and clients.	Product Manager (Mitigation)
R18	Thoroughly evaluate third-party integrations and ensure reliability and compatibility.	Development Team/ Technical Lead (Mitigation)
R19	Implement reputation management strategies and address negative feedback promptly.	Project Manager (Mitigation)
R20	Implement scalable architecture and regularly assess performance as the user base grows.	Development Team/ Technical Lead (Mitigation)
R21	Stay informed about regulatory changes and adapt the platform accordingly.	Product Manager (Mitigation)
R22	Clarify project scope and objectives through regular communication and documentation.	Project Manager/ Technical Lead (Mitigation)
R23	Gather user feedback and iterate on UI design to meet user expectations.	UX/UI Designer/Development Team (Mitigation)
R24	Implement data privacy measures and comply with relevant regulations.	Technical Lead (Mitigation)
R25	Conduct regular usability testing and address any issues affecting user experience.	UX/UI Designer/Development Team (Mitigation)

Chapter 3

Software Requirements Specification

3.1 User Classes and Characteristics

Although stakeholders play a significant role through their interactions, focusing on each individually enables us to address their needs better and understand how each may affect or be affected by the software product.

The intended users are classified into different roles related to core responsibilities and characterized based on their influence on this project:

- **Supervisor:** reviews the written documentation to ensure it meets all required criteria.

- **Project Manager:** manages various aspects of the project, monitors progress, and stays updated through different communication channels.

- **Development Manager:** implements various features to meet the needs of the software product's end users.

- **Client (Business Owner):** uses DevHive to manage projects, track progress, and analyze data.

- **Freelancer (Developer):** uses DevHive to find and manage projects, negotiate rates, and showcase their skills.

- **Customer:** uses DevHive to hire developers, post projects, and track progress.

The Power-Interest grid helps determine the appropriate method for handling each type of user class. Some users are monitored because they have little interest in the project and minimal impact on it. A set of users are grouped to inform them where they are interested in the project but have little to no effect, and the other group must be kept satisfied. Although they have low interest, they can greatly impact the project.

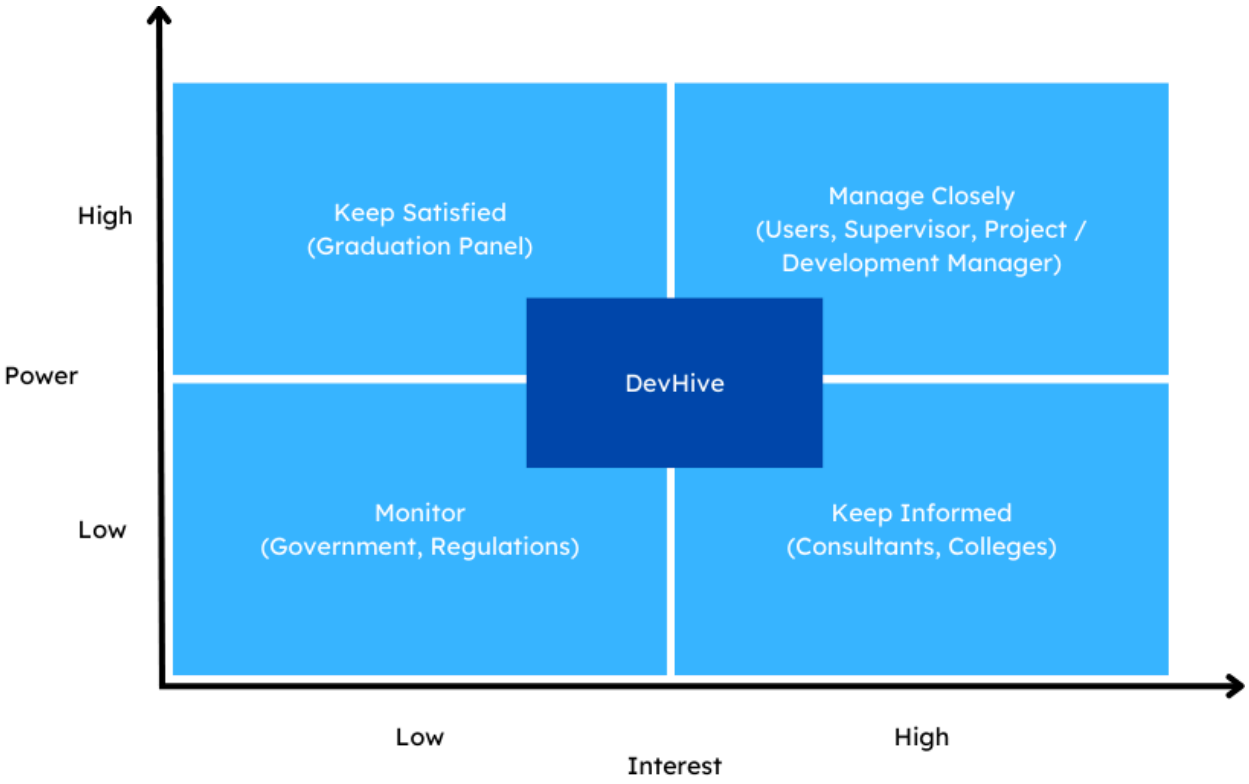


Figure 3.1: Power/Interest Grid

3.2 Product Scope

The product scope for DevHive involves creating a user-friendly freelancing website tailored for developers and clients. The platform will have a consistent interface for all users. Developers can discover and manage projects, negotiate rates, and showcase their skills. Clients can post projects, manage all project aspects, and track progress.

3.3 Product Requirements

This section outlines the requirements engineering process (RE) for DevHive, which involves eliciting and analyzing requirements using modern techniques. This process will help us identify user and system requirements and categorize them appropriately to define the product's operation.

3.3.1 Requirements Elicitation

We used various modern requirements elicitation techniques to gather the user requirements and better understand our users' needs.

3.3.1.1 Personas

Personas were designed to help us better understand the target segment, narrow the search for people to interview, and predict who will use our product.

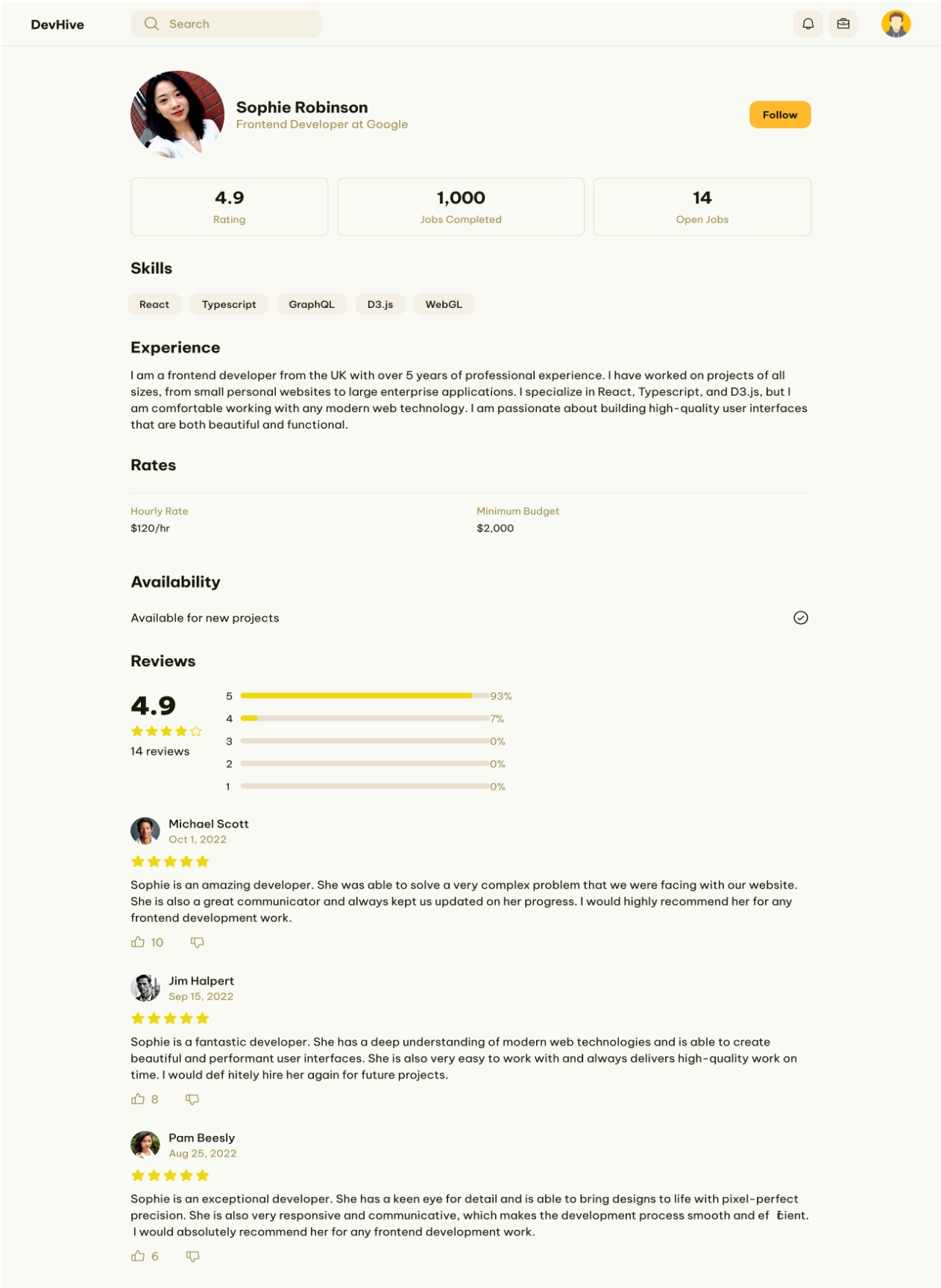


Figure 3.2: Persona 1

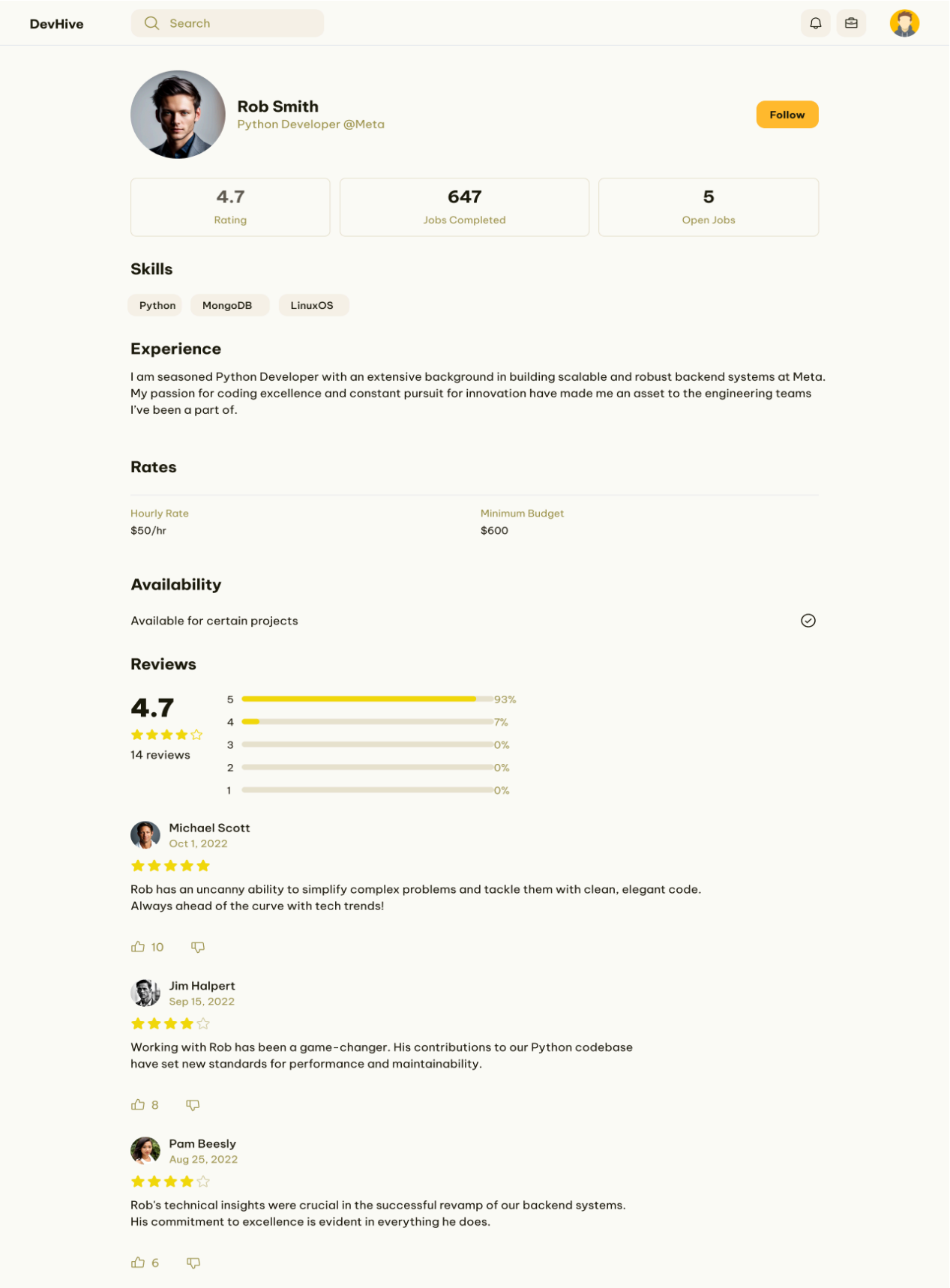


Figure 3.3: Persona 2

3.3.1.2 Interviews

Throughout the project, the team members participated in recurring reviews with the supervisor to assess their progress. The meetings covered completed work, upcoming milestones, and what to improve and focus on.

3.3.1.3 Survey

On May 5th, 2024, a questionnaire was sent to a demographic of developers to gather requirements about DevHive. The aim was to collect information from people who are interested in such a system and how they will incorporate it into their lives. It also aimed to gather information about how satisfied these people are with their current income and how interested they are with the features DevHive is going to provide. The questionnaire was filled out by developers, most of whom are IT students from Princess Sumaya University for Technology.

3.3.1.4 Brainstorming Sessions

Brainstorming sessions were carried out between team members throughout the project's lifecycle. These sessions proved crucial in collecting our thoughts and discussing new ideas and how to approach them effectively. They were essential in shaping the structure of what we are trying to build and provide for our user base.

3.3.1.5 Document Analysis

At the beginning of the project, we conducted a document analysis to study and review similar existing projects. This helped us understand DevHive's specifications, find room for enhancements in existing solutions, and learn how to incorporate these enhancements within the system.

3.3.1.6 Use Case

Using use case diagrams is an effective way to gather requirements, as it illustrates features for each user. The first type of user, the form maker, can interact with the software through various actions such as form creation, form customization, form submission management, and multilingual functionality, among other things. This flowchart visually represents how the user can use our form builder.

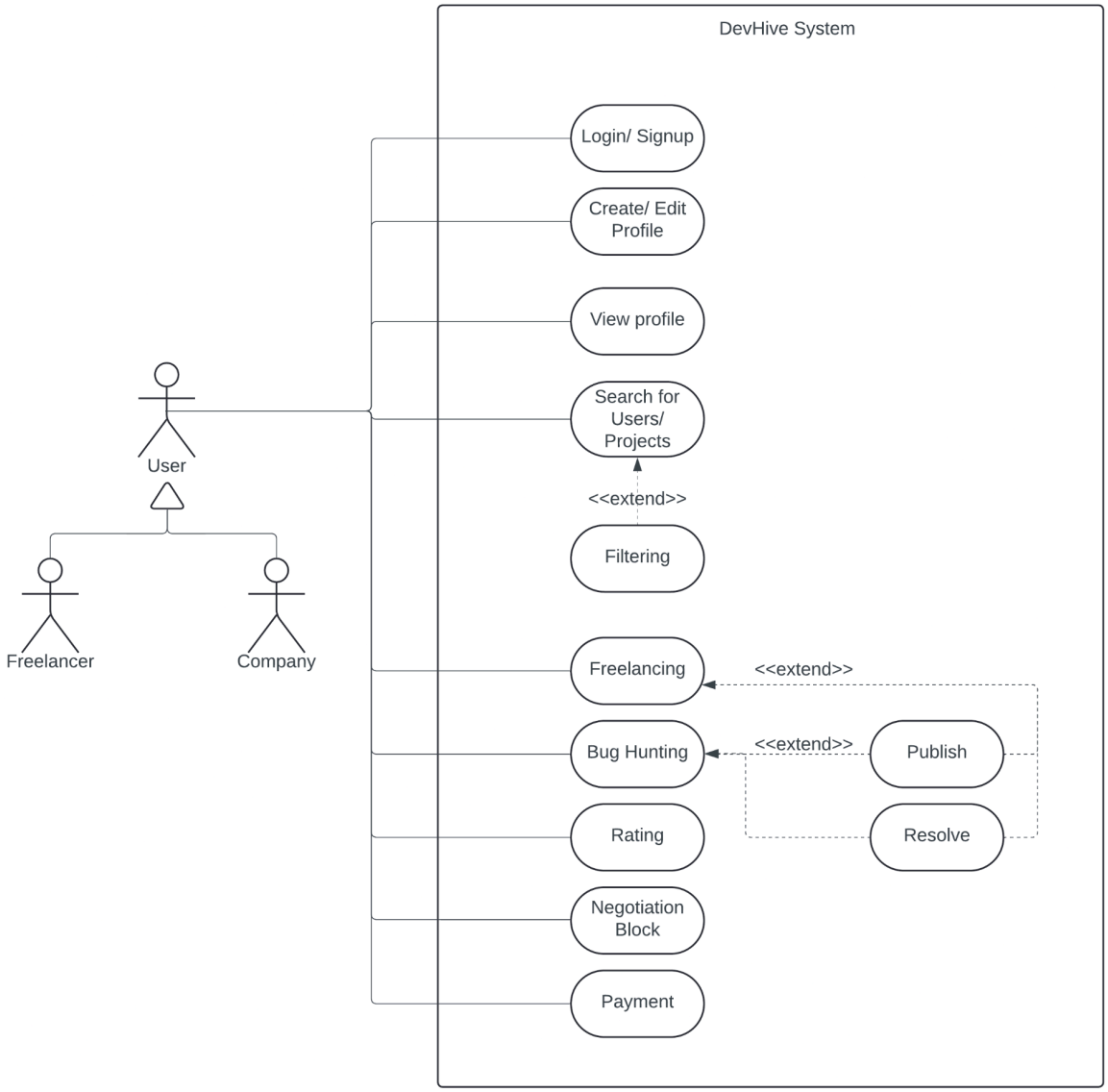


Figure 3.4: Use Case

Name	Login/Signup
Description	This use case covers the login/device 'sup process, which involves browsing the website freely to view DevHive's services and being prompted based on each party's setup whenever they try to use a service.
Actors	User (Freelancer/ Company)
Pre-conditions	The user must have access to the website.
Post-conditions	The user can sign/log into the website.
Flow of Control	1. Users browse the website 2. Users insert email and password when prompted 3. Users verify their email
Alternatives	None

Name	Create profile
Description	After registering their accounts, a window prompts users to insert personal information, work experience, skills, and interests to create their profiles.
Actors	User (Freelancer/ Company)
Pre-conditions	Login/Signup
Post-conditions	Users can now use our services.
Flow of Control	1. After signing up, users must enter their personal information 2. The information provided will be added to their profile 3. Users can interact with job offers and bug-hunting events
Alternatives	After creating their profiles, users can edit them easily if any provided information needs to be corrected.

Name	Freelancing
Description	After creating a profile, users can access the freelancing service, where they can find offers from both individuals and companies.
Actors	User (Freelancer/ Company)
Pre-conditions	Create profile
Post-conditions	Users take freelancing jobs and offer their services to others on the platform for monetary rewards.
Flow of Control	1. Users navigate the freelancing section 2. Users choose the jobs that match their skills and interests.
Alternatives	None

Name	Bug Hunting
Description	This is a feature where users try to fix bug-infested code. You can work as individuals or groups and be rewarded.
Actors	User (Freelancer/ Company)
Pre-conditions	Create profile
Post-conditions	Users join bug-hunting events and offer their services to others on the platform for monetary rewards.
Flow of Control	1. Users navigate the bug-hunting section 2. Users try to solve the bug as individuals or groups
Alternatives	None

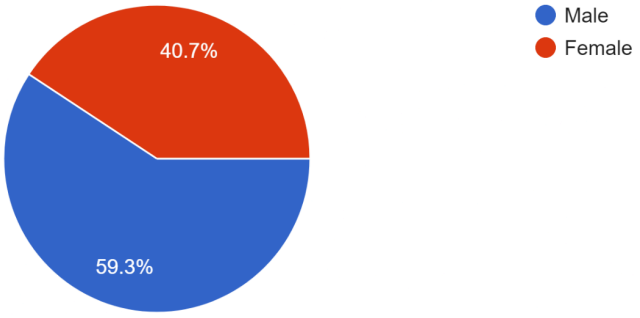
Name	Negotiation Block
Description	This use case covers one of the main features of DevHive, the negotiation block, which allows developers and clients to negotiate the payment and terms depending on different points of view.
Actors	User (Freelancer/ Company)
Pre-conditions	Freelancing
Post-conditions	Users agree upon a price that satisfies both parties.
Flow of Control	1. Users apply for a job offer 2. If they were selected (based on their rating and experience), a negotiation phase commences 3. Both parties negotiate back and forth until they reach an agreement.
Alternatives	None

Name	Search for Users/ Projects
Description	This use case covers a basic but necessary feature in any website, where users can search for other users on the platform or for projects they are interested in.
Actors	User (Freelancer/ Company)
Pre-conditions	Login/ Signup
Post-conditions	Users are presented with the results of what they searched for.
Flow of Control	1. Users navigate to the search button. 2. They search for other users or potential projects. 3. The search results appear in a list.
Alternatives	None

3.3.1.7 Findings

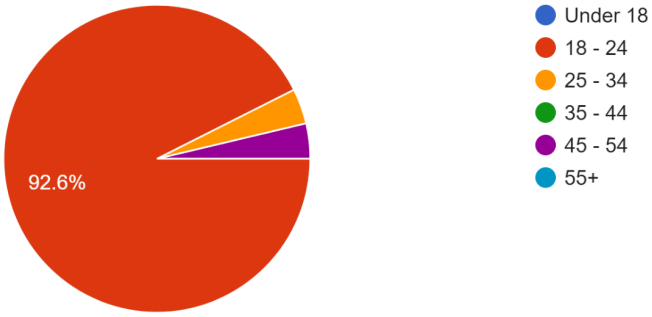
What is your Gender?

27 responses



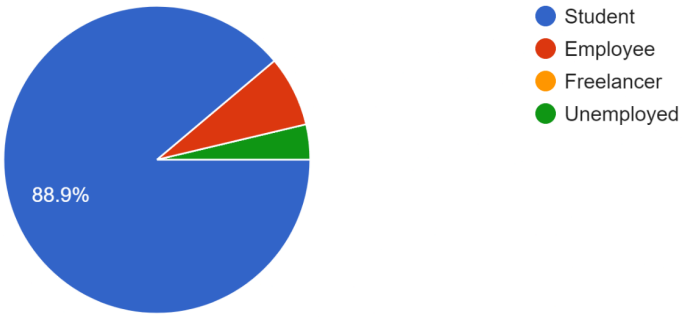
Age

27 responses



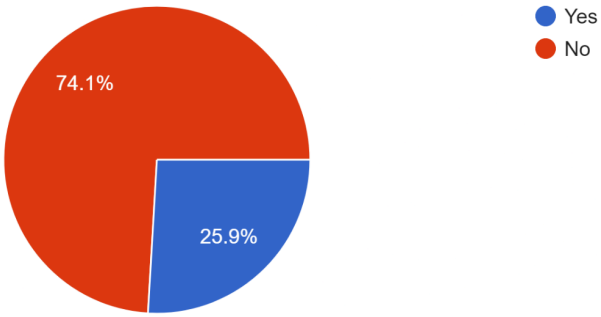
What is your profession?

27 responses



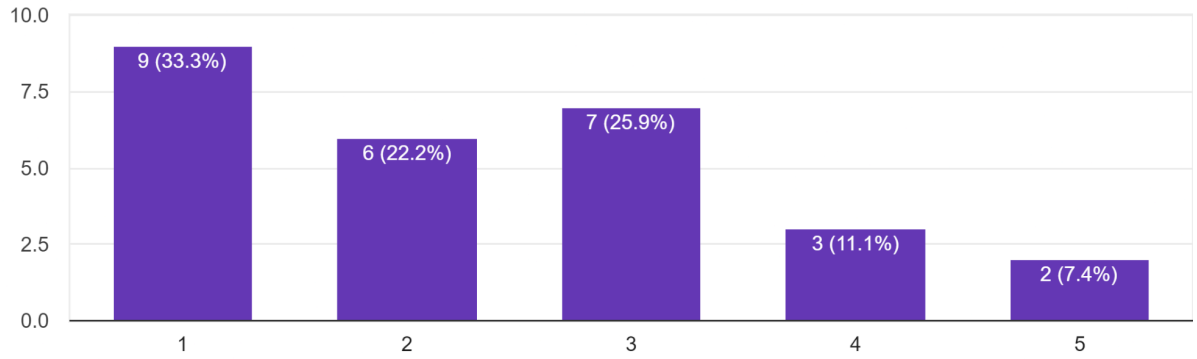
Have you ever used a freelancing platform?

27 responses



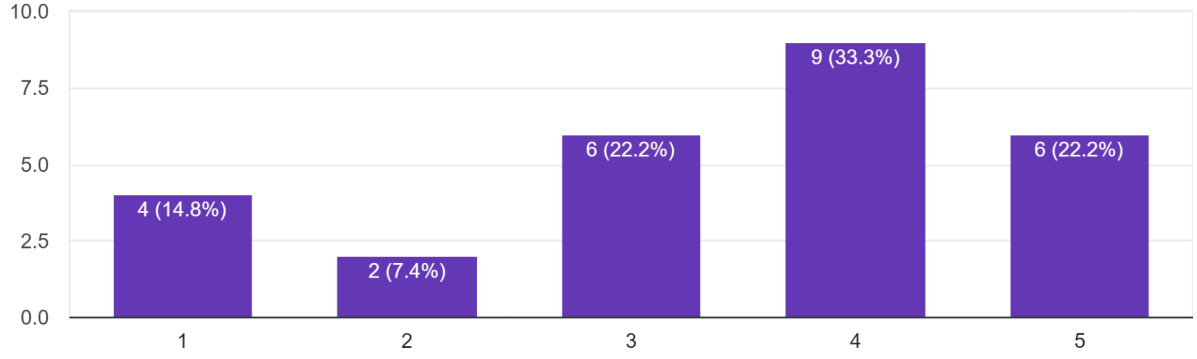
How satisfied are you with your current income?

27 responses

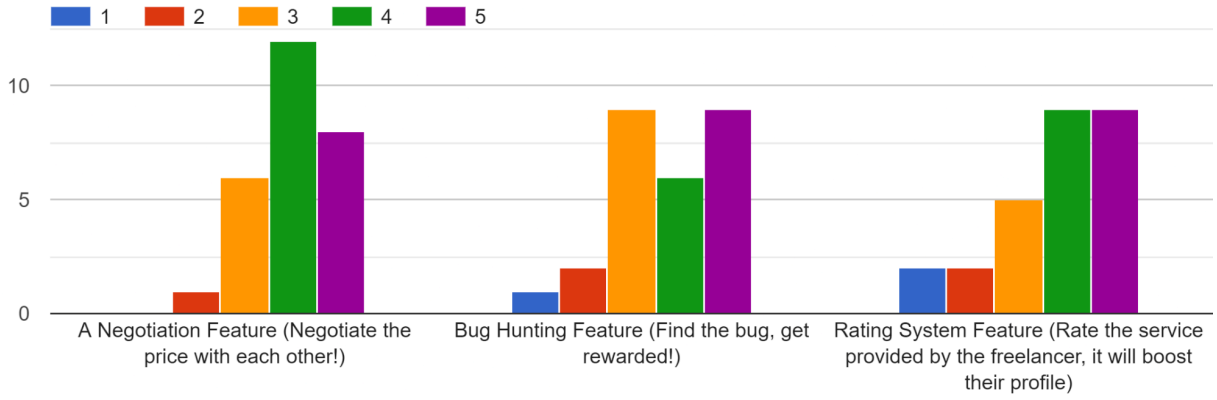


How likely are you going to use DevHive alongside your current profession?

27 responses



On scale from 1 to 5, how interested are you in the following features?



3.4 System Requirements

3.4.1 Functional Requirements

3.4.1.1 User Registration

1) **Email-Based Registration:**
    - Users must be able to register using their email address by providing a username, email, and password.
    - Passwords must be at least 8 characters long and include a combination of letters, numbers, and special characters.
    - Users must provide a copy of their National ID card to ensure secure identities.
2) **Profile Creation:**
    - After registration, users must be able to create a basic profile with details such as their name, skills, and brief bio.

3.4.1.2 Profile Management

1) **Basic Information Update:**
    - Users must be able to update their profile information, including their name, bio, skills, and contact information.
2) **Profile Picture:**
    - Users must be able to update their profile pictures to personalize their accounts.

3.4.1.3 Post Management

1) **Post Jobs:**
    - Users must be able to post jobs, add job requirements, and include payment for projects they post.
2) **Save Jobs:**
    - Users must be able to save job postings that they find interesting for future use (if they are still available).

3) **Post Authorization:**

    - Users can not post or apply for jobs unless they are logged in.

3.4.2 Non-Functional Requirements

3.4.2.1 Usability

1) **User Interface (UI):**

    - The UI must be intuitive, easy to navigate, and consistent design across all pages.

2) **Responsiveness:**

    - The website must be fully responsive, providing an optimal experience for all device types including desktops, tablets, and mobile phones.

3.4.2.2 Maintainability

1) **Documentation**

    - Comprehensive documentation should be provided for both developers and end-users, including API documentation, user guides, and a troubleshooting section.

2) **Version Control:**

    - All code changes should be put to some form of version control (Git) in almost all cases, under the condition that all messages within each commit and the branching strategies for different Java versions do not begin to diverge off to several features.

3.5 Requirements Management

3.5.1 Requirements Interdependency

Interdependency was used to understand the dependencies between the requirements & to understand their relationships clearly.

Table 3.1: Requirements Interdependency

Req. ID	Refines To	Changes To	Similar To	Requires	Conflict with	Increase Cost Of	Increases Value Of	Decrease Cost Of	Decrease Value Of
FR1							UR1		
FR2	FR1		FR1	NFR5	NFR2		UR2		
FR3	FR2		FR2	FR1			UR3		
FR4	FR1			FR1			UR4		
FR5							UR5		
FR6							UR6		
FR7				FR5			UR7		
FR8			FR6, FR7	FR5, FR7			UR8		
FR9				FR5			UR9		
FR10			FR8, FR9	FR8, FR9			UR10		
FR11				FR5, FR6			UR11		
FR12							UR12		
FR13				FR12	NFR2	NFR2	UR13		
FR14				FR12			UR14		
FR15							UR15		
FR16				FR15			UR15, UR16		

FR17							UR17		
FR18				FR17			UR18		
FR19							UR19		
FR20				FR19			UR20		
FR21				FR19			UR21		
FR22				FR19, FR21			UR22		
FR23							UR23		
FR24				FR23			UR24		
NFR1									
NFR2					FR13				
NFR3									
NFR4									
NFR5					FR2				FR1
NFR6				NFR5					
NFR7									
NFR8				FR23,FR24 ,FR25				UR23,UR2 4, UR25	
NFR9									
NFR1 0									
NFR1 1									

3.5.2 Requirements Prioritization

Requirement prioritization prioritizes requirements based on their importance to the stakeholders.

The team also considered the dependencies between requirements since they provide knowledge on how requirements rely on each other, which can affect their priorities. The MoSCoW prioritization technique, a widely accepted method, was used to prioritize the requirements based on several characteristics, which are:

- **Mo: Most Vital**
- **S: Important but not vital**
- **Co: Nice to have**
- **W: Things that provide little to no value.**

Table 3.2: Risk Prioritization

Requirement ID	MoSCoW ID
FR1	Mo
FR2	Mo
FR3	Mo
FR4	Mo
FR5	Mo
FR6	Mo
FR7	Mo
FR8	Mo
FR9	Mo
FR10	Mo
FR11	Mo
FR12	S

FR13	S
FR14	S
FR15	S
FR16	S
FR17	S
FR18	S
FR19	Co
FR20	Co
FR21	Co
FR22	Co
FR23	Co
FR24	Co
NFR1	Mo
NFR2	Mo
NFR3	Mo
NFR4	Mo
NFR5	Mo
NFR6	S
NFR7	S
NFR8	S
NFR9	Co
NFR10	Co
NFR11	Co

Chapter 4

Software Design Description

4.1 Context Diagram

The Context Diagram was used to identify significant functionalities in the system and gather system requirements.

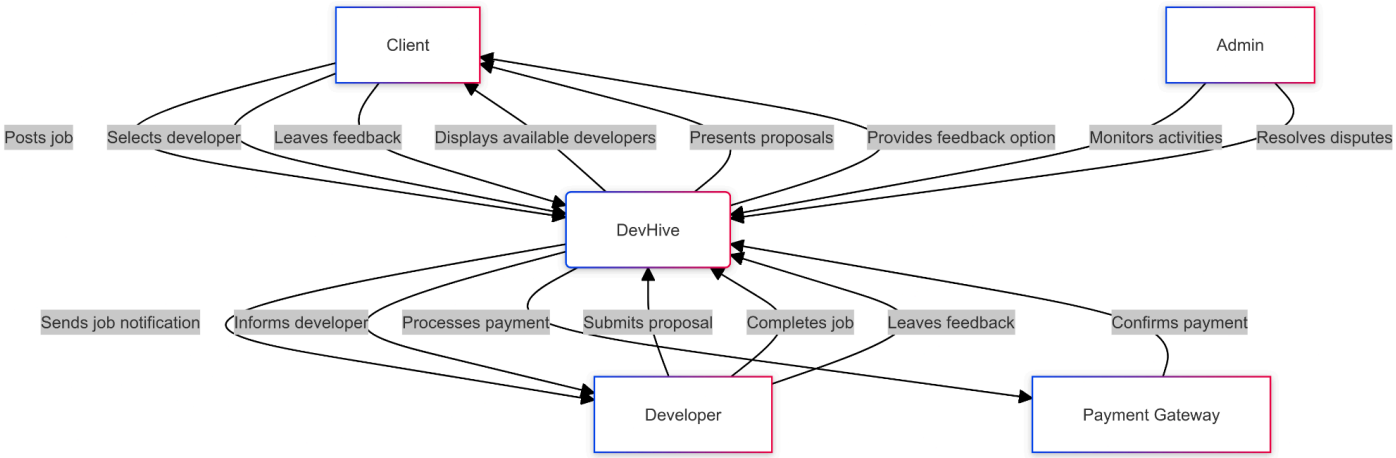


Figure 4.1: Cotext Diagram

4.2 Class Diagram

A class diagram describes the classes that the software product will consist of. It outlines the relationships between these classes and how they interact with each other. The specified class diagram is a "Domain Model Diagram," which represents the model classes found in the application.

This is an initial class diagram that will help us in GP2. It will be updated and improved in the future:

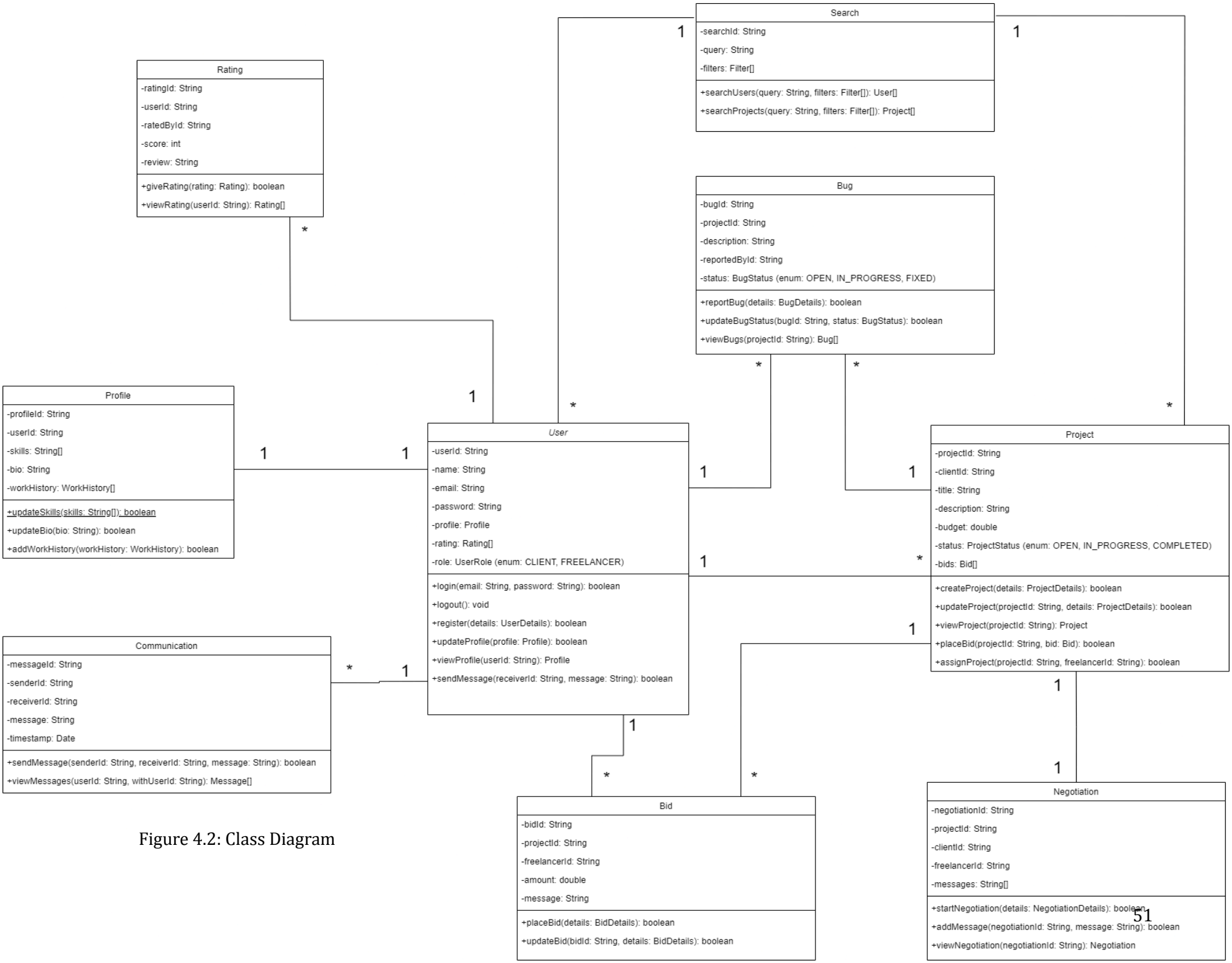


Figure 4.2: Class Diagram

4.3 Sequence Diagram

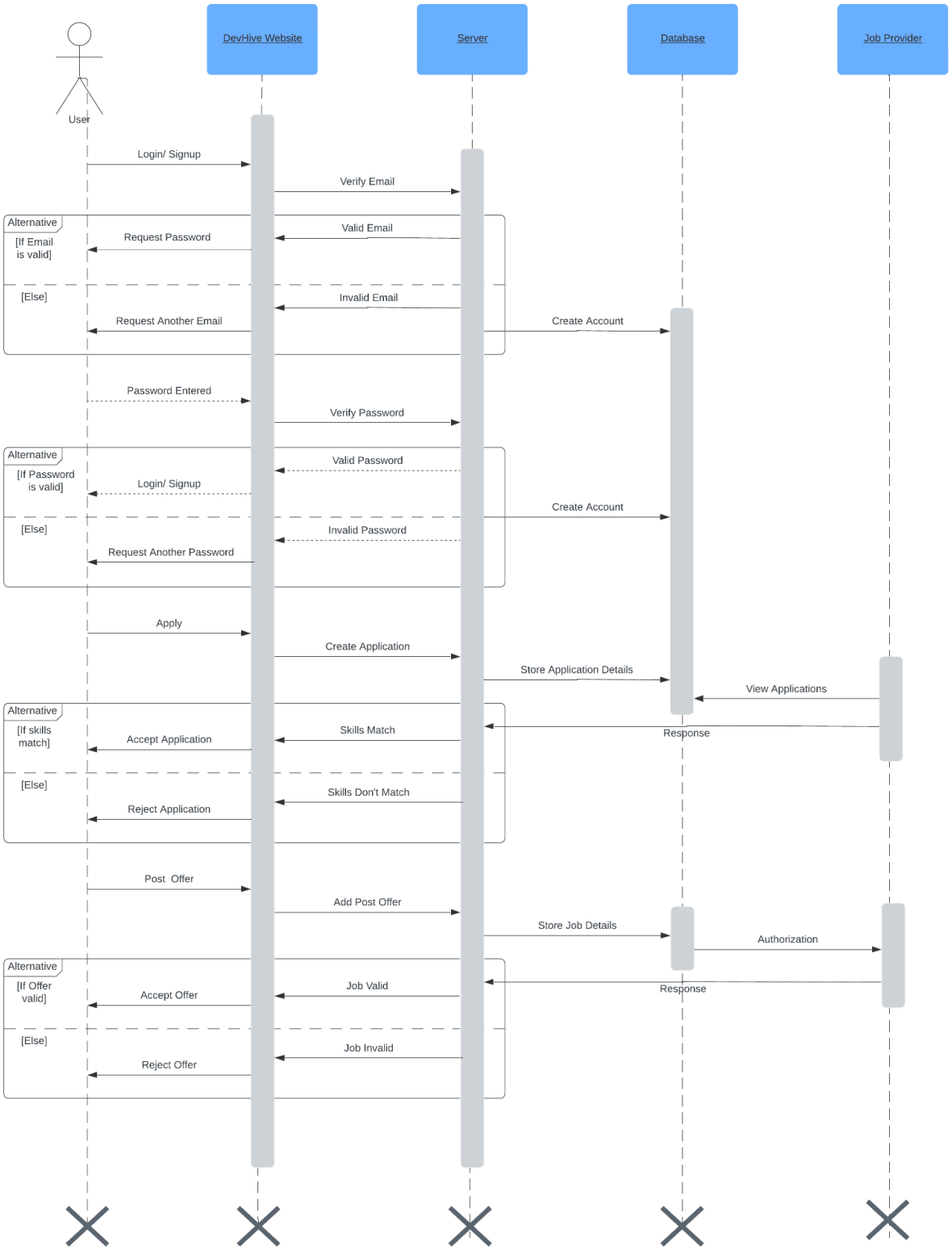


Figure 4.3: User Sequence Diagram

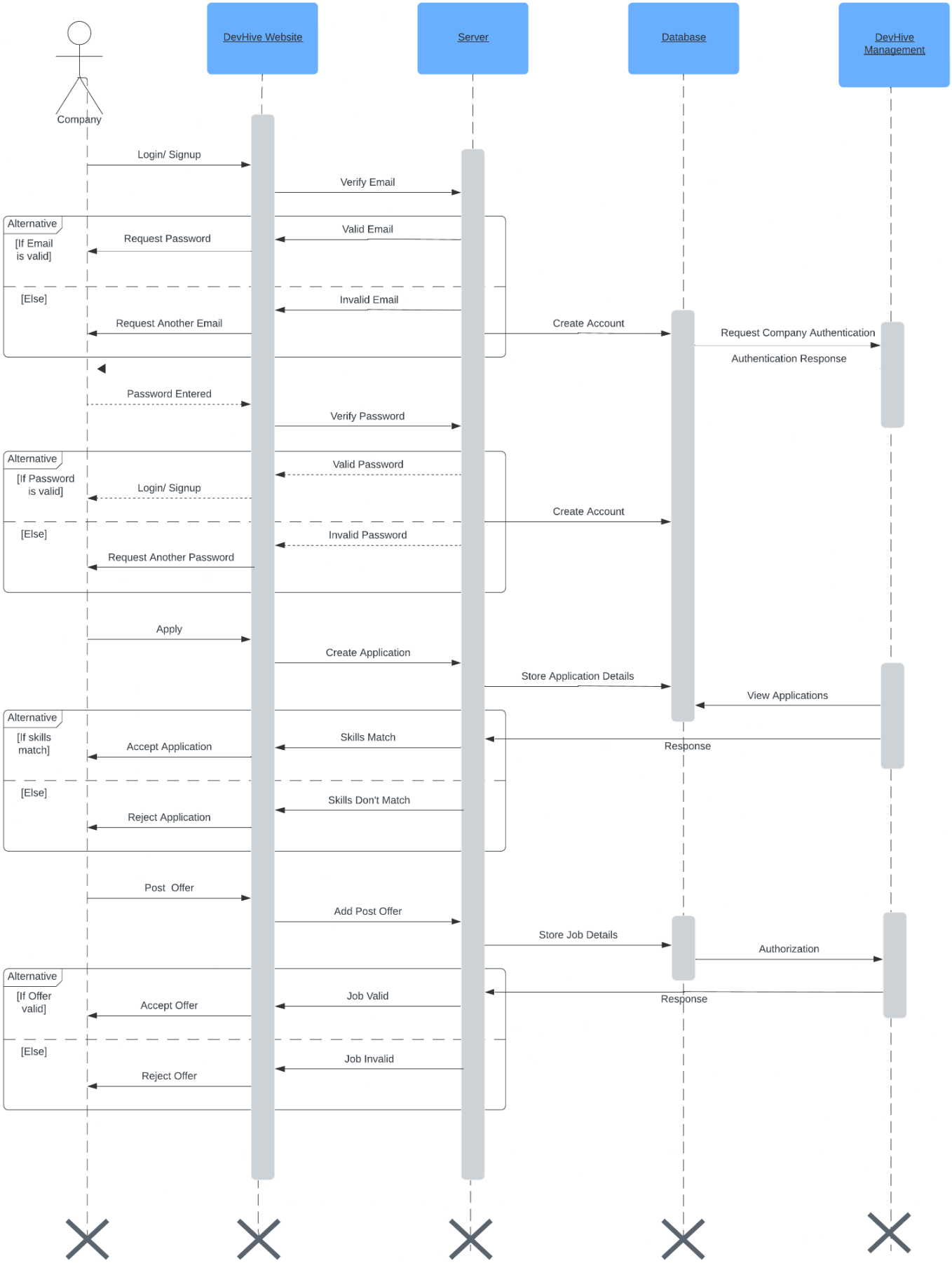


Figure 4.4: Company Sequence Diagram

Chapter 5

Software Implementation

5.1 Completed and Implemented Requirements

The project development team worked very hard to cover and finish as many requirements as possible, through various means of communication, the project team with the

stakeholders involved accepted the delivered functionalities in the production software. These functionalities cover the main scenarios of the following:

- **Log In:** Users log into the platform with pre-registered credentials securely.
- **Sign Up:** A new user signs up by providing the details of email, password, and basic profile information.
- **Profile Management:** Modify personal information like name, bio, and skills.
- **View Listing:** Users would be able to view various listings, which could be projects, posts, or profiles depending on the focus of the website.
- **Content Interacting:** A user can create a post, reply to it, or discuss it.
- **Rate and Review:** A user shall have the facility of rating and reviewing certain content or another user.
- **Manage Appointments:** The user manages appointments, sessions, or any other activity
- **Log-Out:** The user logs out safely from the platform.

5.2 Programming Languages and Technologies Used

5.2.1 Frontend

The project's frontend is built using modern web technologies to ensure a responsive and interactive user experience. The team utilized React for building the user interface, coupled with React Router for efficient navigation and seamless transitions between pages. For API communication Axios was used to ensure smooth communication with the backend. Moreover, the styling was managed with SCSS, which offers advanced features like variables and nested rules, while Bootstrap is integrated to create a responsive and multi-resolution-friendly design. The build is optimized using Vite, which provides a fast development server and efficient bundling for production.

5.2.2 Backend

The backend of the project is designed to provide a scalable and efficient server-side infrastructure using modern web development technologies. Node.js serves as the runtime environment, which enables JavaScript to be used for server-side scripting. Express.js is the web application framework utilized to easily build the RESTful API, handling routing, middleware, and HTTP requests. MongoDB is a NoSQL database that was used for data storage in this project, it provides flexibility and scalability in managing user data and application content. Mongoose, which is an ODM (Object Data Modeling) library, is employed to facilitate the interaction with MongoDB, offering schema-based data validation and querying capabilities. Security is reinforced with JSON Web Tokens (JWT) for authentication and authorization, ensuring secure access to the application's resources. Additionally, Bcrypt.js is used for hashing passwords, adding an extra layer of security to user credentials.

5.2.3 Design Patterns

1) **Component-Based Design:** In this design pattern, the whole code in each file is put up within a React functional component as shown in 'ListPage' here for instance. This methodology aligns with the component-based design pattern, which involves encapsulating UI elements and their functionalities in discrete reusable parts.

```
import Filter from "../../components/filter/Filter";
import { listData } from "../../lib/dummydata";
import "./listPage.scss";
import Card from "../../components/card/Card";
import Map from "../../components/map/map.jsx";

function ListPage() {
  const data = listData;
  return (
    <div className="listPage">
      <div className="listContainer">
        <div className="wrapper">
          <Filter />
          {data.map((item) => (
            <Card key={item.id} item={item} />
          ))}
        </div>
      </div>
      <div className="mapContainer">
        <Map items={data} />
      </div>
    </div>
  );
}

export default ListPage;
```

Figure 5.1: ListPage Component

2) **Router Pattern:** This design pattern organizes navigation and routing within a React application. It allows the declaration of routes that map URL paths to specific components.

```jsx
import { useContext } from "react";
import SearchBar from "../../components/searchBar/SearchBar";
import { AuthContext } from "../../context/AuthContext";
import "./homePage.scss";

function HomePage() {

  const {currentUser} = useContext(AuthContext);

  console.log(currentUser)
  return (
    <div className="homePage">
      <div className="textContainer">
        <div className="wrapper">
          <h1 className="title">From Developers, To Developers</h1>
          <p>
          DevHive is Jordans premier freelancing platform for developers. We
            connect talented developers with exciting projects, providing a
            space to showcase skills, collaborate, and grow. Join us today to
            advance your freelancing career and be a part of Jordans thriving
            tech community.
          </p>
          <SearchBar />
          <div className="boxes">
            <div className="box">
              <h1>16+</h1>
              <h2>Years of Experience</h2>
            </div>
            <div className="box">
              <h1>200</h1>
              <h2>Award Gained</h2>
            </div>
            <div className="box">
              <h1>2000+</h1>
              <h2>Job Opportunities</h2>
            </div>
          </div>
        </div>
      </div>
      <div className="imgContainer">
        <img src="/bg.png" alt="" />
      </div>
    </div>
  );
}

export default HomePage;
```

Figure 5.2: HomePage Router Design

3) **Layout Pattern:** The layout pattern enables consistency in layout across several pages of an application. Normally, it contains the common UI elements, including headers, footers, and navigation that are shared by different routes. The 'Layout' component is used as a wrapper around routes like '/', '/list', and '/register', ensuring that these routes share a common layout.

```
import { useContext } from "react";
import { Navigate, Outlet } from "react-router-dom";
import Navbar from "../../components/navbar/Navbar";
import { AuthContext } from "../../context/AuthContext";
import "./layout.scss";

function Layout() {
  return (
    <div className="layout">
      <div className="navbar">
        <Navbar />
      </div>
      <div className="content">
        <Outlet />
      </div>
    </div>
  );
}
```

Figure 5.3: Layout Pattern

4) **Protected Route (Authorization) Pattern:** This pattern is used to restrict access to certain routes or components unless specific conditions (like user authentication) are met.

   A good use of the 'RequireAuth' component is to wrap certain routes or views that

are to be guarded. In this case, the routes,/profile' and '/profile/update' are protected, meaning the user has to be authenticated to view the version needed.

```
const navigate = useNavigate();
 const handleSubmit = async (e) => {
   e.preventDefault();
   const formData = new FormData(e.target);

   const { username, email, password } = Object.fromEntries(formData);

   try {
     const res = await apiRequest.put(`/users/${currentUser.id}`, {
       username,
       email,
       password,
       avatar,
     });

     updateUser(res.data)
     navigate("/profile");
   } catch (err) {
     console.log(err);
     setError(err.response.data.message);
   }
 };
```

Figure 5.4: Protected Route Pattern

5.2.4 Implementation Details

1) **Project Structure and Setup**
    a) **Technology Stack:**

i) *Frontend:* React, JacaScript XML, SCSS.

ii) *Routing:* react-router-dom for client-side routing.

b) Project Structure:

i) *'DevHive/':* Root directory of the project

(1) *'.git/':* Contains Git version control files.

(2) *'package.json':* Lists project dependencies and scripts.

(3) *'package-lock.json':* Locks dependency versions.

(4) *'src/':* Source code directory.

2) Routing Implementation

a) Router Setup:

i) Use 'react-router-dom' to define client-side routes in the application.

ii) Routes are divided into public and protected routes:

(1) Public Routes:

(a) *'/' (HomePage):* The landing page of the application.

(b) *'/list' (ListPAge):* Displays a list of items, such as job listings.

(c) *'/:id' (SinglePage):* Shows detailed information about a specific item.

(d) *'/login' (Login):* Allows users to log in to their accounts.

(e) *'/register' (Register):* Allows new users to create an account.

(2) Protected Routes:

(a) *'/profile' (ProfilePage):* Displays the user's profile, accessible only to authenticated users.

(b) *'/profile/update' (ProfileUpdatePage):* Allows users to update their profile information.

iii) Component Structure:

(1) *Layout:* Common layout shared across various pages, ensuring consistent UI components like headers and footers.

(2) *RequireAuth:* A higher-order component that protects certain routes, requiring users to be authenticated before granting access.

3) Component Implementation:

a) Main Components:

i) *HomePage:* the main landing page, providing an overview of the platform and easy navigation to other sections.

```
import { useContext } from "react";
import SearchBar from "../../components/searchBar/SearchBar";
import { AuthContext } from "../../context/AuthContext";
import "./homePage.scss";

function HomePage() {
  const { currentUser } = useContext(AuthContext);

  console.log(currentUser);
  return (
    <div className="homePage">
      <div className="textContainer">
        <div className="wrapper">
          <h1 className="title">From Developers, To Developers</h1>
          <p>
            DevHive is Jordan's premier freelancing platform for developers. We
            connect talented developers with exciting projects, providing a
            space to showcase skills, collaborate, and grow. Join us today to
            advance your freelancing career and be a part of Jordan's thriving
            tech community.
          </p>
          <SearchBar />
          <div className="boxes">
            <div className="box">
              <h1>16+</h1>
              <h2>Years of Experience</h2>
            </div>
            <div className="box">
              <h1>200</h1>
              <h2>Award Gained</h2>
            </div>
            <div className="box">
              <h1>2000+</h1>
              <h2>Job Opportunities</h2>
            </div>
          </div>
        </div>
      </div>
      <div className="imgContainer">
        <img src="/bg.png" alt="" />
```

```
      </div>
    </div>
  );
}

export default HomePage;
```

ii) *ListPage:* Displays a list of available job opportunities or other relevant items, with filters and search functionality

```
import Filter from "../../components/filter/Filter";
import { listData } from "../../lib/dummydata";
import "./listPage.scss";
import Card from "../../components/card/Card";
import Map from "../../components/map/map.jsx";

function ListPage() {
  const data = listData;
  return (
    <div className="listPage">
      <div className="listContainer">
        <div className="wrapper">
          <Filter />
          {data.map((item) => (
            <Card key={item.id} item={item} />
          ))}
        </div>
      </div>
      <div className="mapContainer">
        <Map items={data} />
      </div>
    </div>
  );
}

export default ListPage;
```

iii) *SinglePage:* Shows detailed information for a specific item selected from the ListPage. //Must update the code first

iv) *Login:* Provides user authentication with forms for email and password input. //Check with Alaa about the code

v) *ProfilePage:* Shows user-specific information like name, email, and other profile details. //Check with Alaa about the code

vi) *ProfileUpdatePage:* A form allowing users to update their profile information. //Check with Alaa about the code

vii) *Register:* User registration page, capturing details required for creating a new account. //Check with Alaa about the code

4) Styling and Theming

a) SCSS Implementation:

i) Use global styles defined in 'index.scss' for base styling and 'responsive.scss' for responsive design.

ii) Utilize Bootstrap for pre-built responsive components and utility classes.

iii) Custom styling should achieve a consistent look and feel across all components.

5) State Management

a) State and Context:

i) Use React's 'useState' and 'useContext' for local and global state management.

ii) Global states such as user authentication status could be managed through a context provider, making it accessible across different components.

6) Package Management and Build

a) Package Management:

i) Manage dependencies using 'package.json'.

ii) Install required packages via npm.

iii) Common dependencies include 'react-router-dom', bootstrap', etc.

b) Build and Deployment:

i) Build the project using Vite's optimized build tools.

ii) Configure scripts for building and running the development server.

5.2.5 Libraries Used

5.2.5.1 Frontend Libraries

- React: “react”, “react-dom”.
- React Router: ”react-router-dom”.
- Bootstrap: “bootstrap”.
- SCSS Preprocessors: “node-sass”, “sass”.
- Vite: Vite was used as the building tool for the project.

5.2.5.1 Backend Libraries

- Express: “express”.
- Database Libraries: “mongoose” for MongoDB.
- Authentication: “jsonwebtoken”, “bcryptjs”.
- Environment Management: “dotenv”.

Chapter 6

Software Test Document

6.1 Test Items

DevHive, Version 1.0

6.2 Features To Be Tested

What follows is a list of features that will have to be determined and focused on during the testing phase:

1. User Sign-up
2. User Sign in
3. Post Job
4. Save Job
5. Chat System
6. Logout

6.3 Features Not To Be Tested

What follows is a list of features that will not be addressed during the testing phase, instead, it will be indirect as a result of other testing efforts:

1. Security
2. Update Profile
3. Location

6.4 Testing Approaches

6.4.1 Testing Levels

Unit Testing is done by the Development Manager and Test Manager, and proof of the unit testing (Sample output) is to be provided by the test manager. Acceptance Testing will be carried out by the supervisor because, after the testing stage, the supervisor must make sure that they satisfy the requirements of the platform.

6.4.2 Meetings

The test manager, project manager, and development manager shall meet once every 3 days to check the testing phase and see if there are any challenges to resolve them as soon as possible in case any critical issue arises during the testing phase. The situation will be handled through an emergency meeting.

6.5 Items Pass/Fail Criteria

The test process will start right after completing the prototype and getting the initial approval from the supervisor if no critical severity defects exist, test cases are prepared and reviewed, and usability test scripts are recorded. The test process will be complete once the run rate achieves 100%, and the pass rate reaches 80% after that we will need to satisfy most of the following requirements once all test cases are executed, Critical, high, and medium defects are closed. At most 15 low defects are still open, with no delay longer than 5 days.

6.6 Suspension Criteria and Resumption Requirements

- If team members report that there are 45% of test cases failed, then testing should be suspended until all test cases are fixed.
- The user must be registered to use our services.
- If the chat system isn't working as it's supposed to.

6.7 Test Deliverables

Test deliverable documents and information that should be delivered. This will generate the following deliverables at the end of testing:

- Test Plan
- Test Cases
- Test Execution Report
- Test Log
- Test Scripts

6.8 Environmental Needs

The following list will be required to back up the testing phase and make sure it runs smoothly:

- Windows-supported laptops, since most testing tools only work on Windows OS.
- Any device that has a browser to open the website.
- Minimal internet speed to test features that require internet connectivity.

6.9 Responsibilities

The Test manager is responsible for test execution and verification. It should allow for acceptance of all unit testing plans. The test plan and documentation are specifically the work of the project manager and test manager. The whole project team will be involved in the review of system details and any other changes suggested by the users of the website as it will enhance defect discovery which is part of the testing and development phases.

Table 6.1: Test Responsibilities

	Test Manager	Development Manager	Project Manager	Supervisor
Test Specification Document	**X**		**X**	
Test Cases	**X**			
Test Execution Report	**X**			
Test Procedures and Rules	**X**	**X**	**X**	
Test Scripts	**X**			
Acceptance Test Documentation & Execution	**X**		**X**	**X**
Test Execution		**X**		

6.10 Testing Methods

A) Static Testing:

High-Level Specification:

In this phase, we carefully go through the Requirements Specification Document to check for any discrepancies between DevHive's requirements and the actual development work done. It ensures that the requirements specification follows the standards and high-quality guidelines. Smoke testing is also conducted to ensure that the main functionalities are working as expected. This testing phase should yield a stable build that is ready for moving on to the next stage.

Low-Level Specification Testing:

The test manager and project manager ensure in this phase that all the requirements are testable. A requirement that cannot be verified cannot be tested well. We also ensure all requirements are clear, complete, consistent, traceable, and free of implementation details. This ensures good quality of the requirements document, hence everything is understandable and ready for the development team.

B) Dynamic Testing:

1. Test to Pass:

In this test stage, test cases are executed to validate that the system or application under test performs as expected. This stage is meant to verify whether the code has passed a set of test cases and is working fine.

2. Equivalent Partitioning:

In this phase, we will test various features of the DevHive platform using equivalent partitioning. We will test individual features of the testing application to check whether the variation at the input end is correctly handled. For example, the following functionality shall be tested in the DevHive application:

Sign Up:

This will be tested by inputting the functionality with username, email, and password, to check whether sign-up works smoothly and securely with different kinds of input.

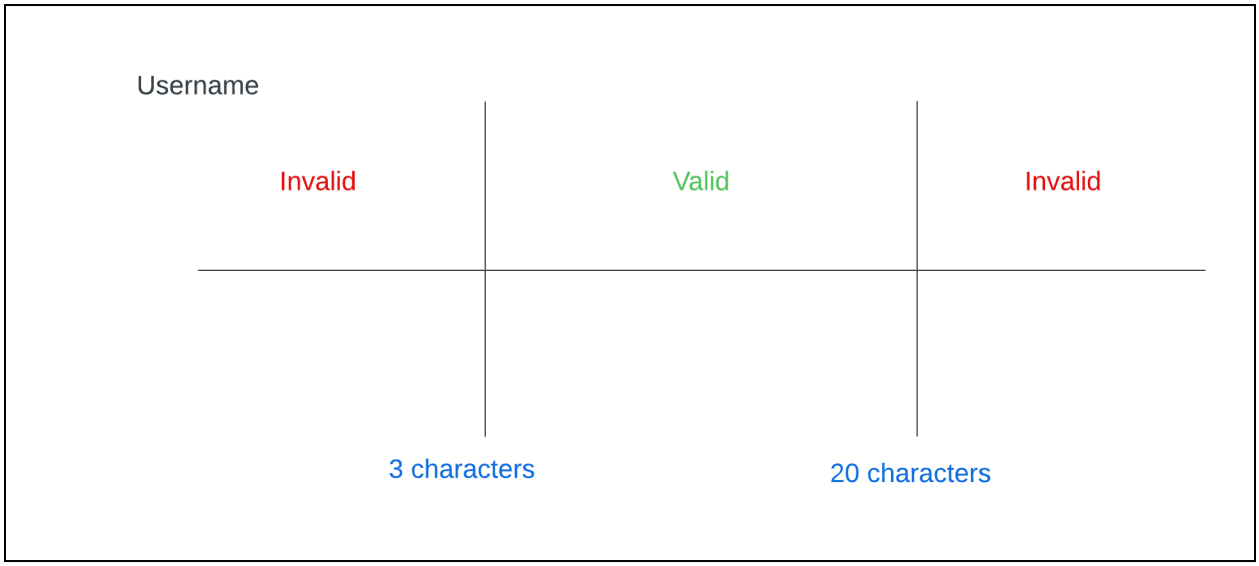


Figure 6.1: Username Equivalence Partition

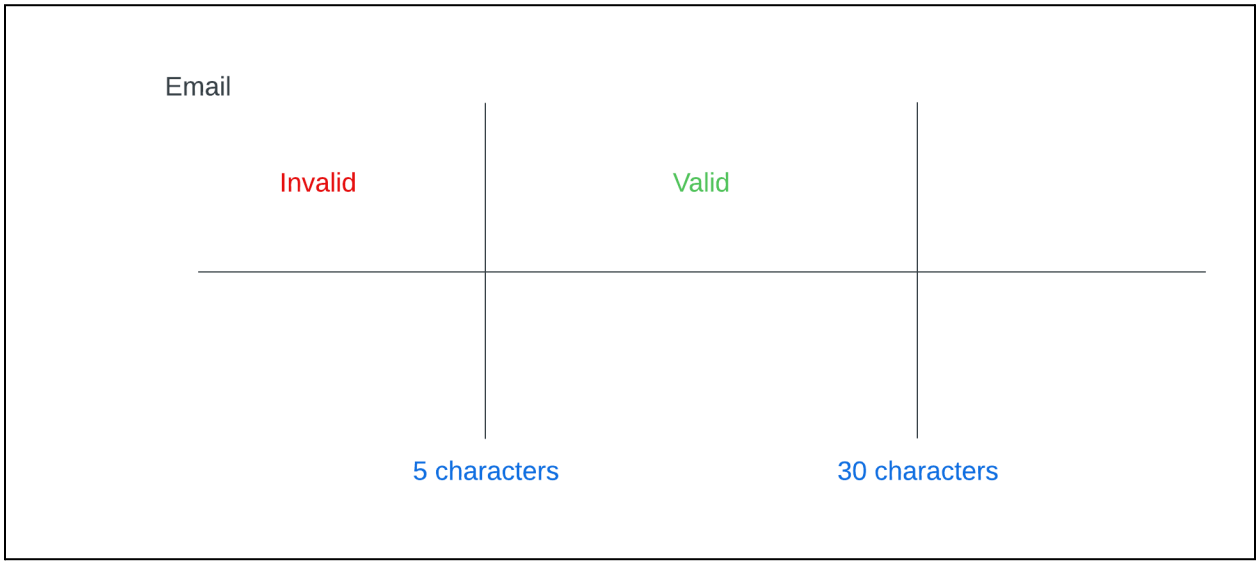


Figure 6.2: Email Equivalence Partition

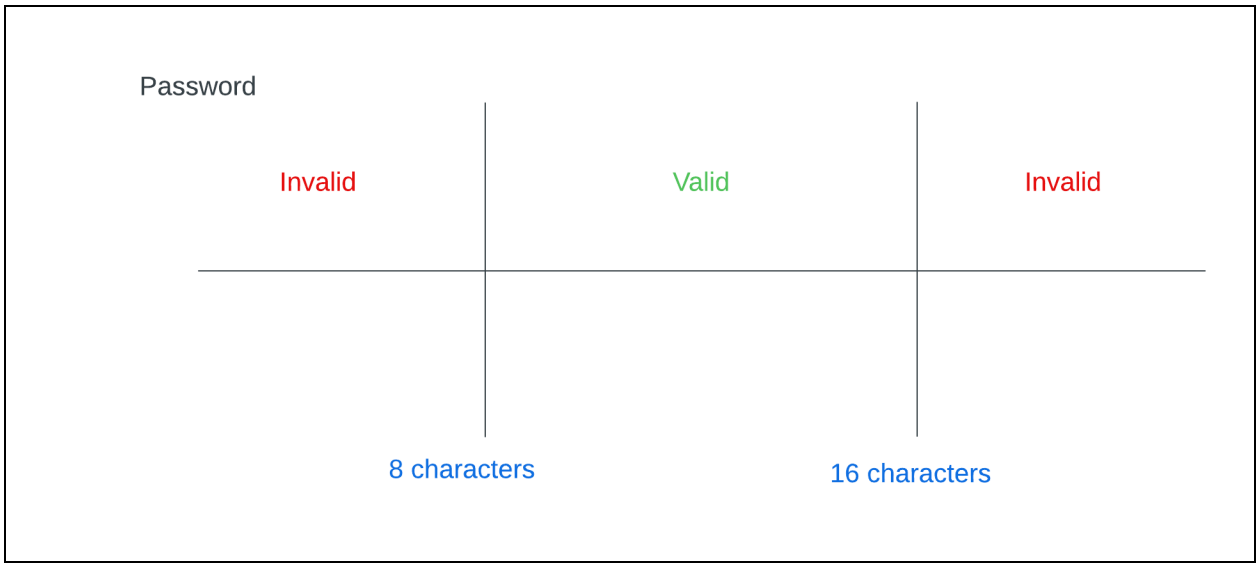


Figure 6.3: Password Equivalence Partition

3. Data Testing:

We will follow the three testing methods: boundary testing, null testing, and bad data testing. The methods will be applied to the following DevHive platform functionalities:

Sign-up: We will test this functionality for the following inputs: name, email, password, confirm password, and phone number. The following data testing methods will be used:

Testing at the edges for acceptable input ranges will be performed. For example, we would test the minimum and maximum character limits of both the name and password fields to check that edge cases are handled correctly by the system.

Null Testing: This refers to the testing of the system in the case of empty or absent inputs. We'll see how the functionality behaves when one or more fields are empty.

Bad Data Testing: This is where invalid or malformed data will be inserted to test its handling within the system. It could be in the form of an email formatted incorrectly, special characters in the name field, or the password and confirm password fields do not match. The objective of this test is to ensure that the system gracefully handles and rejects invalid data.

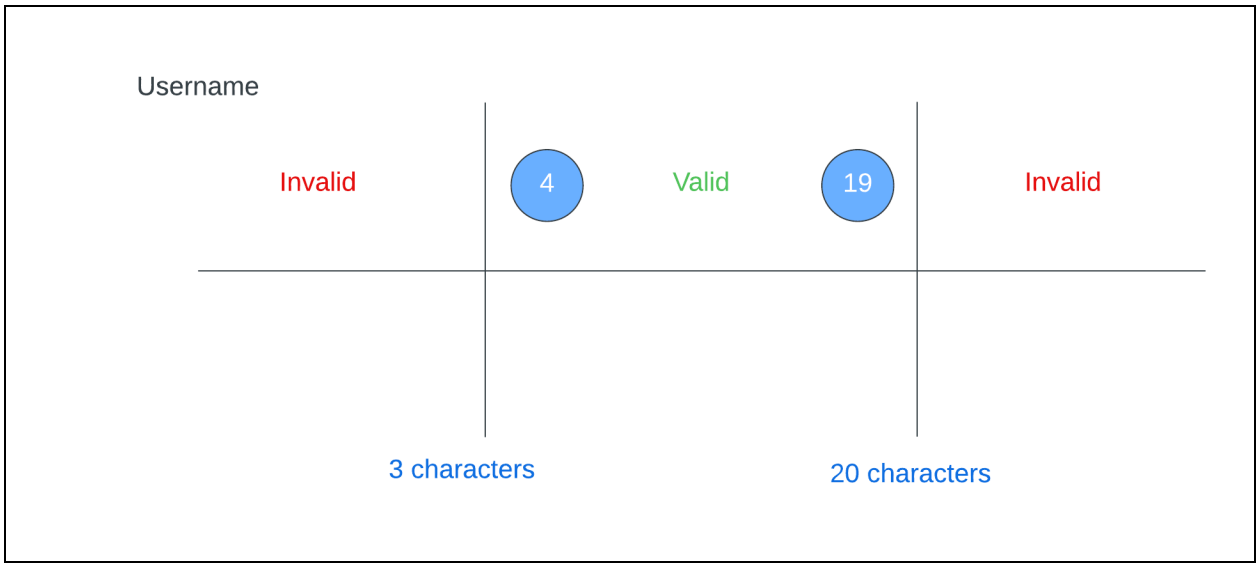


Figure 6.4: Username Equivalence Partition (Data Testing)

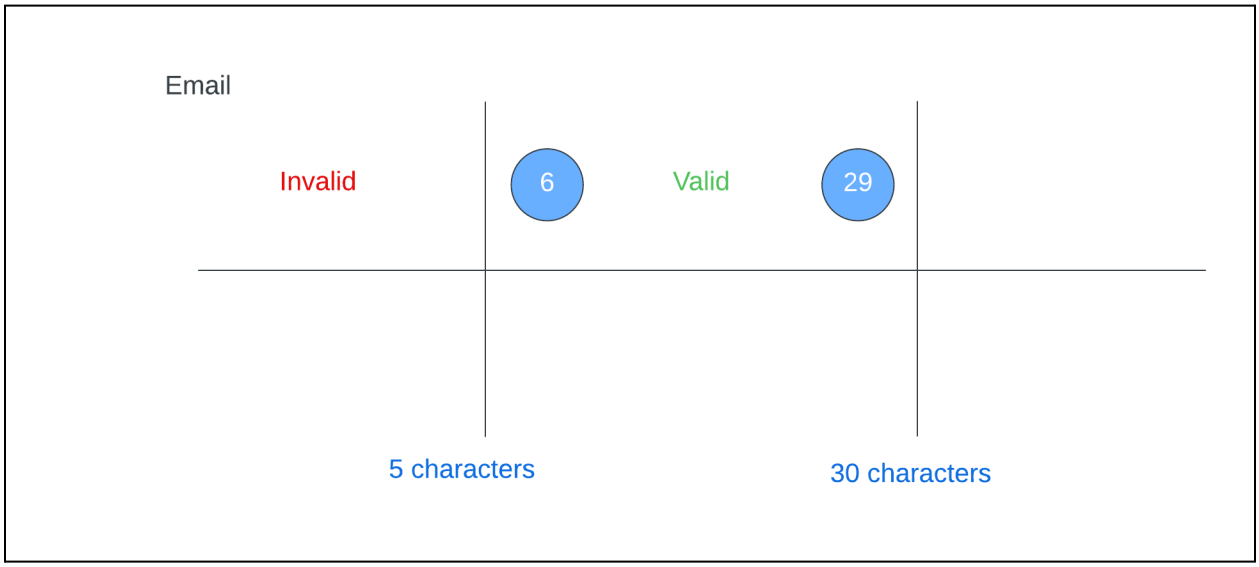


Figure 6.5: Email Equivalence Partition (Data Testing)

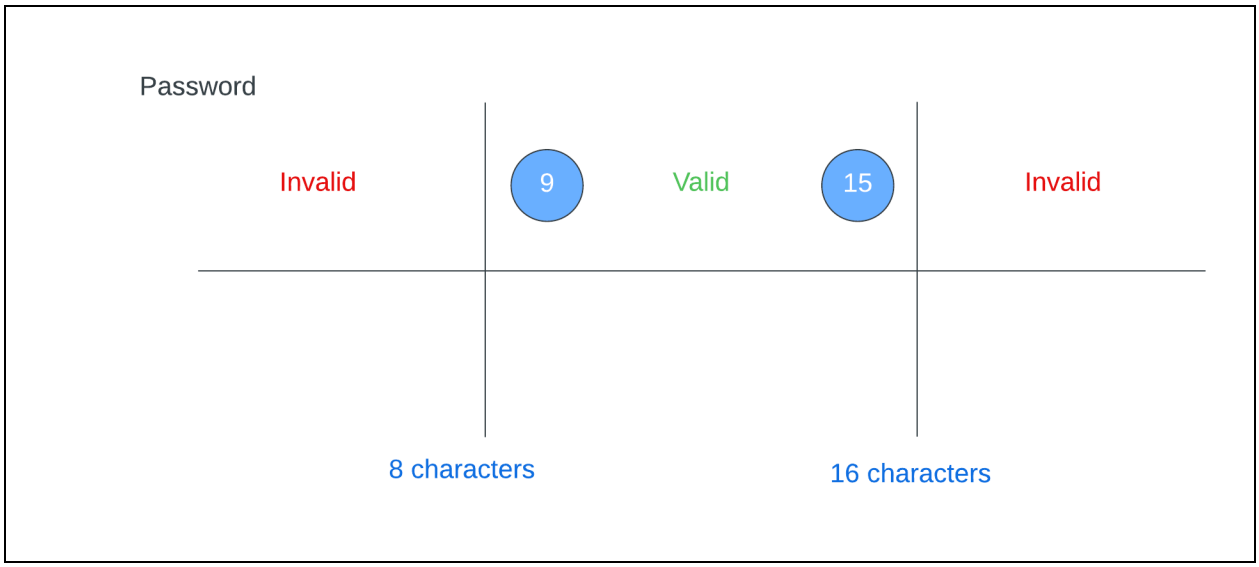


Figure 6.6: Password Equivalence Partition (Data Testing)

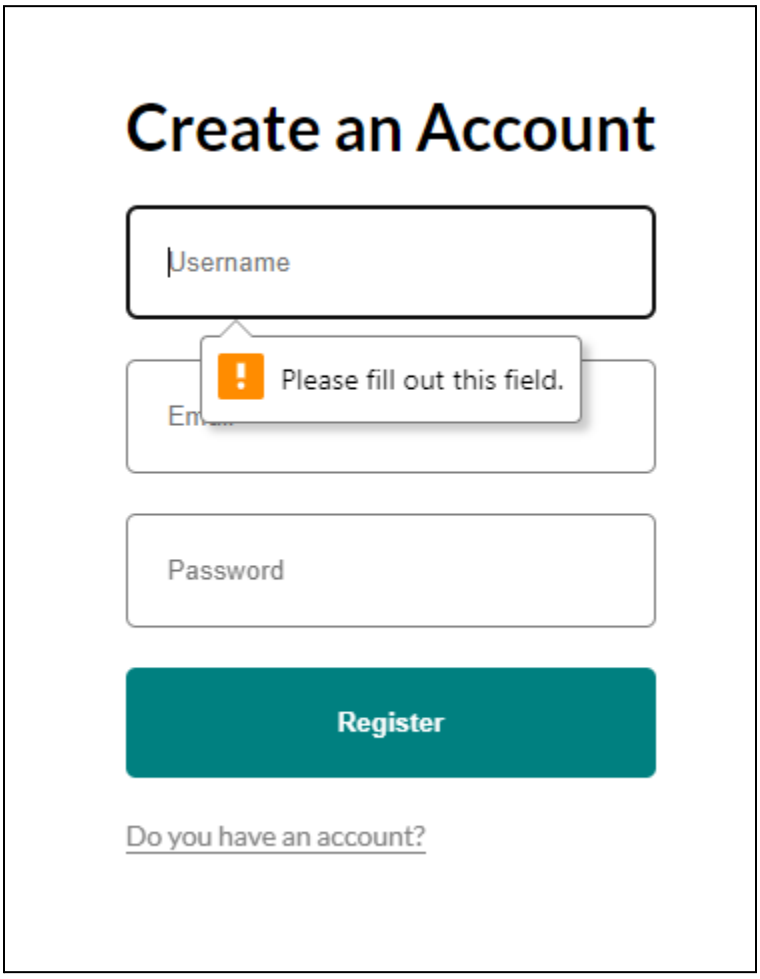


Figure 6.7: Register Null Data Testing

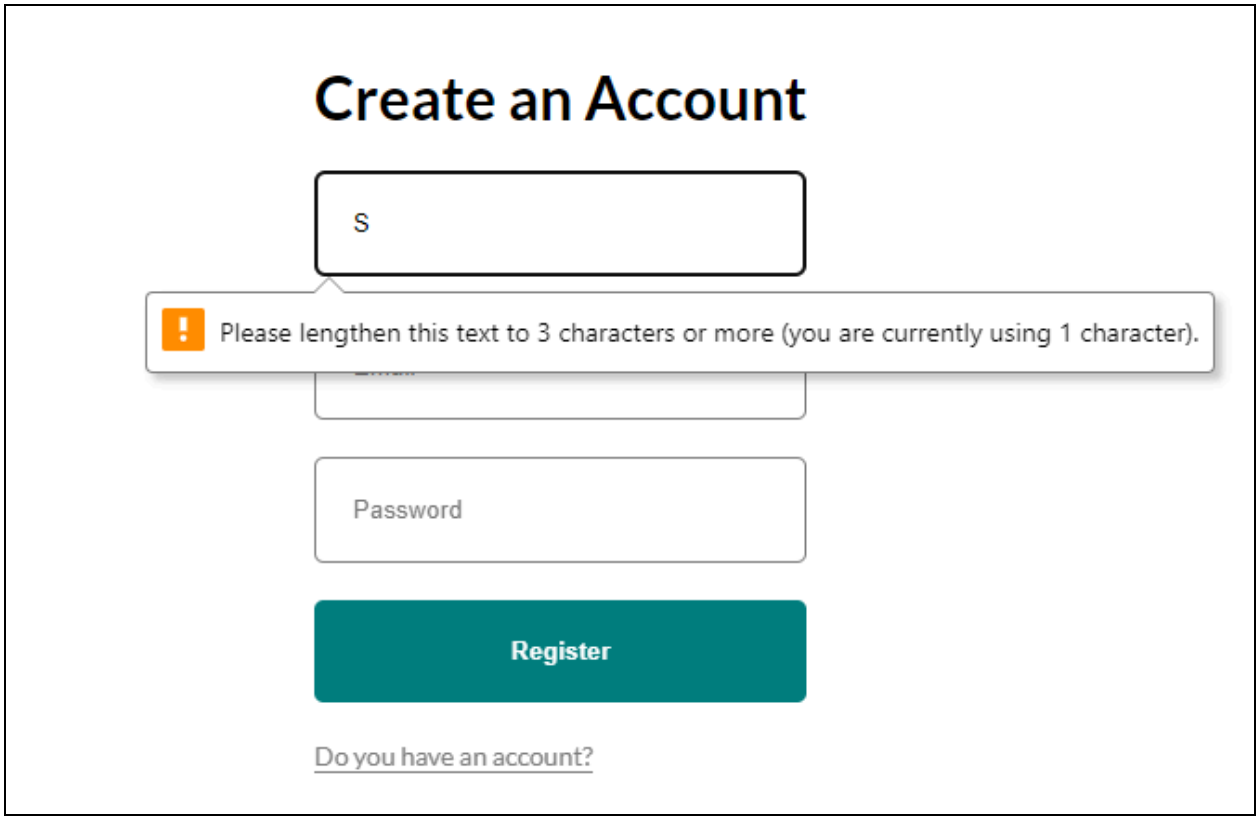


Figure 6.8: Register Bad Data Testing

Table 6.2: Test Case Sample Tables

Case Sample #1

Test Case: User Registration	Date Written: 11/8/2024	Date executed: 13/8/2024
Description	The user signs up with a username, email, and password	
Expected Result	The user signs up and creates an account	
Actual Result	As expected	
Verdict	PASS	

Case Sample #2

Test Case: User Sign in	Date Written:11/8/2024	Date executed:13/8/2024
Description	User signs into their account using email and password	
Expected Result	The user is signed in after providing an email and password	
Actual Result	As expected	
Verdict	PASS	

Case Sample #3

Test Case: Post Job	Date Written: 11/8/2024	Date executed: 13/8/2024
Description	The user inserts details and posts a job offer	
Expected Result	User posts job after providing information	
Actual Result	As expected	
Verdict	PASS	

Case Sample #4

Test Case: Save Job	Date Written: 11/8/2024	Date executed: 13/8/2024
Description	Users can save a job offer	
Expected Result	The user clicks the save icon and the job is saved on the profile	
Actual Result	As expected	
Verdict	PASS	

Case Sample #5

Test Case: Chat System	Date Written: 12/8/2024	Date executed: 13/8/2024
Description	Users can use the chat box to negotiate and provide information	
Expected Result	Messages are sent back and forth in real-time	
Actual Result	The message did not arrive to the recipient at all	
Verdict	FAIL	

Case Sample #6

Test Case: Logout	Date Written: 12/8/2024	Date executed: 13/8/2024
Description	Users can click on the logout button to exit their account	
Expected Result	The user clicks on the logout button in the profile to sign out	
Actual Result	Nothing happens when clicking on the button	
Verdict	FAIL	

Decision Table Testing

In this technique, we will be creating tables that have conditions and actions.
We will be testing the following functionalities:

Sign-up: This functionality has the following conditions:

- Username should be at least 3 characters
- Email should be more than 5 characters
- Password must be 8 characters to 16 characters in length.
-

Table 6.3: Decision Table

Condition	1	2	3	4	5	6	7	8
Username should be at least 3 characters	T	T	T	T	F	F	F	F
Email should be more than 5 characters	T	T	F	F	T	T	F	F
Password should be between 8 and 16 characters	T	F	T	F	T	F	T	F
Action								
Sign up successful	X							
Show error message for Username					X	X	X	X
Show error message for Email			X	X			X	X
Show error for Password		X		X		X		X

State Transition Diagram

The diagram below shows how the main functionalities change states:

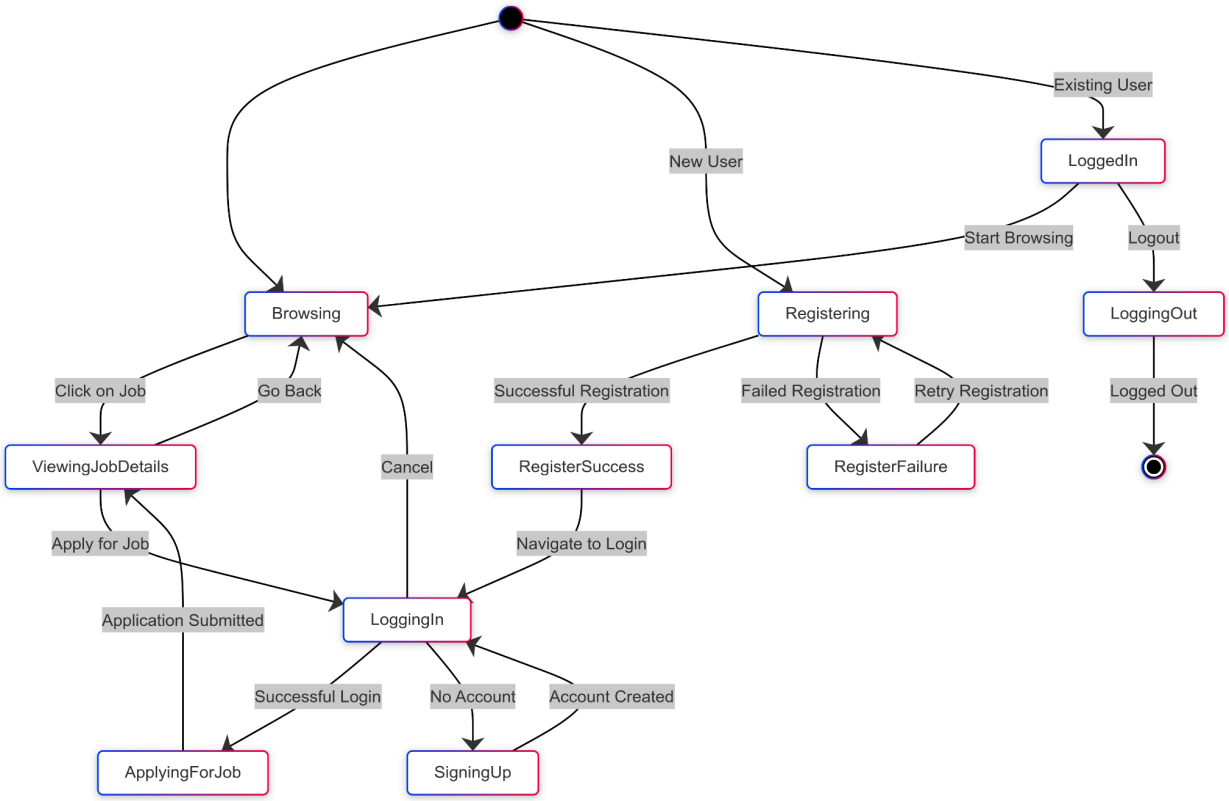


Figure 6.9: State Transition Diagram

Table 6.4: State Transition Table

Current State	Event	Next State
Initial State	New User	Registering
Initial State	Existing User	LoggedIn
Registering	Successful Registration	RegisterSuccess
Registering	Failed Registration	RegisterFailure
RegisterFailure	Retry Registration	Registering
RegisterSuccess	Navigate to Login	LoggingIn
LoggedIn	Start Browsing	Browsing

LoggedIn	Logout	LoggingOut
Browsing	Click on Job	ViewingJobDetails
Browsing	Go Back	Browsing
ViewingJobDetails	Apply for Job	LoggingIn
ViewingJobDetails	Cancel	Browsing
LoggingIn	Successful Login	ApplyingForJob
LoggingIn	No Account	SigningUp
LoggingIn	Account Created	LoggingIn
LoggingOut	Logged Out	Final State

These state transitions illustrate how various actions on the platform drive changes in the user's state within the system.

6.11 Testing Types

6.11.1 Usability Testing

As for usability, it was found that using the heuristic approach would work through specifying a set of rules related to usability and then rating each action/screen on how they match the specified rules, that being said here are the rules that were selected:

1. Strive for consistency. (by Shneiderman): which simply means that repeating actions should be done in possible ways using the same steps, and repeating commands, prompts, menus, and error messages should use the same terminology.

2. Offer informative feedback. (by Shneiderman): which means that for each action there should be some form of feedback, and the intensity of the feedback should match the importance and recurrence.

3. Error prevention. (by Nielsen): which means that the system should prevent the user from making errors.

4. Simple error handling. According to Shneiderman, it means that when the user makes an error, the system should provide the user with an easy way to correct it.

5. Visibility of system status. According to Nielsen, it means that the user has to be informed all the time of what is going on. For example, when the system is loading something, there should be some kind of indicator on the screen.

6. Design for closure. According to Shneiderman, meaning that every set of activities should be designed to produce a definite closure, giving the operator relief upon ending that n-tuple of activities and letting operators prepare for the next n-tuple of activities.

7. Allow easy reversal of actions. By Shneiderman, permitting the user to experiment Familiar options in the system should be made available so users know they can reverse their actions should the result not satisfy them.

8. Support internal locus of control (by Shneiderman), where the user should always feel like they're in control and that the system responds to their actions, meaning that the user should be the initiator of the action.

9. Make things visible. (by Norman): that is to say that things should be visible on the execution side, (i.e. the user should know that an action exists and how it is done) and on the evaluation side (i.e. the user should tell the effects of his actions).

Chapter 7

Future Work

7.1 What's Next?

Future work on DevHive may include several significant improvements aimed at the domains of user experience, efficiency in the platform, and market reach. First, sophisticated matching algorithms applying machine learning to line up appropriate projects with appropriate developers could make the whole process of hiring much easier. On the other hand, internationalization by multilingual support and localizing currency options must be implemented to attract more users to the system. More than this, building a robust feedback system in which both clients and developers can review each other in detail may establish a more transparent environment that builds trust. The next versions of DevHive may include blockchain technology to protect transactions and intellectual property, thus ensuring the safety of all parties involved. Finally, concerning the interface and user experience, constant improvements updated based on user feedback will be vital in staying competitive and relevant to users.

Resources

- Determinants of client satisfaction in web development projects from freelance marketplaces | Emerald Insight.
- The impact of Recent Layoffs on Freelancing - Small Business Labs (smallbizlabs.com).
- Freelance Forward 2023 | Upwork
- React Component Design Patterns - Part 1 - DEV Community
- https://medium.com/@reactmasters.in/unit-test-in-react-js-6e8f96186ff3
