City, University of London

MSc in Software Engineering with Cloud Computing

Project Report

2024

Technology-based Startup Team Organization Platform

Ali Momenzadeh Kholenjani

Supervised by: Dr. Christos Kloukinas

9 December 2024

Declaration

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: Ali Momenzadeh Khalenjani

Abstract

In the fast-paced realm of technology-based startups, effective team organization is pivotal for success. This project develops a tailored online platform specifically designed to address the unique challenges of startup teams, facilitating streamlined team formation and management. The solution targets existing gaps in current team organization tools by offering a specialized approach to enhance talent acquisition, team management, and team coordination within technology-based startups. Key features of the platform include job advertisement posting, portfolio management, a sophisticated matching algorithm, and goal and milestone definitions. The matching algorithm enhances the accuracy of candidate-job pairing, while the user-focused design optimizes the overall user experience.

The platform leverages web technologies and a modular monolith architecture with cloud integration to balance scalability and user-friendliness. Evaluation through automated testing, usability assessments, and performance analysis will validate its potential to improve team dynamics and operational efficiency in startup environments.

Keywords: Technology-based Startup, Tech Startup Team, Startup Team Management, Startup Team Organization, Startup Recruitment Solutions

Acknowledgments

Sincere gratitude is extended to Dr. Christos Kloukinas for his unwavering support and invaluable insights throughout this project. Appreciation is also conveyed to family members for their continuous encouragement and understanding during this endeavour.

Contents

1 Introduction and Objectives

This chapter provides an overview of the project's background and motivation, outlines its aims and objectives, and explains its contribution to academic knowledge. Additionally, it highlights the benefits the project offers to various stakeholders.

1.1 Background

Entrepreneurship through startup initiatives has grown in popularity over the recent years. Startups are recognized as short-lived and innovative enterprises that experience rapid expansion and predominantly operate within technology-driven markets (Skala, 2019). Skala highlights that startup founders may disengage from growth due to psychological factors, such as a preference for safety, lack of vision, self-doubt, insufficient resources, or complacency, which can affect their venture's evolution. According to the CB Insights (2021) report, effectively distributing responsibilities within a motivated startup team is crucial for sustaining enthusiasm and minimizing burnout. The report also identifies team disharmony and a lack of diverse skills, coupled with the difficulty startups face in recruiting essential talent, especially skilled professionals, as significant factors contributing to startup failures. Noui and Dehane (2023) further underscore that poor team formation remains a critical organizational failure factor across various stages of startup development.

Patzelt et al. (2021) reveal that team formation is crucial in the initial phase of a startup, as founders need to assemble their team to launch the venture. However, their study highlights ongoing uncertainty regarding team formation approaches, suggesting that founders often rely on personal networks or external support, while facing ongoing confusion about the process and its impacts. The significance of assembling teams with complementary skills to drive innovation in software startups is highlighted by Seppänen et al. (2017), stating that team members are typically hired based on strategic assessments of existing strengths and skills gaps, initially the founder. Seppänen (2020) further underscores the necessity of additional human resources to complement the founder's skills, even if the founder is technically proficient. This study explores diverse hiring practices in technology startups, noting that startups often hire a mix of less experienced and more seasoned individuals, either directly or through external contracts, mainly due to resource constraints that limit team size.

The paper by Diakanastasi et al. (2018) on startup team dynamics identifies several factors affecting teamwork performance, including a shared vision between members and founder, a clear set of expectations, and a relevant and complementary set of skills, the latter of which aligns with Seppänen et al. (2017) observations. Additionally, Diakanastasi et al. emphasize the significance of transparent communication, a set of well-defined roles, and the designation of tasks. Considering the dynamic

nature of startups, the team composition is liable to change over time. As Patzelt et al. (2021) note, team members might discontinue their cooperation with the entrepreneurial venture, or it might become apparent that more competencies and resources are required to address particular challenges. These changes can necessitate hiring new members during the venture development, which involves the redistribution of tasks and roles within the startup team. Insufficient strategies for task and role assignments, coupled with inadequate transparency in teamwork, can ultimately lead to startup disorganization (Cantamessa et al., 2018).

Existing tools for team organization and recruitment often provide a generic set of functionalities for task management, professional networking, or job posting. These solutions, however, fall short of addressing the specialized needs of technology-driven startup teams. They lack the advanced capabilities required for precise skill matching, dynamic role assignments, task management, and effective hiring of experts into startup teams. Moreover, they adapt inadequately to the rapid changes and unique challenges startup teams face. This creates a noticeable gap in the market for a software solution specifically designed to address these issues. Informed by this gap, the project seeks to provide a more effective and adaptive solution for startup team organizations to enhance team performance and operational efficiency in fast-evolving tech environments.

1.2 Aims and Objectives

The platform aims to serve as a central hub for the tech startup community, enabling startup founders to assemble and manage well-organized teams effectively. Additionally, skilled individuals can showcase their skills and portfolio to demonstrate their capabilities and join startup teams. This addresses the practical challenge of designing and implementing a unified platform that organizes tech startup teams by combining recruitment and team management features, facilitating expert matching, and creating a talent pool of skilled professionals.

The project objectives are as follows:

Objective 1: To enable startup founders to organize startup teams by hiring skilled individuals.

Test: To evaluate the usability and effectiveness of the job posting and team formation features through user feedback.

Objective 2: To enable skilled individuals to showcase their portfolios and join startup teams.

Test: To assess the ease of portfolio sharing and the impact of the hiring process through user feedback.

Objective 3: To match suitable skilled individuals to startup job postings and assist founders in finding the right talent.

Test: To measure the accuracy of the matching algorithm along with user feedback.

Objective 4: To facilitate startup team management through goal setting, milestone tracking, role definitions, and task management.

Test: To evaluate the clarity and usability of the goal setting, milestone tracking, role definitions, and task management features through user feedback.

These objectives have been refined from the initial proposed ones to focus on core aspects essential for the timely delivery of the project within the given timeframe and successful completion.

1.3 Project Rationale

Having spent several years immersed in the startup environment and recognizing the challenges inherent in managing startups, the researcher has gained a strong interest in startup initiatives. The decision to create this platform stems from a clear gap in existing team organizational tools, which often fail to meet the specific requirements of technology-based startups.

1.4 Beneficiaries

The startup community, especially startup founders and skilled individuals, are the primary beneficiaries of the platform. Startup founders can use the platform to post job advertisements, find skilled individuals based on specific needs, create startup teams, and manage team roles, goals, and milestones. Skilled individuals can create and share their portfolios, apply for job opportunities, and discover startup job postings through the platform's matching algorithm. Moreover, the researcher will advance their knowledge in frontend development, cloud services, and system architecture.

1.5 Scope

The project encompasses the design, implementation, testing, and deployment of an online platform aimed at organizing technology-based startup teams. It follows a phased approach, beginning with preparation through literature review and requirements analysis, followed by design, development, testing, and deployment to a cloud environment. This scope has been refined to prioritize core functionalities essential for team formation and management, including user registration, job advertisement, portfolio management, and a matching algorithm. Additionally, essential team management features are considered, subject to project priorities. Although the platform will address a range of features in the design phase, the implementation will prioritize the most critical functionalities first to ensure timely delivery. Aspects such as the hiring process, team communication tools provided by social media platforms, and other factors beyond team organization are outside the project's scope.

1.6 Report Structure

The next chapter provides the context, summarizing literature findings on team organization, dynamics, hiring practices, current software solutions, and the justification for a new solution. Chapter 3 details the research strategy, development approach, and tools used. The results from the software development, including any changes or new findings, are presented in Chapter 4. Chapter 5 offers a detailed comparison of the project outcomes with the initial objectives. Finally, Chapter 6 evaluates the project's success, key decisions, and lessons learned.

2 Context

The context chapter is foundational to understanding the intricate challenges and dynamics involved in technology-based startup team organization. This chapter will explore the critical elements influential to team formation, recruitment, and management within the high-stakes technology startup environment. The discussion integrates theoretical frameworks and current tools to identify gaps and establish the necessity for a specialized solution.

2.1 Overview of Technology-Based Startups

Startups represent a distinguished sector in the global economy, characterized by their potential for rapid innovation, market disruption, and a strong reliance on technology—hence, they are often described as 'technology-based startups' (Skala, 2019). These emerging businesses, established on innovative concepts, aim to disrupt markets with new technologies or business models while facing considerable uncertainty and requiring flexibility in resource management (Salamzadeh and Kawamorita Kesim, 2015). Despite their promise, 90% of these entrepreneurial endeavours fail (Genome, 2024). An analysis of startup post-mortem cases by Noui and Dehane (2023) identified recurring themes in failed ventures, including financial challenges, product-market misalignment, regulatory hurdles, and issues related to team competency. Moreover, critical factors in startup failures include the founder's lack of passion and direction, as well as shortcomings in human resource competency and coordination, as reported by CB Insights (2021) and highlighted by Skala (2019). These observed challenges underscore the importance of strategic foresight, strong leadership, and a well-organized team.

Furthermore, challenges in tech startups often stem from factors such as IT expenses, team skill gaps, and limited resources, which are critical in the early stages (Santisteban et al., 2023). A systematic literature review by Santisteban et al. (2023) identified significant failure factors, particularly highlighting how a lack of either technological or entrepreneurial skills in the founding team can impede a startup's growth. Cantamessa et al. (2018) further emphasize that even when technical expertise is present, deficiencies in business acumen can still prevent the startup from thriving. Thus, assembling a team with a balanced skill set, including technical and business expertise, is paramount. Founders should focus on building a talent pool with complementary technical and business skills, which is crucial for addressing the diverse challenges of startups. This strategic approach, as highlighted by Melegati and Kon (2020), is essential for navigating the complexities and risks that often impede success, particularly in the early stages.

2.2 Team Formation and Dynamics in Startups

In the early stage of a startup, founders need to assemble a team to launch the venture, a process that begins with an understanding of the founder's knowledge gaps, as analysed by Patzelt et al. (2021) and discussed by Klada (2018). This is similar to the findings of Seppänen et al. (2017), who also emphasize the importance of recognizing these gaps when forming the initial team. Although Patzelt et al. (2021) study suggests that founders might rely on personal and professional networks or external support, Thirasak (2019) observed that founders often used their personal networks to have brainstorming sessions and recruit their initial team members. Another study by Seppänen et al. (2017) found that technology-based startups can enhance their capabilities by utilizing experts, new employees, or subcontractors—a decision influenced by the founder's knowledge, experience, and financial constraints. Their research showed that while internal hires included experienced and less experienced individuals, subcontracting was reserved exclusively for seasoned professionals. These hires are strategically selected to maintain the equilibrium between business needs and financial resources, aiming to complement the current skills within the team, or initially the founder (Seppänen, 2020).

From the founder's perspective, hiring new members is constrained by the legitimacy and recognition of the new organization, as well as by financial limitations and a limited pool of talent, often drawn from acquaintances in the early stages (Klada, 2018; Thirasak, 2019; Kemell et al., 2020). While some teams are built through startup networks or friends, there are instances where founders struggle to find suitable candidates who meet their specific needs (Klada, 2018; CB Insights, 2021; Kemell et al., 2020). Founders also initially face knowledge inadequacies, which complicate the addition of new members to the team (Klada, 2018). In these situations, founders often rely on referrals and implement trial periods for new members (Klada, 2018). The findings of Klada (2018), much like those of Seppänen (2020), emphasize balancing team members' skills with financial compensation, initially prioritizing resilience, soft skills, and part-time roles, while offering competitive salaries. However, unlike Seppänen's focus on immediate hiring through subcontracting or external resources to manage small teams, Klada's findings emphasize an evolving strategy (Seppänen, 2020; Klada, 2018). As the startup grows, it shifts from prioritizing soft skills and part-time roles to hard skills and full-time hires, reflecting a more dynamic adjustment of hiring priorities over time, rather than a static, resource-driven approach (Klada, 2018).

2.3 Team Organization in Technology Startups

Given the uncertainties and dynamic nature of startups, combined with unique organizational challenges resulting from their nascent stage, limited resources, and need for agile and adaptative team structures, team organization emerges as a critical concern (Blank and Dorf, 2012; Ries, 2011). The Resource-

Based View (RBV) emphasizes that human resources are often the most strategic asset for startups, and managing these resources effectively is essential for maintaining a competitive advantage (Barney, 1991). Moreover, resource constraints namely financial and human resources, in particular, can necessitate strategic allocation of tasks and responsibilities to maximize efficiency. This requires effectively leveraging the strengths of team members, and striking a balance between soft skills and technical expertise while fostering team cohesion, communication, and shared leadership, as emphasized by the Theory of Team Work (Katzenbach and Smith, 2015). As startups evolve, the team composition may need to change, with new roles emerging to meet the increasing demands of the business. This fluid nature of startup teams necessitates continuous adjustments in roles, tasks, and even team structure, aligning with the principles of Agile Methodology, which highlights iterative development and flexibility (Beck et al., 2001).

Building on the importance of strategic resource allocation and team structure, effective team organization is essential for optimizing performance and maintaining cohesion within technology startups. Diakanastasi et al. (2018) stress the importance of a shared vision and clear expectations among team members, noting that aligning team members with the startup's goals is crucial for maintaining integrity and focus. They further emphasize that acquiring a relevant and complementary set of skills is vital, with clear communication and well-defined roles and responsibilities necessary for smooth operations and effective collaboration (Diakanastasi et al., 2018). Furthermore, the dynamic nature of startups can necessitate changes in team composition over time. As Patzelt et al. (2021) discuss, team members might discontinue their cooperation with the entrepreneurial venture, or the founder may realize that more competencies and resources are required to address particular challenges. This scenario can result in the need to hire new members during the venture's development, leading to the reallocation of tasks and roles within the startup team. Onboarding new members presents challenges, as highlighted by Melegati and Kon (2020), particularly when there is inadequate transparency and documentation of team activities (Cantamessa et al., 2018).

Theories of team dynamics provide valuable insights into how startups can manage these challenges. For example, Belbin's Team Roles Theory emphasizes the importance of balancing different roles, such as "Coordinator," "Plant," and "Implementer" within a team to maximize overall effectiveness by matching team members' skills and strengths with project or task needs (Belbin, 2011). Similarly, Tuckman's Stages of Group Development highlights the different phases that teams go through, from formation to performance, and the need for management practices that support each stage to improve team cohesion and performance (Tuckman, 1965). Hackman's Model of Team Effectiveness further underscores the importance of clear goal-setting, well-defined roles, efficient processes, and task allocation, which are critical in the fast-paced startup environment where the ability to rapidly iterate

and refine approaches can be a significant competitive advantage (Hackman, 2002). Hackman's Model also emphasizes the importance of task characteristics, team composition, and processes for effective team functioning (Hackman, 2002). Furthermore, recent research highlights the growing importance of managing remote teams and the impact of digital collaboration tools on team dynamics (Kirkman et al., 2002). As remote and hybrid work environments become more prevalent, startups ought to incorporate features that support these modes of operation, ensuring that team organization remains effective across different work settings.

2.4 Current Software Solutions for Team Organization

Technology-based startups encounter significant challenges, including resource constraints, skill gaps, and high failure rates, often due to ineffective team formation and management. As startups evolve, team formation transitions from informal networks to more structured approaches, yet founders face constraints like limited legitimacy and a narrow talent pool. Effective team organization is crucial, requiring clear goals and milestones, well-defined roles, and efficient task management. These factors highlight the necessity for specialized tools that facilitate job postings, provide access to a broader talent pool, support portfolio sharing, and effectively match talent with startup needs.

In response to these challenges, various software solutions exist to facilitate team organization, offering a range of functionalities. However, these solutions often fail to comprehensively address the unique challenges of technology-based startups, which demand more dynamic, flexible, and specialized tools. The following sections explore the strengths and limitations of these existing solutions, particularly in the context of technology startups.

2.4.1 Recruitment and Networking Platforms

Platforms like LinkedIn and Indeed offer generic solutions for job posting and candidate searches, making them essential tools for talent acquisition (LinkedIn, 2024; Indeed, 2024). LinkedIn, in particular, excels in professional networking, enabling hiring managers to connect with potential candidates and industry experts (LinkedIn, 2024). Its recommendation algorithms assist in identifying potential hires based on network connections, endorsements, and profile similarities (LinkedIn, 2024). Additionally, gig-based platforms, including Upwork (Upwork, 2024) and Freelancer (Freelancer, 2024), cater to the needs of startups looking for temporary or project-based hires by providing access to a global pool of freelancers who can be onboarded quickly to meet specific project demands. However, while these platforms are effective in connecting hiring managers with talent, they are not specifically designed for managing startup job postings or profiles of individuals interested in the startup environment. Moreover, they lack integrated team management functionalities, which are necessary for sustaining team cohesion in a startup environment. The absence of features like role assignment, goal

setting, milestone definition, and ongoing hiring within these platforms limits their effectiveness in the fast-paced, evolving nature of startup teams. These limitations may stem from the platforms' focus on broad market appeal and the complexities of integrating recruitment with team management. Developing such features requires a specialized design to handle the rapid changes characteristic of startups, which may not align with the platforms' primary objectives.

2.4.2 Collaboration Platforms

Microsoft Teams and Slack are widely used collaboration platforms that offer real-time communication and file-sharing capabilities (Microsoft, 2024; Slack, 2024). These platforms are essential for maintaining continuous communication within startup teams, particularly in remote or hybrid work settings. Microsoft Teams also integrates seamlessly with other Microsoft 365 tools, providing a cohesive experience for document management and collaborative work (Microsoft, 2024). Slack, on the other hand, excels in facilitating communication through its channels, direct messaging, and notification system, keeping team members informed and aligned (Slack, 2024). Despite their effectiveness in communication, these platforms do not adequately address the critical aspects of team organization within startups. They lack tools for job posting and sophisticated algorithms for matching individuals with startup jobs. Additionally, they offer limited functionality in terms of role assignment and goal management within startup settings, as well as the dynamic ability to restructure the team by hiring new members as needed. This is likely because their primary focus is facilitating communication rather than managing team composition or recruitment processes. Incorporating such features would require a significant shift in their core functionalities and could introduce complexities that might detract from their user-friendly interfaces.

2.4.3 Project Management Tools

Project management tools, namely Asana, Trello, and Jira, are widely adopted for their robust capabilities in task management, team collaboration, and milestone tracking (Asana, 2024; Trello, 2024; Atlassian, 2024). These tools effectively manage workflows, task assignments, and progress monitoring. Asana, for instance, is renowned for its user-friendly interface and flexibility in handling various project types, making it a reasonable choice for startup project management (Asana, 2024). Trello offers a visual approach to task management through its card-based system, allowing teams to track tasks and projects with ease (Trello, 2024). Lastly, Jira is primarily used in software development environments, providing advanced functionalities for bug tracking and agile project management (Atlassian, 2024). However, despite their strengths, these tools exhibit significant limitations regarding team organization within technology startups. Notably, they lack integrated features for talent acquisition and dynamic role assignment, which are crucial in a startup environment where team

composition and roles can change rapidly. Additionally, these platforms do not offer sophisticated matching algorithms that could assist startups in finding the right talent more efficiently. These limitations arise because project management tools focus primarily on task execution and workflow management, not team formation or recruitment. Adding features like talent acquisition and dynamic role assignment would require significant changes to their core design, potentially complicating their usability.

2.4.4 Integrated Solutions

Some platforms like Monday.com and Basecamp attempt to offer a more integrated approach to team organization by combining project management, collaboration, and recruitment features (Monday.com, 2024; Basecamp, 2024). Monday.com (Monday.com, 2024) offers customizable workflows and integrates well with various third-party tools, while Basecamp (Basecamp, 2024) focuses on simplicity and ease of use to provide a central hub for project management, team communication, and file sharing. Nonetheless, even these integrated solutions do not fully meet the specialized needs of technology startups. They often lack advanced job posting features tailored to startups, as well as sophisticated matching capabilities to find the most suitable candidates. Additionally, these platforms inadequately address the dynamic nature of startup teams, where roles and team structure frequently evolve in response to project demands and market conditions. These features may be absent because implementing advanced matching algorithms and dynamic team management functionalities is challenging. Such specialized capabilities may not align with the broader focus and business models of these platforms, which are designed to serve a wide audience rather than the specific needs of startups.

2.4.5 Summary of Existing Solutions

The comparative analysis of existing solutions reveals considerable shortcomings in meeting the specific requirements of technology-based startups. While each platform provides valuable features, such as job posting, goal setting, or task management, they typically function in isolation, missing the integrated capabilities essential for dynamic team management, effective talent acquisition, and the continual adjustments of team roles and structure that startups frequently require. Most platforms specialize in a particular area, but they do not offer the comprehensive functionality needed to address the multifaceted organizational challenges of startups. The following table outlines the strengths and limitations of these platforms, emphasizing the need for a tailored solution that can address the unique organizational demands of startup teams.

Feature	Asana	Trello	Jira	LinkedIn	Indeed	Upwork	Freelancer	Microsoft Teams	Slack	Monday.com	Basecamp	Specialized Solution
Job Posting				✓	✓	✓	✓					✓
Portfolio Showcase				✓		✓	✓					✓
Matching Algorithm				Limited		Limited	Limited					✓
Integrated Team and Talent Management										Limited	Limited	✓
Dynamic Role Assignment			✓									✓
Goal Setting	✓	✓	✓					✓	✓	✓	✓	✓
Milestone Tracking	✓	✓	✓					Limited	Limited	✓	✓	✓
Task Management	✓	✓	✓					✓	✓	✓	✓	✓

Table 2.1: Comparison of Features Across Team Organization Solutions (Asana, 2024; Trello, 2024; Atlassian, 2024; LinkedIn, 2024; Indeed, 2024; Upwork, 2024; Freelancer, 2024; Microsoft, 2024; Slack, 2024; Monday.com, 2024; Basecamp, 2024).

2.5 Necessity for a Specialized Startup Team Organization Solution

The literature review reveals noticeable shortcomings and challenges as outlined below:

- **Inadequate Integration of Recruitment and Team Management:** Traditional recruitment platforms are disconnected from the tools needed for ongoing team management, making it difficult to maintain a cohesive and high-performing team. This separation forces startups to manage multiple platforms, leading to inefficiencies and potential misalignments between recruitment and team operations.
- **Lack of Dynamic Role Assignment:** Existing tools generally do not support the fluid team structures and frequent role changes common in technology-based startups. Most cannot assign multiple roles to a single user, which is crucial in startups where team members often handle multiple responsibilities simultaneously.
- **Insufficient Talent Matching Algorithms:** Current platforms fail to offer sophisticated matching algorithms that align the specific skills and experiences presented in candidates' portfolios and CVs with the evolving needs of startup teams. While some platforms provide basic keyword or job title matching, they often lack the depth needed to accurately pair

candidates with the multifaceted roles in startups. This shortfall hinders startups from identifying candidates whose detailed skills and experiences match their specific requirements.

- **Limited Support for Startup-Specific Challenges:** Existing solutions are not tailored to address the unique demands of startup environments, such as specialized job postings, ongoing team reformation, and goal/milestone assignments. Startups often require tools that can evolve with them, supporting rapid growth, changes in direction, and the need to quickly onboard new team members.

Despite the availability of various tools, startups often face challenges in integrating talent acquisition, portfolio management, dynamic role assignment, and team management into a unified, cohesive system. These tools typically operate in isolation, leading to a fragmented approach that complicates workflow and hinders the comprehensive management of team dynamics. This disjointed approach reveals a significant gap in the market for a solution capable of effectively unifying these functions. This project addresses these gaps by developing a specialized platform that integrates critical features such as job advertisements, portfolio management, a sophisticated matching algorithm, team and task management, and dynamic role assignment into a unified system. Central to the platform's functionality is the matching algorithm, which aligns individuals' skills and experiences with the specific needs of startup teams, as detailed in the next chapter (see Section 3.7).

3 Methods

This chapter outlines the methods employed in executing the project. It details the systematic approach to preparation, design, development, deployment, and evaluation, ensuring academic rigor and practical relevance. The chosen methods were geared toward fulfilling the project's objectives and ensuring the platform's usability, functionality, and scalability.

3.1 Research Approach

The project adopted a Design and Creation research approach to develop a practical solution addressing gaps in technology-based startup team organization. This involved systematically designing, developing, refining, and evaluating a software system. Key tasks included requirements preparation, architectural framework design, and iterative platform development. Usability testing was incorporated to ensure the platform effectively met user needs and addressed the identified technological gaps (Oates, 2006, Chapter 8).

3.2 Preparation Phase

The preparatory phase was initiated with a literature review to identify gaps in existing industry software solutions. This review was critical in guiding the project's direction, ensuring the platform addresses current limitations and unmet needs in team organization for technology-based startups. Basing requirements on literature review rather than direct stakeholder input is a common and practical approach when stakeholder engagement is constrained (Sommerville, 2016, Chapter 4). Insights from the literature review were used to compile and categorize the project's functional and non-functional requirements. These requirements were written in natural language (see Appendix B) to guide the design and development phases, ensuring critical features for a startup team organization platform were addressed.

3.3 Design Phase

3.3.1 Use Case Definitions

At the start of the design phase, use case definitions were developed based on the requirements, detailing interactions between user roles (skilled individual and startup founder) and the platform (see Appendix C). A simplified Cockburn Use Case Template (Cockburn, 2001) was used to emphasize essential elements like actor interactions and event flows, consolidating actions such as creating, viewing, updating, and deleting into single use cases to maintain clarity and simplicity.

3.3.2 Requirements Traceability Matrix (RTM)

A Requirements Traceability Matrix (RTM) was created to link each functional requirement to its corresponding use cases, ensuring alignment and verification throughout design and development (see Appendix D) (Sire, 2024). This approach prioritized critical components, supported project management, mitigated risks, and guided testing activities. Non-functional requirements were excluded from the RTM.

3.3.3 System Architecture

Careful planning of the system architecture was essential for scalability, performance, and maintainability. According to Richards (2022, Chapters 2 and 4), smaller client-server systems gain advantages from a monolithic architecture because of its straightforwardness. Despite certain scalability limitations, monolithic architectures are favoured over microservices, which add complexity to transaction management and data consistency. Given the project's limited scope and objectives, a monolithic architecture was chosen. To enhance scalability and maintainability, a modular monolith approach was adopted, combining the development speed of monoliths with the extensibility and maintainability of microservices (GeeksforGeeks, 2024; Thoughtworks, 2024).

3.3.4 System Design and Visual Modelling

Unified Modelling Language (UML) was employed to design and visualize the system architecture and interactions (Booch et al., 2005). UML's standardized notation facilitated clear communication of complex system behaviours through various diagrams (Booch et al., 2005). A component diagram (see Appendix H) adhering to the modular monolith architecture established a structured framework (Sommerville, 2016, Chapter 6). Additionally, a class diagram (see Appendix F), multiple sequence diagrams (see Appendix I), and a deployment diagram (see Appendix J) were developed to detail the system's internal structure and dynamic interactions, supporting deployment and integration (Sommerville, 2016, Chapter 5; IBM Rational Software Architect, 2023). Visual Paradigm (2024) was chosen to create these diagrams due to the researcher's familiarity.

3.3.5 User Interface (UI) and User Experience (UX) Design

User experience was prioritized to ensure platform usability. Wireframes were created using Canva (2024) as blueprints for frontend development to meet usability requirements and provide a foundation for the user interface and overall user experience. Due to resource constraints, human participants were not involved at this stage. Instead, the design was reviewed by the researcher to ensure it aligned with established web development best practices and usability principles.

3.4 Development Phase

3.4.1 Software Development Life Cycle Style

The development phase utilized a hybrid software development life cycle model, integrating elements of Waterfall and iterative/incremental approaches to balance upfront planning with flexibility for refinements (Sommerville, 2016, Chapter 2). This model facilitated the timely delivery of a functional Minimum Valuable Product (MVP) (Ries, 2011) by focusing on high-priority features such as job advertisement, portfolio management, and team management. Initially, the project followed a structured Waterfall-like process, including a literature review, requirement capture, use case definition, system architecture design, and wireframing (Sommerville, 2016, Chapter 2). This established a foundation to ensure that core requirements were well understood and the system architecture was scalable.

Subsequently, development proceeded in distinct phases, each targeting specific features. The iterative/incremental elements allowed for gradual development and refinement, maintaining flexibility in scheduling and accommodating ongoing adjustments (Sommerville, 2016, Chapters 2 and 9). The hybrid approach was selected after evaluating various SDLC models. While the Waterfall model was too rigid for the project's evolving needs, and models like Rational Unified Process (RUP) and Spiral were too resource-intensive, the iterative/incremental model offered the necessary flexibility and manageability (Sommerville, 2016). This enabled continuous development, early delivery of functional components, and prioritization of critical features. The project adopted a structured testing strategy that included incremental automated testing throughout the development process and final usability testing upon MVP completion (Sommerville, 2016, Chapter 8; Ries, 2011). This strategy ensured that software functionality was continuously validated against requirements, while usability testing confirmed that the platform met user needs and expectations (detailed in Section 3.6).

3.4.2 Tools, Technologies, and Development Environment

The development leveraged a range of tools and technologies. Backend development was implemented utilizing ASP.NET Core, leveraging the researcher’s expertise in C# and .NET (Microsoft, 2024). Swagger was employed for automatic API documentation and interactive endpoint testing (Swagger, 2024). Entity Framework Core (EF Core) managed database interactions by accessing and manipulating data (Microsoft, 2024). For backend testing, the xUnit library combined with Moq facilitated isolated component testing (xUnit, 2024; Moq, 2024). Frontend development utilized React.js for its moderate learning curve and strong community support (React, 2024; Rathinam, 2023). The Fetch API handled HTTP requests, while Node Package Manager (npm) managed dependencies and scripts (MDN Web Docs, 2024; npm, 2024). React Hook Form streamlined form creation and validation, and Jest ensured the functionality and reliability of React components (React Hook Form, 2024; Jest, 2024). Visual

Studio was used for .NET backend development, and Visual Studio Code (VSCode) provided a lightweight environment for frontend tasks (Microsoft, 2024). GitHub managed version control and code reviews, utilizing Git to track changes (GitHub, 2024).

3.4.3 Implementation Architecture and Patterns

The backend was developed using a modular monolith architecture to support independent module development and integration (Ghannoum, 2024; Murugan, 2021). Each module, representing distinct functionalities, was built as an independent ASP.NET Core project and integrated into a central ASP.NET Core API (Ghannoum, 2024). The API acted as a gateway (façade), managing requests and responses across modules and facilitating seamless communication through extension methods for service registration and route configuration (Murugan, 2021). Each module adhered to a layered pattern, including controllers, models, Data Transfer Objects (DTOs), services, and a data access layer, ensuring logical separation and maintainability (Ghannoum, 2024). This architecture design allows for future scalability, enabling modules to function independently while interacting cohesively within the system (Murugan, 2021). The React frontend rendered all user interface views and communicated with the backend via a RESTful API. React managed client-side routing and state, while the backend served as the data provider. Each module used separate schemas within the same database, ensuring logical separation and enabling a modular structure for potential transition to a microservices architecture (Jovanović, 2023). Referential integrity was enforced using foreign keys, enabled by default in Microsoft SQL Server during development and Azure SQL Database during deployment (Microsoft, 2024).

3.4.4 Implementation of Features

After analysing the software system's magnitude and complexity, the functional and non-functional requirements (see Appendix B) were prioritized based on their impact, implementation complexity, and interdependencies. Core functionalities, such as user authentication and job posting, received high priority due to their essential roles in platform operations. For example, account creation (User Management 1.1, see Appendix B.1) and job advertisement management (Job Management 2.1, see Appendix B.1) were foundational, supporting critical components like portfolio and team management. User authentication enabled registration and login, necessary for managing portfolios and applying for jobs, while job posting allowed founders to post positions and individuals to apply, facilitating team formation. Conversely, less essential features, such as task management (Task Management 6.1, see Appendix B.1), were assigned lower priority and scheduled for later development increments.

3.5 Deployment Phase

The deployment phase focused on ensuring that the platform was accessible, reliable, and scalable by utilizing the Microsoft Azure cloud platform (Microsoft, 2024). Azure was chosen for its robust integration with the .NET ecosystem, excellent support for Microsoft SQL Server via Azure SQL, and its versatile deployment architectures (Microsoft, 2024; Kingsley, 2024). Microsoft SQL Server was selected as the relational database to ensure ACID properties and seamless integration with the .NET technology stack (Microsoft, 2024; Sahatqija et al., 2018; Plesk, 2024). The deployment strategy employed Docker (2024) for containerization and Kubernetes (2024) for orchestration, managing the frontend and backend as separate components within the same Kubernetes cluster. This setup allowed the frontend and backend to run in independent pods, ensuring modularity, scalability, and ease of maintenance. Azure Kubernetes Service (AKS) orchestrated the deployment of containers, while Azure Container Registry (ACR) securely stored Docker images (Microsoft, 2024). Application data was handled via Azure SQL Database (Microsoft, 2024), providing a scalable and secure data storage solution with high availability and minimal downtime in case of hardware failures. Regular backups were configured to ensure data integrity and support disaster recovery. Azure Blob Storage (Microsoft, 2024) was utilized for storing files, such as user profile pictures, uploaded CVs, and other documents, ensuring performant and scalable file management with seamless backend integration. Environment variables, including Azure SQL Database connection string and Azure Blob Storage credentials, were passed into the backend through the deployment YAML file.

3.6 Testing and Evaluation

Testing and evaluation were crucial to ensure the platform met functional and non-functional requirements. A structured strategy was adopted where continuous feedback from automated tests aligned the platform with requirements, while final usability testing validated user experience and satisfaction. Focused testing at key lifecycle points validated software functionality and gathered user feedback, supporting the platform's alignment with project objectives and timeline flexibility.

3.6.1 Incremental Automated Testing

Backend unit tests were conducted using the Arrange-Act-Assert (AAA) pattern (Telerik, 2024) with Moq for mocking dependencies, ensuring each module met specifications and functioned correctly in isolation. After backend validation, frontend component testing verified individual UI elements. Automated tests were run after each development increment to validate new features and identify defects early, allowing necessary adjustments before proceeding.

3.6.2 Usability Testing

Usability testing was conducted after the final increment to assess the user experience and gather feedback on the complete MVP. Following Nielsen's heuristic approach, five participants from relevant demographics performed key tasks such as job posting, job application, portfolio creation, and team management (Nielsen, 2000). This approach ensured realistic scenario testing and actionable feedback (Barnum, 2011; Murphy, 2018). Additionally, ethical guidelines were adhered to, ensuring informed consent and participant anonymity (Yocco, 2020).

3.6.2.1 Participant Selection and Personas

Participants were selected based on their startup experience, either as founders, skilled individuals, or both, capable of performing relevant tasks. Diverse technical and leadership backgrounds ensured comprehensive feedback (Barnum, 2011; Murphy, 2018).

3.6.2.2 Informed Consent and Anonymization

Participants received Information and Informed Consent Sheets detailing the study and data usage (see Appendix M). Anonymized user accounts ensured no personally identifiable information (PII) was stored (Barnum, 2011; Yocco, 2020).

3.6.2.3 User-Centric Approach

The usability test focused on real-world use cases, with participants receiving a Usability Test Overview (see Appendix N) outlining tasks like posting jobs and managing teams. Screen interactions and voice were recorded to capture real-time actions (Barnum, 2011; Murphy, 2018; Yocco, 2020; MIT, 2024).

3.6.2.4 Survey Design

Post-test, participants completed a Qualtrics (2024) survey with rating-scale and open-ended questions on usability, feature experiences, and overall satisfaction. The survey assessed navigation ease, job posting intuitiveness, and platform response time, with open-ended questions for feedback (Murphy, 2018; Barnum, 2011). The offline survey is in Appendix O.

3.6.2.5 Data Analysis

Data from recordings and surveys were analysed to assess usability from the users' perspective. Both quantitative ratings and qualitative responses were reviewed to identify trends in satisfaction, ease of use, and usability issues (Barnum, 2011; Murphy, 2018; MIT, 2024).

3.7 Matching Algorithm Implementation

The matching algorithm is essential for recommending suitable job advertisements to skilled individuals and helping founders identify the most aligned applicants. A hybrid approach was selected after evaluating various methods, including rule-based (Keyword Matching, Boolean Search), statistical

techniques (Cosine Similarity, Latent Semantic Analysis), and advanced machine learning models (Word Embeddings, Deep Learning) (Manning et al., 2008; Turing, 2024). This approach combines Term Frequency-Inverse Document Frequency (TF-IDF) with Cosine Similarity, enhanced by N-Grams and a custom dictionary of multi-word phrases dedicated to technology-based startups (Manning et al., 2008).

3.7.1 Algorithm Design and Selection

The choice of this hybrid approach was influenced by the following considerations:

3.7.1.1 Text Similarity Accuracy

TF-IDF quantifies the importance of terms in job descriptions and CVs by considering term frequency and inverse document frequency (Manning et al., 2008, Chapter 6). This ensures that the most relevant terms are given higher weight. Cosine Similarity is then applied to the $TF - IDF$ vectors to measure the similarity between job descriptions and skilled individuals' portfolios and CVs (Manning et al., 2008, Chapter 6). This combination provides a solid method for assessing relevance between job advertisements and applicants (Manning et al., 2008, Chapter 6).

The formula for $TF - IDF$ is given as:

$$TF - IDF = TF \times IDF \tag{1}$$

where:

- TF is the term frequency.
- IDF is calculated as:

$$IDF = \log\left(\frac{N}{n+1}\right) + 1 \tag{2}$$

- N is the total number of documents.
- n is the number of documents containing the term (Manning et al., 2008, Chapter 6).

3.7.1.2 Contextual Understanding via N-Grams and Phrases

By generating N-Grams (unigrams and bigrams), the algorithm captures the contextual relationships between sequences of words, allowing it to understand nuances that single-word analysis might miss (Nithyashree, 2021). Additionally, a custom dictionary of multi-word phrases like "machine learning" and "data science" was integrated to treat these phrases as cohesive units during text analysis (Kalra et al., 2022). This prevented fragmentation and enhanced matching accuracy for complex job descriptions and portfolios.

3.7.1.3 Efficiency and Interpretability

This hybrid model is computationally efficient and scalable compared to more resource-intensive deep learning models. It allows for easier debugging and ongoing improvements while maintaining real-time performance, making it well-suited for the platform's needs (Manning et al., 2008).

3.7.2 Implementation Process

3.7.2.1 Data Preprocessing

Before applying TF-IDF, the text from job advertisements, portfolios, and CVs undergoes preprocessing. This involves cleaning the text by removing stop words, punctuation, and special characters, improving the quality of the data for analysis (Manning et al., 2008, Chapter 2). Tokenization is then applied to generate individual terms and multi-word phrases (N-Grams) from the custom dictionary (Kalra et al., 2022).

3.7.2.2 Feature Extraction

Once pre-processed, the cleaned text is transformed into numerical vectors using TF-IDF, measuring the relevance of terms within each document. This step enriches the text data representation by assigning higher weights to important terms (Manning et al., 2008, Chapter 6). N-Grams are generated to capture word sequences, thereby improving contextual understanding.

3.7.2.3 Similarity Computation (Cosine Similarity)

For job descriptions and skilled individuals' portfolio items and CVs, TF-IDF vectors are generated to represent the text data. Cosine Similarity is then applied to these vectors to compute the similarity scores between documents. Two similarity scores are calculated—one for unigrams (N = 1) and another for bigrams (N = 2)—to capture individual word-level and multi-word phrase relationships. The average of these two scores is used to ensure a balanced assessment of both types of matches, preventing the algorithm from being overly sensitive to individual terms or too reliant on specific phrases. Results are ordered by similarity score, with those below a set threshold excluded to focus on the most relevant matches (Manning et al., 2008; Karabiber, 2024; N-Gram, 2024).

The formula for Cosine Similarity (Cosine similarity, 2024) is given as:

$$\boldsymbol{Cosine\ Similarity} = \frac{\sum_{i=1}^{n} \boldsymbol{A_i B_i}}{\sqrt{\sum_{i=1}^{n} \boldsymbol{A_i^2}} \sqrt{\sum_{i=1}^{n} \boldsymbol{B_i^2}}} \quad (3)$$

Where:

- $\boldsymbol{A_i}$ and $\boldsymbol{B_i}$ are components of the TF-IDF vectors for documents A and B, respectively (Cosine similarity, 2024).

3.7.2.4 Matching Process and Ranking

The algorithm matches skilled individuals to job advertisements based on their portfolios and helps founders identify suitable applicants using their portfolios and CVs. A similarity threshold filters out low-relevance matches, and a configurable parameter limits the results to the top matches, returning only the highest-scoring matches to the user.

3.7.3 Evaluation and Testing

The matching algorithm was evaluated using both synthetic and real-world datasets. Synthetic data was generated with ChatGPT (OpenAI, 2024) to simulate realistic job advertisements and skilled individual profiles, including job titles, skills, experience, portfolios, and CVs, to test the algorithm under realistic conditions.

3.7.3.1 Accuracy Assessment

The algorithm's correctness was measured using accuracy, precision, recall, and F1-Score (Labelf.ai, 2024). A ground truth dataset was generated via ChatGPT (OpenAI, 2024) to compare provided matches with expected results. The similarity threshold was fine-tuned to reduce false positives and improve overall accuracy.

3.7.3.2 Threshold Optimization

The similarity threshold was iteratively adjusted to balance strictness and leniency, prioritizing high-relevance matches while minimizing irrelevant results (false positives) and missed opportunities (false negatives).

3.8 Security Considerations

Secure data storage and communication were ensured by utilizing Azure SQL Server with Transparent Data Encryption (TDE) and SSL/TLS for data in transit (Microsoft, 2024). The backend API employed the ASP.NET Core Identity framework for hashed passwords and role-based access control, restricting access to sensitive data (Microsoft, 2024). JSON Web Tokens (JWTs) were generated to authenticate and authorize frontend users (Auth0, 2024), while HTTPS was enabled for secure client-API communication. Additionally, Google authentication was integrated as an alternative method, leveraging Google's authentication workflow (Google, 2024).

4 Results

This chapter presents the detailed findings and outcomes obtained throughout the various stages of the project, including preparation, design, development, deployment, and testing and evaluation of the platform. Each section outlines specific results and observations at various project stages, providing a comprehensive account of all findings.

4.1 Preparation Phase Results

The requirement engineering process yielded 26 functional and 20 non-functional requirements, tailored to technology-based startups. The functional requirements are grouped into six categories: User Management, Job Management, Portfolio Management, Team Management, Matching Algorithm, and Task Management. For instance, in Job Management, startup founders can create, update, and delete job advertisements, including details like startup name, stage, industry, job title, description, required skills, and application deadlines. The non-functional requirements are categorized into Scalability, Security, etc.; for instance, the system must efficiently scale to accommodate increasing users and data. Further details are available in Appendix B.

4.2 Design Phase Results

4.2.1 Use Case Definition Results

A total of 16 use cases were identified to capture user interactions with the platform, focusing on the roles of startup founders and skilled individuals. An example is presented in Table 4.1, detailing the use case for Managing Job Advertisements. This use case outlines how startup founders can create, update, or delete job advertisements. The startup founder must be signed in to perform these actions. The flow of events includes steps for managing job advertisements, which consolidate multiple actions into a streamlined process. As a result, the job advertisement is created, updated, or deleted according to the founder's actions. The complete set of use case definitions is available in Appendix C.

Title: Managing Job Advertisements
ID: 06
Description: Allows startup founders to create, update, and delete job advertisements.
Actor: Startup Founder
Preconditions: Startup founder must be signed in.
Flow of Events: 1. Startup founder navigates to the job advertisements page. 2. Startup founder selects the create, edit, or delete option. 3. **IF** creating a new job advertisement **THEN**

3.1 Startup founder enters startup details (e.g., startup name, stage, industry) and job details (e.g., title, description, required skills, and application deadline). 3.2 System saves the new job advertisement and makes it visible to all users. 4. **IF** updating an existing job advertisement **THEN** 4.1 Startup founder selects the job advertisement to update. 4.2 Startup founder modifies the job advertisement details as needed. 4.3 System saves the updated advertisement. 5. **IF** deleting a job advertisement **THEN** 5.1 Startup founder selects the job advertisement to delete. 5.2 System prompts for confirmation. 5.3 Startup founder confirms the deletion. 5.4 System deletes the job advertisement.
Postconditions: Job advertisement is created, updated, or deleted as per the startup founder's action.

Table 4.1: Use Case Details for Managing Job Advertisements

4.2.2 System Architecture and Design Results

Adopting a modular monolith architecture resulted in the development of five distinct backend modules: Job Management, Matching Management, Portfolio Management, Team Management, and User Management. Each module addressed multiple closely related use cases, as illustrated in Figure 4.1, which shows the modular organization of the backend system. A Shared component was created to facilitate common functionalities across modules, and an API component enabled interaction between the frontend and backend.

Although modules were built independently, some dependencies emerged, necessitating inter-module communication at runtime. For example, the matching algorithm required functionalities from the Job Management and Portfolio Management modules, which are highlighted in Figure 4.1. These dependencies were managed internally without the need for external API calls. For a complete view of the overall component diagram, including frontend interactions, refer to Appendix H.

The database design, illustrated in Figure 4.2, used schema separation for each module to ensure data isolation and enforce referential integrity. For example, the Teams, Milestones, Goals, TeamRoles, and TeamMembers tables were organized within the TeamManagement schema, while the JobApplications and JobAdvertisements tables were kept in the JobManagement schema. This separation helped maintain modularity and logical organization of data structures. For a complete view of the data model, refer to Appendix G.

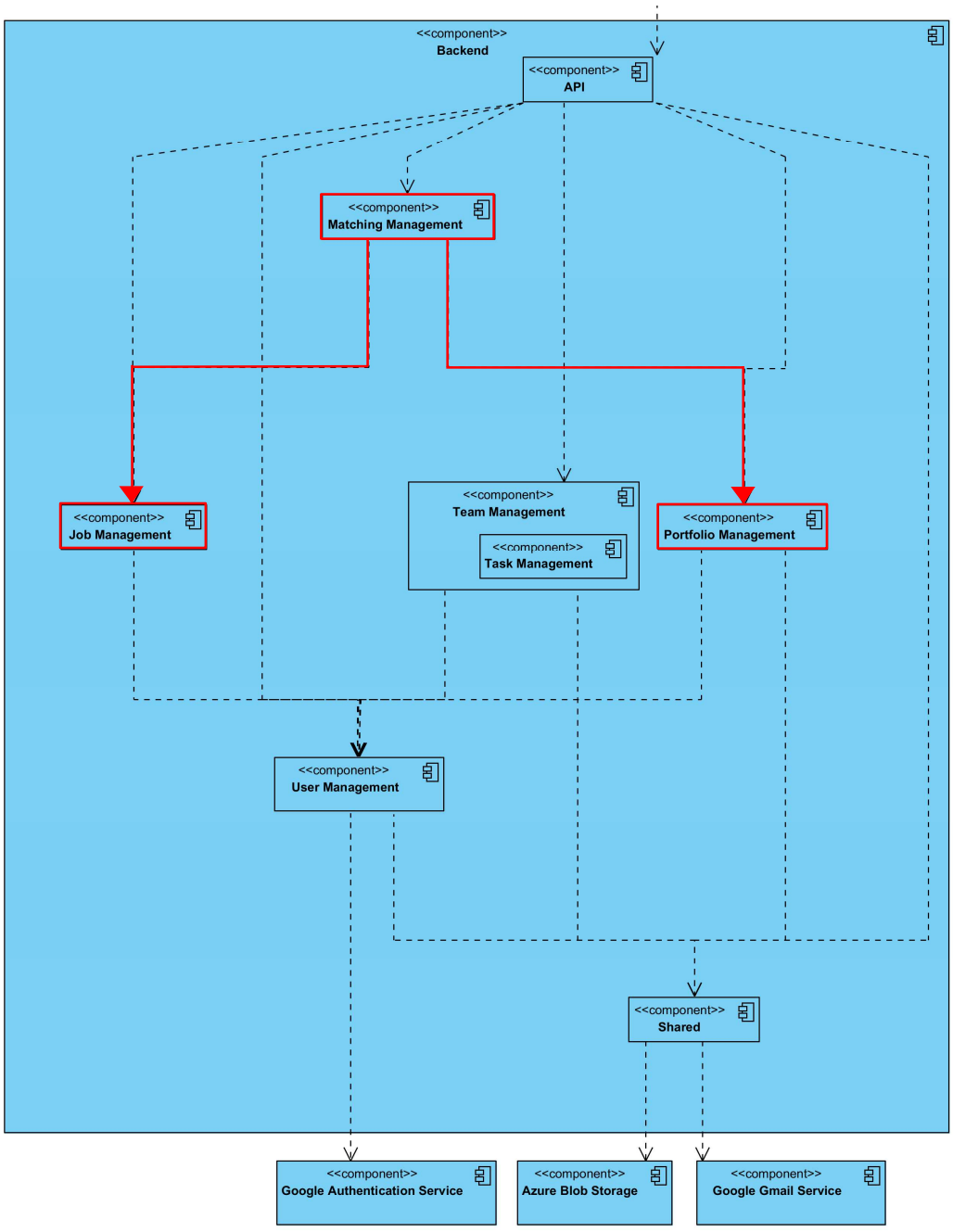


Figure 4.1: Modular Monolith Backend Component Diagram

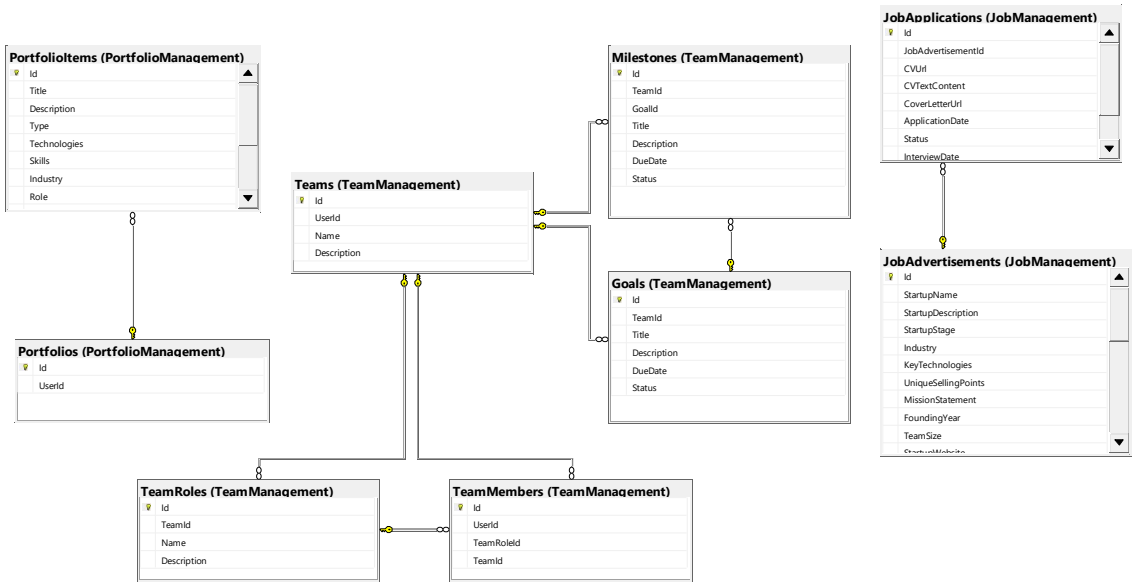


Figure 4.2: Partial Data Model of the Platform

4.2.3 User Interface and User Experience (UI/UX) Design Results

Sixteen wireframes were created to illustrate key sections of the platform, such as job posting, portfolio management, and team management. To avoid redundancy, only one representative form was depicted for similar entities (e.g., job advertisement creation and update). Figure 4.3 shows the wireframes for the job applications management page for startup founders. The complete set of wireframes is provided in Appendix K.

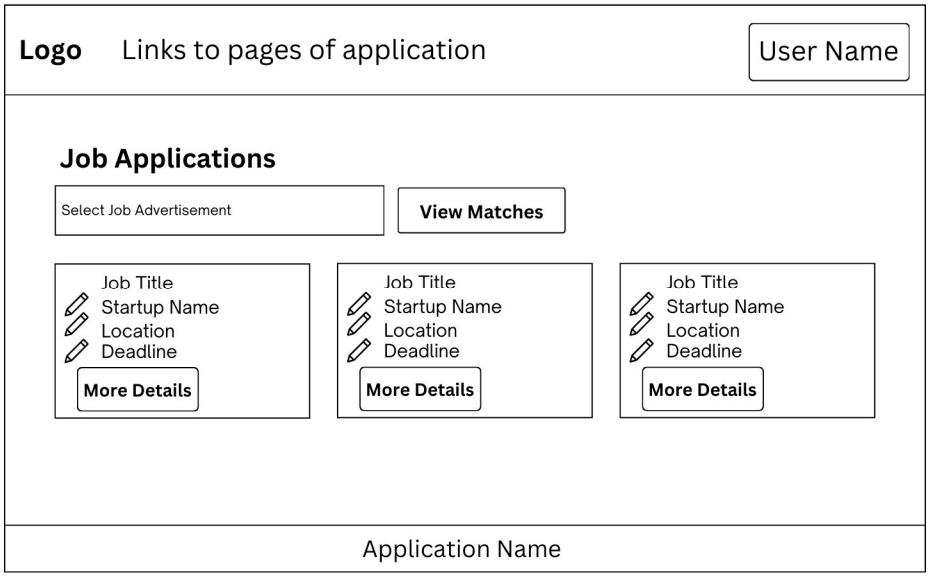


Figure 4.3: Job Applications Management Page Wireframe

4.3 Development Phase Results

4.3.1 Solution Structure and Organization

The platform's backend and frontend solutions were structured to promote modularity, scalability, and maintainability. Figure 4.4 illustrates the backend solution structure, where modules such as Job Management and Team Management are organized in a modules directory, following the naming convention StartupTeam.Module.ModuleName. Each module is divided into specific subdirectories: Controllers define API endpoints, Data contains the database context and Entity Framework migrations, DTOs manage API communication objects, and Extensions provide module registration logic invoked in the central API project (StartupTeam.Api). Models define database entity classes, Services implement business logic accessed by controllers, and Validation provides custom attributes for DTO property validation, such as due date checks. The Dockerfile is placed in the StartupTeam.Api directory to manage containerization. This structured approach ensures logical separation, reusability, and maintainability across the backend solution.

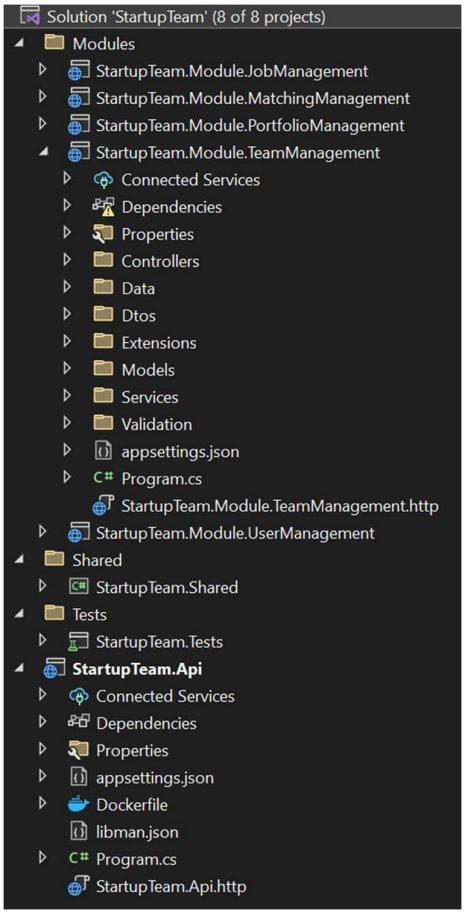


Figure 4.4: Backend Solution Structure in ASP.NET Core

Figure 4.5 shows the frontend solution structure. Reusable components like forms and cards are in the components directory, while context manages application-wide states like authentication. The pages directory organizes application pages by functionality, such as job applications or teams, with subdirectories for specific forms and management features. Tests are in the tests directory, utility functions for API requests and error handling are in utils, and styles are centralized in App.css. Routing is configured in App.js, and a Dockerfile in the root directory handles containerization for deployment.

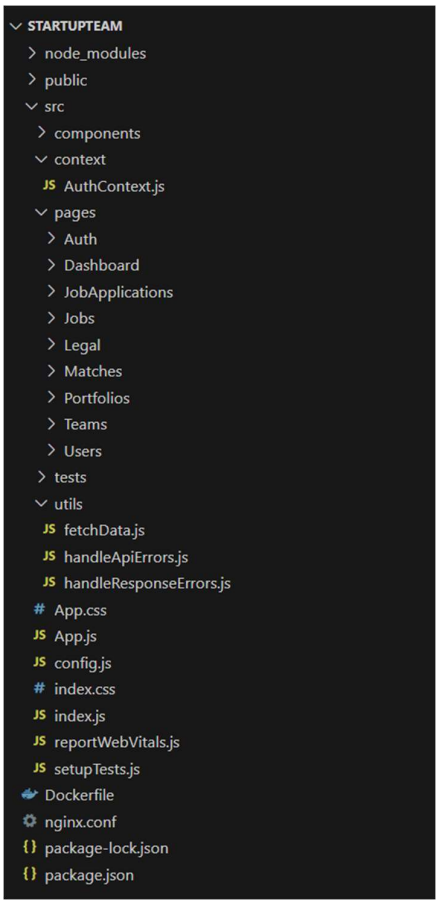


Figure 4.5: Frontend Solution Structure in React

4.3.2 Implementation of Features Results

Adhering to the hybrid development model, the platform's features were implemented over three main increments. Detailed screenshots of the implemented features are provided in Appendix L.

4.3.3 Increment 1: Foundational Features

The first increment focused on the core functionalities essential for the basic operation of the platform. The modules developed during this phase included User Management, Job Management, and Portfolio

Management. This increment established the foundational elements of the platform, ensuring that critical functionalities were operational and stable.

User Management allowed for secure user registration and login, with email verification and role-based access control. This included login via email and password, along with Google Authentication (see Appendix L.1 for SignIn and SignUp Pages). Users could update their profiles through the Profile Update Form (see Appendix L.4) and view the public profiles of others registered on the platform. Figure 4.6 illustrates the All Users Page, where users can browse public profiles, including information about both startup founders and skilled individuals (see Appendix L.6 for a Startup Founder Public Page and Appendix L.7 for a Skilled Individual Public Page).

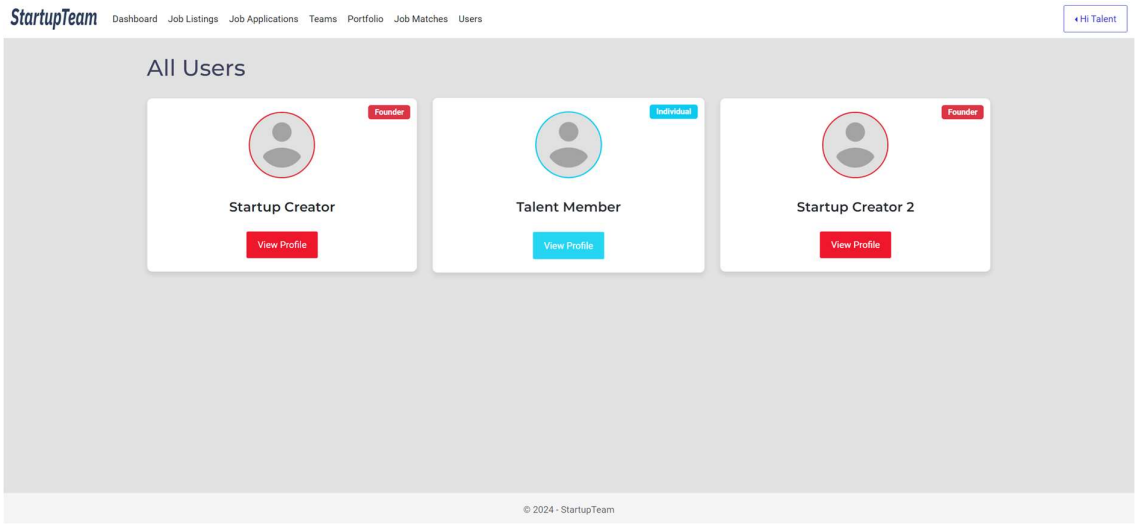


Figure 4.6: All Users Page

Job Management allowed startup founders to create, update, and delete job advertisements, supporting all necessary operations for managing job postings. Figure 4.7 illustrates the Job Advertisements Management Page, where startup founders can create, update, and view applications for their job advertisements. Job Management also facilitated the visibility of job posts to all users. Figure 4.8 shows the Job Listings Page, where skilled individuals can browse and apply for job advertisements.

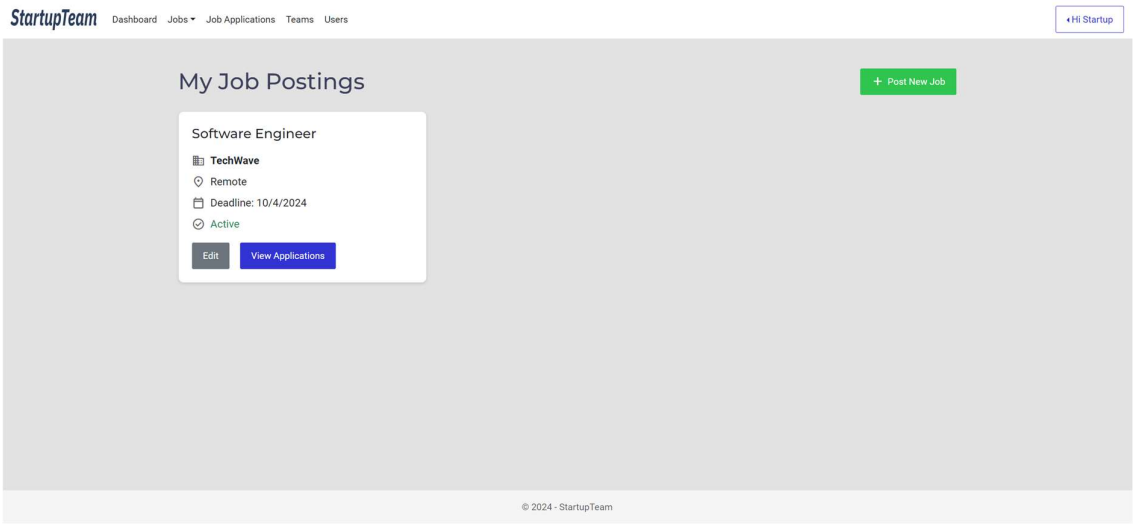


Figure 4.7: Job Advertisements Management Page

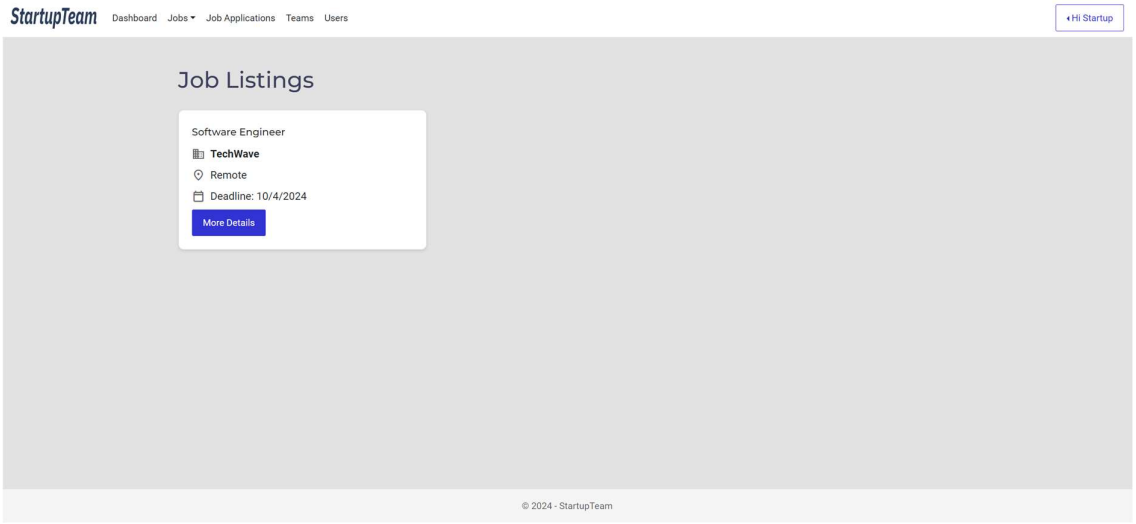


Figure 4.8: Job Listings Page

Additionally, Figure 4.9 displays the Job Applications Management Page, enabling startup founders to review applications for each job posting and apply the matching algorithm to find the most suitable candidates (implemented in Increment 2). For more details on forms and other views, see Appendix L.8 for the Job Advertisement Creation Form, Appendix L.11 for the Job Advertisement Page, and Appendix L.13 for the Startup Founder Job Application Update Page.

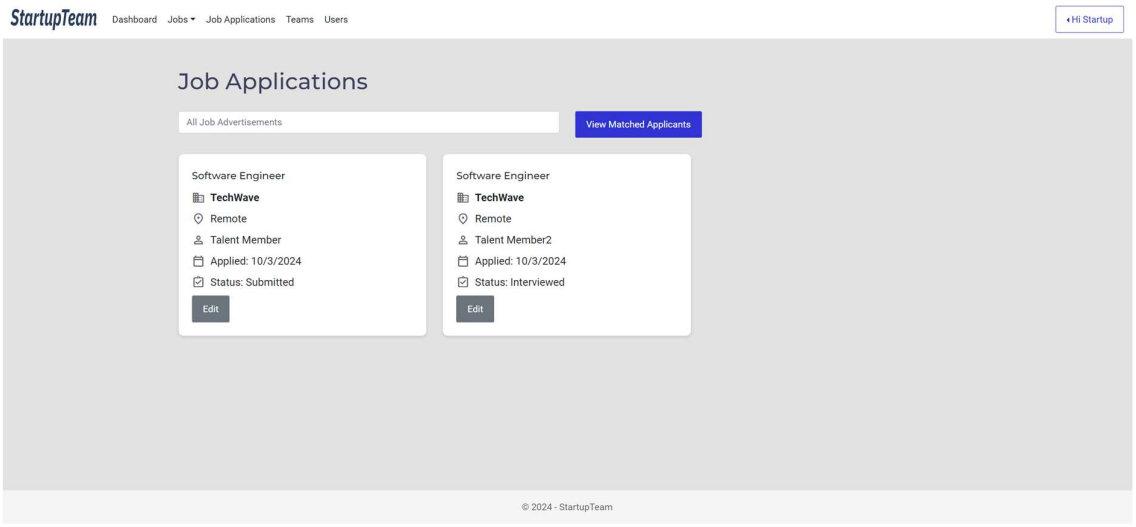


Figure 4.9: Job Applications Management Page

Portfolio Management provided a streamlined process for skilled individuals to create, update, and manage their portfolios, ensuring that their professional details were accessible and visible to others. Figure 4.10 illustrates the Portfolio Item Creation Form, where skilled individuals can add an item to their portfolio by providing the portfolio item title, description, type, technologies, and other relevant details. For details on managing portfolio items, see Appendix L.16 for the Portfolio Management Page.

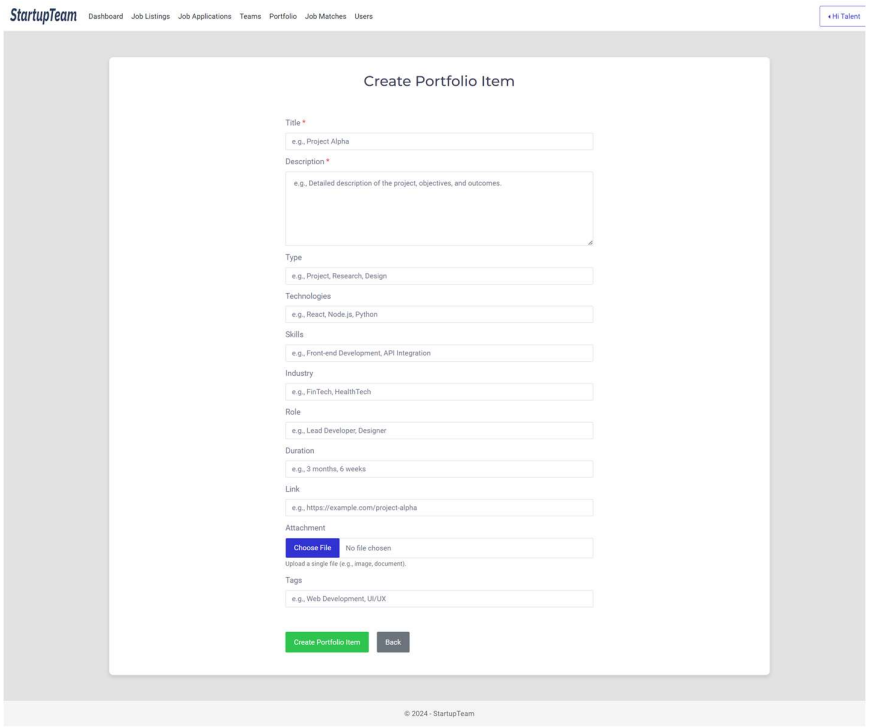


Figure 4.10: Portfolio Item Creation Form

4.3.4 Increment 2: Enhanced Features and Matching Algorithm

The second increment focused on additional high-priority features, including Team Management and Matching Algorithm. This increment enhanced the platform's functionality, allowing for improved team collaboration and user interactions.

Team Management enabled startup founders to create, update, and delete teams. Figure 4.11 illustrates the Team Creation Form, where startup founders can create a startup team by entering the team name and description.

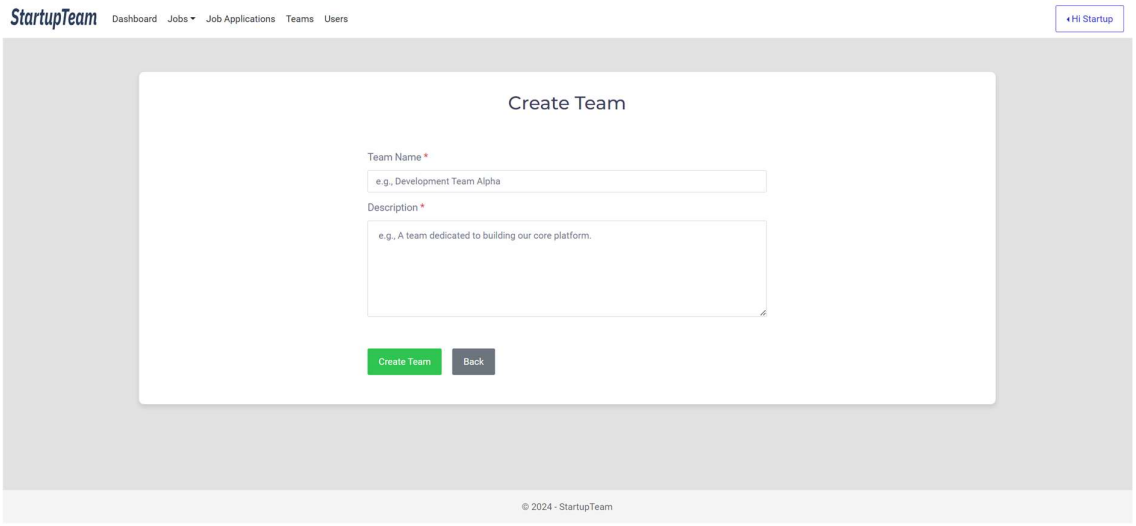


Figure 4.11: Team Creation Form

Once teams are created, founders can manage their members and roles. Figure 4.12 shows the Team Overview Page, where team members, roles, goals, and milestones are displayed. Startup founders could edit these details, while skilled individuals could only view their assigned teams. Additionally, Figure 4.13 depicts the Team Member Addition Form, allowing startup founders to add members by selecting job advertisements and assigning roles to accepted candidates. For more details on team management features, see Appendix L.20 for the Teams Management Page and Appendix L.23 for the Team Role Creation Form.

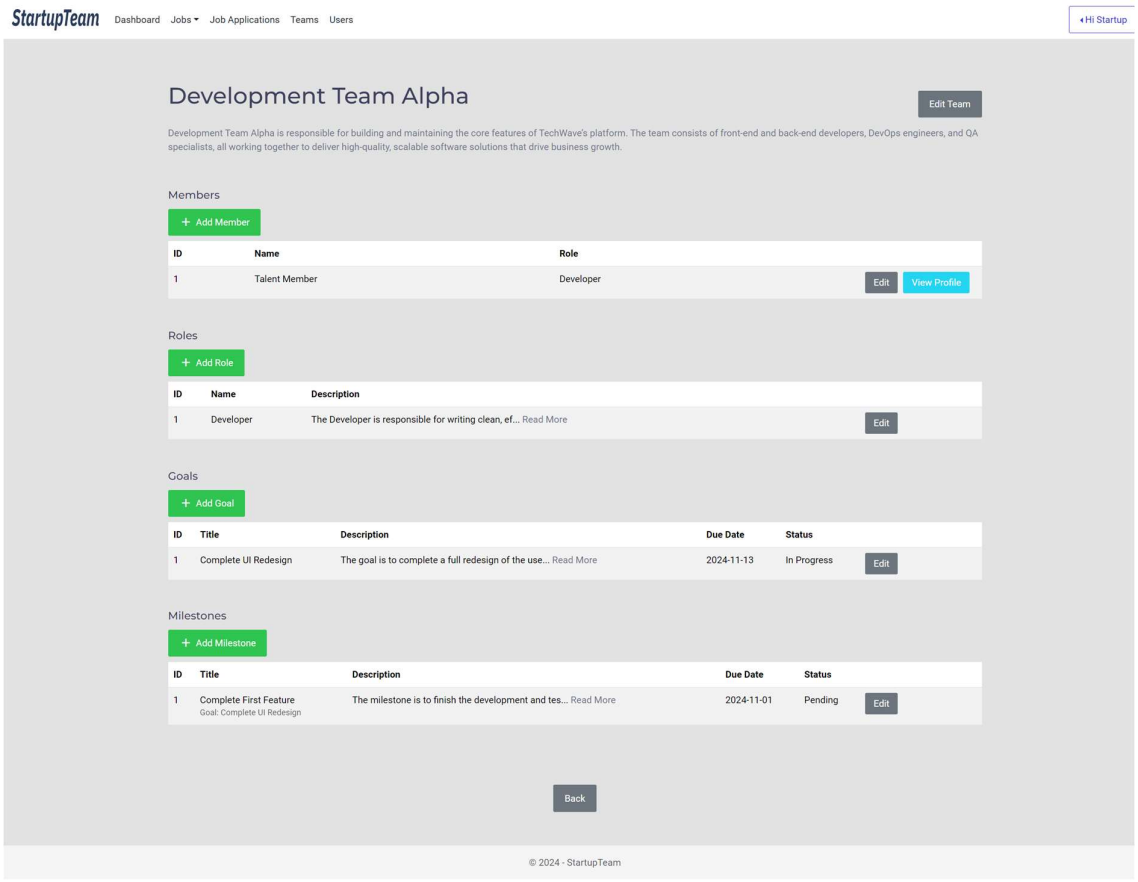


Figure 4.12: Team Overview Page

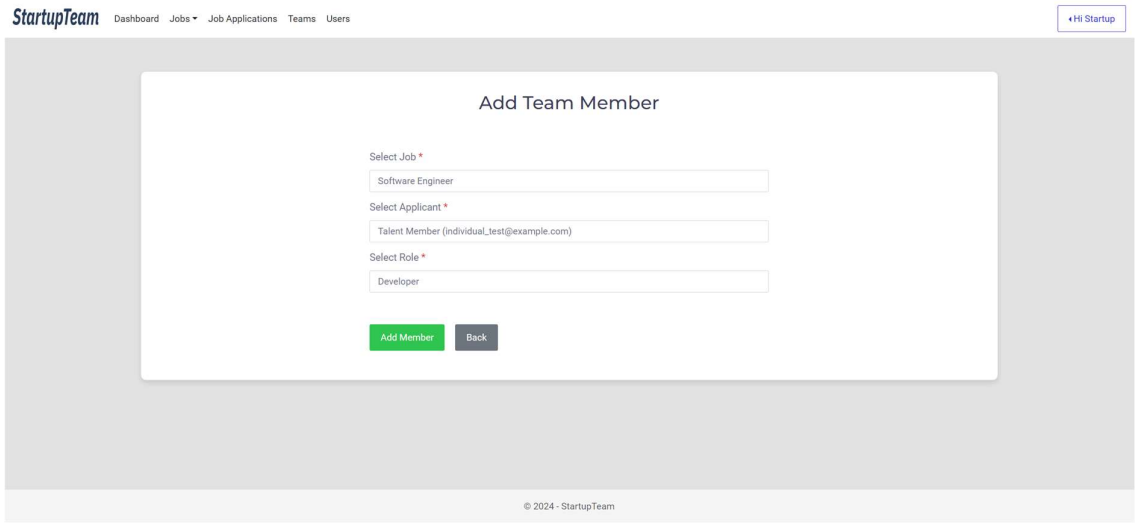


Figure 4.13: Team Member Addition Form

Matching Management enabled startup founders to identify experienced individuals based on job applications and allowed skilled individuals to discover suitable job advertisements matching their portfolios. Figure 4.14 illustrates the Job Applicants Matching Page, where startup founders can view the top matches from skilled individuals who applied for their job postings.

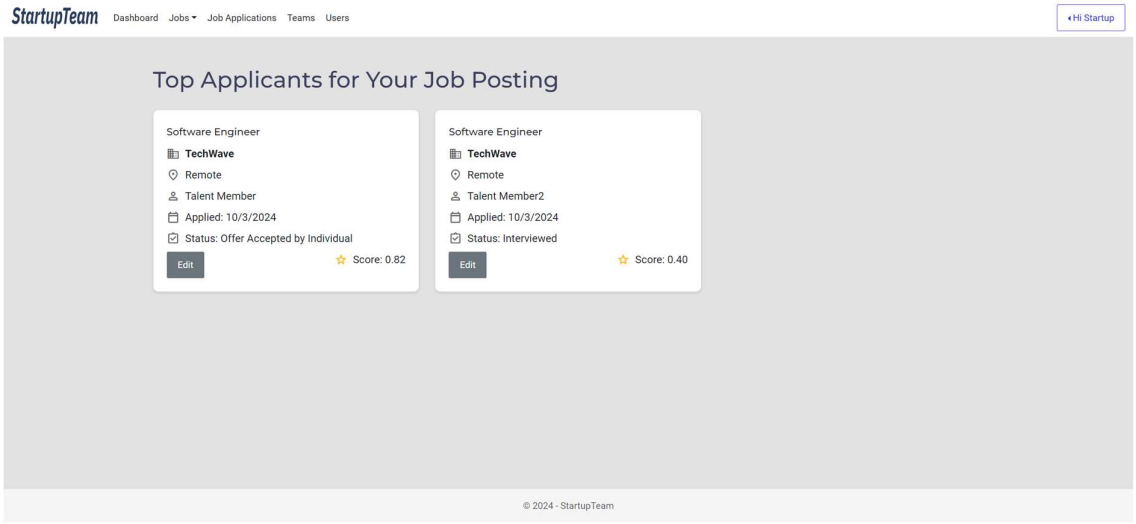


Figure 4.14: Job Applicants Matching Page

Similarly, Figure 4.15 depicts the Job Advertisements Matching Page, enabling skilled individuals to view the top matches from job advertisements that align with their portfolio. Dedicated results of the Matching Algorithm are detailed in Section 4.4.

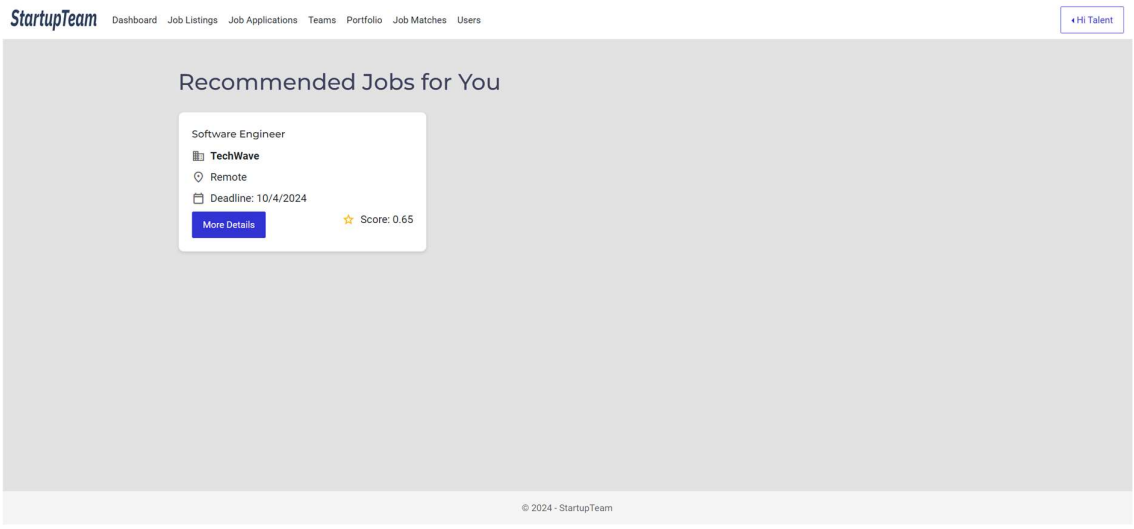


Figure 4.15: Job Advertisements Matching Page

4.3.5 Increment 3: Enhanced Team Management and Refinement

The third increment focused on enhancing the Team Management features and refining the overall system. Startup founders were enabled to manage team goals and milestones. Additionally, Task Management was considered for implementation; however, it was soon realized that it would hinder the timely delivery of the MVP and the overall refinement of the project. As a result, this feature was dropped, and the remaining effort was concentrated on refining the previously developed modules and conducting code refactoring for improved code structure and maintainability.

Team Goals and Milestones Management allowed startup founders to create, update, and delete goals and milestones for their teams. Figure 4.16 illustrates the Milestone Creation Form, where startup founders can define milestones by specifying the title, description, due date, and status, and optionally associate a goal to the milestone to break down the goal into manageable parts. For more details on goal creation, see Appendix L.24 for the Goal Creation Form.

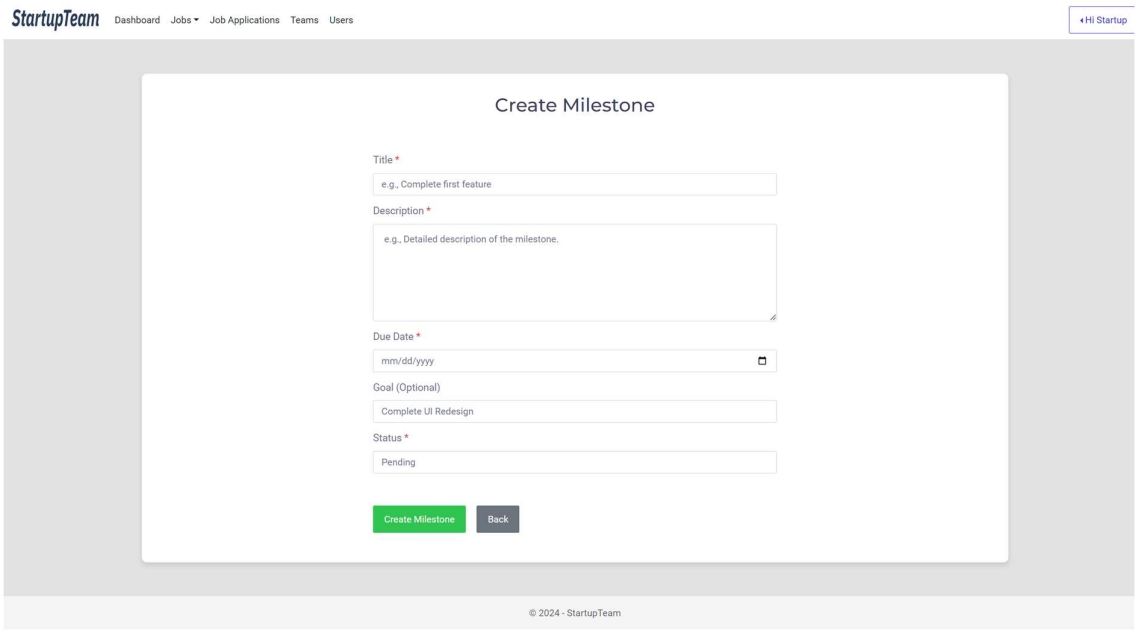


Figure 4.16: Milestone Creation Form

4.4 Matching Algorithm Results

The matching algorithm was developed and refined through a two-phase process: initial standalone testing and subsequent integration into the Matching Management module.

4.4.1 Core Algorithm Implementation

The core of the matching algorithm leverages Term Frequency-Inverse Document Frequency (TF-IDF) and Cosine Similarity to quantify and compare the relevance of job advertisements with portfolios and CVs. The calculation of term frequency (TF) is implemented in the CalculateTermFrequency method, which is detailed in Listing 4.1. This method processes the text to determine how frequently each term appears within a document, normalizing the frequencies to account for document length. Similarly, the calculation of inverse document frequency (IDF) is handled by the CalculateIDF method, as presented in Listing 4.2. This method assesses the importance of each term across all documents by evaluating how common or rare a term is within the entire dataset.

```
// Calculate Term Frequency (TF) for a specific N-Gram (n)
private Dictionary<string, double> CalculateTermFrequency(string document, int n)
{
    var termFrequency = new Dictionary<string, double>();
    var phrases = ExtractPhrases(document, n);

    foreach (var phrase in phrases)
    {
        if (!termFrequency.ContainsKey(phrase))
            termFrequency[phrase] = 0;

        termFrequency[phrase]++;
    }

    // Normalize by the number of terms
    var totalTerms = termFrequency.Values.Sum();
    foreach (var term in termFrequency.Keys.ToList())
    {
        termFrequency[term] /= totalTerms;
    }

    return termFrequency;
}
```

Listing 4.1: CalculateTermFrequency Method

```
// Calculate Inverse Document Frequency (IDF)
private double CalculateIDF(string term)
{
    if (!_idfCache.ContainsKey(term))
    {
        var df = _documentFrequency.ContainsKey(term) ? _documentFrequency[term] : 0;
        var idf = Math.Log((double)_totalDocuments / (df + 1)) + 1;
        _idfCache[term] = idf;
    }

    return _idfCache[term];
}
```

Listing 4.2: CalculateIDF Method

N-Grams and a custom phrase dictionary are employed to capture contextual relationships by generating unigrams and bigrams. The PhraseDictionary detailed in Listing 4.3 includes multi-word phrases relevant to technology-based startups, enhancing matching accuracy. Additionally, the ExtractPhrases

method shown in Listing 4.4 manages the extraction of these phrases and the generation of N-Grams, ensuring that key industry terms are treated as cohesive units during text analysis.

```
private static readonly HashSet<string> PhraseDictionary = new HashSet<string>
{
    // Common phrases for data science and machine learning roles
    "machine learning", "deep learning", "artificial intelligence",
    "data science", "data analysis", "predictive analytics",
    "natural language processing", "reinforcement learning",
    "neural networks", "big data", "data visualization", "data engineering",
    "time series forecasting", "random forest", "support vector machines",

    // Additional phrases . . .
};
```

Listing 4.3: PhraseDictionary

```
// Extract predefined phrases and N-Grams from text
private IEnumerable<string> ExtractPhrases(string text, int n)
{
    var phrases = new List<string>();
    var lowerText = text.ToLower();

    // Extract predefined phrases
    foreach (var phrase in PhraseDictionary)
    {
        if (lowerText.Contains(phrase))
        {
            phrases.Add(phrase);
        }
    }

    // Add N-Grams based on the specified value of N
    phrases.AddRange(Tokenize(text, n));

    return phrases;
}
```

Listing 4.4: ExtractPhrases Method

Data preprocessing improves data quality through text cleaning and tokenization. The Tokenize method, detailed in Listing 4.5, manages tokenization and stop word removal using a predefined set of stop words, as shown in Listing 4.6.

```
// Tokenize and extract N-Grams from text
private IEnumerable<string> Tokenize(string text, int n = 1)
{
    var words = Regex.Split(text.ToLower(), @"\W+")
        .Where(term => !string.IsNullOrEmpty(term)
        && !StopWords.Contains(term)) // Exclude stop words
        .ToList();

    var ngrams = new List<string>();

    for (int i = 0; i < words.Count - n + 1; i++)
    {
        ngrams.Add(string.Join(" ", words.Skip(i).Take(n)));
    }

    return ngrams;
```

```
}
```

Listing 4.5: Tokenize Method

```
// Define a list of common stop words
private static readonly HashSet<string> StopWords = new HashSet<string>
{
    "a", "an", "the", "and", "is", "in", "at", "on", "of", "for", "with", "to",
    "from", "by", "it", "this", "that", // add more as needed
};
```

Listing 4.6: StopWords Set

Feature Extraction and Similarity Computation are performed using TF-IDF vectorization through the Transform method detailed in Listing 4.7. Cosine Similarity scores are calculated using the CalculateAverageMatchingScore method presented in Listing 4.8, which internally calls the CalculateMatchingScore method for different N-Gram sizes (N = 1 for unigrams and N = 2 for bigrams) as shown in Listing 4.9. The actual computation of cosine similarity between vectors is handled by the CosineSimilarity method outlined in Listing 4.10.

```
// TF-IDF Vectorizer that accepts N-Grams
public Dictionary<string, double> Transform(IEnumerable<string> documents, int n)
{
    var termFrequency = new Dictionary<string, double>();
    _totalDocuments = documents.Count();

    // Calculate document frequency (DF)
    foreach (var doc in documents)
    {
        var tf = CalculateTermFrequency(doc, n);
        foreach (var term in tf.Keys)
        {
            if (!_documentFrequency.ContainsKey(term))
                _documentFrequency[term] = 0;

            _documentFrequency[term]++;
        }
    }

    // Calculate TF-IDF
    foreach (var doc in documents)
    {
        var tfidf = CalculateTermFrequency(doc, n).ToDictionary(
            kvp => kvp.Key,
            kvp => kvp.Value * CalculateIDF(kvp.Key)
        );

        foreach (var term in tfidf.Keys)
        {
            if (!termFrequency.ContainsKey(term))
                termFrequency[term] = 0;

            termFrequency[term] += tfidf[term];
        }
    }
    return termFrequency;
}
```

Listing 4.7: Transform Method

```
// Method to calculate average matching score using both unigrams (N = 1) and bigrams
 (N = 2)
public double CalculateAverageMatchingScore(string docA, string docB)
{
    // Calculate unigrams matching score (N = 1)
    var unigramScore = CalculateMatchingScore(docA, docB, 1);

    // Calculate bigrams matching score (N = 2)
    var bigramScore = CalculateMatchingScore(docA, docB, 2);

    // Return the average of unigram and bigram scores
    return (unigramScore + bigramScore) / 2.0;
}
```

Listing 4.8: CalculateAverageMatchingScore Method

```
// Method to calculate matching score between two documents using specific N-Grams
public double CalculateMatchingScore(string docA, string docB, int n)
{
    var vectorA = Transform(new[] { docA }, n);
    var vectorB = Transform(new[] { docB }, n);

    return CosineSimilarity(vectorA, vectorB);
}
```

Listing 4.9: CalculateMatchingScore Method

```
// Calculate Cosine Similarity between two vectors
private double CosineSimilarity(Dictionary<string, double> vectorA, Dictionary<string
, double> vectorB)
{
    var dotProduct = vectorA.Keys.Intersect(vectorB.Keys).Sum(key => vectorA[key] * v
ectorB[key]);
    var magnitudeA = Math.Sqrt(vectorA.Values.Sum(val => val * val));
    var magnitudeB = Math.Sqrt(vectorB.Values.Sum(val => val * val));

    if (magnitudeA == 0 || magnitudeB == 0)
        return 0;

    return dotProduct / (magnitudeA * magnitudeB);
}
```

Listing 4.10: CosineSimilarity Method

4.4.2 Standalone Testing Phase

Before integration, the matching algorithm was tested using a standalone console application. This testing phase evaluated the algorithm's ability to recommend suitable job advertisements to skilled individuals and assist startup founders in identifying the most aligned applicants across various scenarios. Performance metrics—Accuracy, Precision, Recall, and F1 Score—were calculated at thresholds ranging from 0.1 to 0.9. Here, a threshold represents the similarity score cutoff used to determine whether a match is considered relevant. The testing process is shown in Listing 4.11, which systematically iterates through different threshold values to assess the algorithm's performance under varying levels of strictness. For a practical demonstration of how the threshold influences match decisions, refer to Listing 4.12.

```
public static void Main(string[] args)
```

```
{
    double[] thresholds = { 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 };

    foreach (var threshold in thresholds)
    {
        Console.WriteLine($"Testing with Threshold: {threshold:F1}");

        TestDocumentSimilarityMatching(threshold);
        TestIndividualMatchingJobAds(threshold);
        TestJobApplicationsMatchingJobAds(threshold);
        TestComplexPortfolioMatchingJobAds(threshold);
        TestComplexJobApplicationsMatchingJobAds(threshold);
        TestIrrelevantPortfolioItems(threshold);
        TestPartialPortfolioCvAlignment(threshold);

        Console.WriteLine();
    }
}
```

Listing 4.11: Standalone Matching Algorithm Test Implementation

```
// Tests document similarity using a collection of example documents.
public static void TestDocumentSimilarityMatching(double threshold)
{
    // Documents
    var documents = new List<string>
    {
        "Artificial intelligence and machine learning are revolutionizing many indust
ries.",
        "Machine learning and artificial intelligence are key areas of data science
    };

    // Ground truth (expected matches)
    var groundTruth = new List<(int doc1, int doc2, bool isMatch)>{};
    var predictions = new List<bool>();

    var similarityCalculator = new TextSimilarityCalculator();
    foreach (var (doc1, doc2, _) in groundTruth)
    {
        var score = similarityCalculator
         .CalculateAverageMatchingScore(documents[doc1 - 1],documents[doc2 - 1]);

        var isMatch = score > threshold;

        predictions.Add(isMatch);
        Console.WriteLine($"Document {doc1} vs Document {doc2},
                 Score: {score:F6}, Predicted Match: {isMatch}");
    }
    // Extract actual matches from the ground truth for comparison
    var actualMatches = groundTruth.Select(g => g.isMatch).ToList();

    // Calculate metrics based on actual matches vs. predicted matches
    var (accuracy, precision, recall, f1Score) =
         EvaluationMetrics.CalculateMetrics(actualMatches, predictions);
    Console.WriteLine(
        $"Accuracy: {accuracy:F2}, Precision: {precision:F2}, Recall: {recall:F2}, F1
 Score: {f1Score:F2}");
}
```

Listing 4.12: Simplified `TestDocumentSimilarityMatching` Method

The results of these tests are presented below, utilizing the following color-coding for interpretation:

Green: High (≥ 0.90), **Orange**: Moderate (0.70–0.89), **Red**: Low (≤ 0.69), **Gray**: Neutral/expected (edge cases).

4.4.2.1 Simple Document Similarity Matching Test

This test evaluated the algorithm's ability to identify similar documents by comparing known related documents against the ground truth. As shown in Table 4.2, At thresholds 0.1 to 0.3, the algorithm achieved perfect scores, correctly identifying all true matches without false positives or negatives. At thresholds 0.4 and 0.5, accuracy remained high (0.94), but recall dropped to 0.50, indicating some missed matches. Thresholds of 0.6 and above saw significant declines, with the F1 Score falling to 0.00 as the algorithm failed to identify any true matches.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00
0.4	0.94	1.00	0.50	0.67
0.5	0.94	1.00	0.50	0.67
0.6	0.88	0.00	0.00	0.00
0.7	0.88	0.00	0.00	0.00
0.8	0.88	0.00	0.00	0.00
0.9	0.88	0.00	0.00	0.00

Table 4.2: Performance Metrics for Simple Document Similarity Matching Test

4.4.2.2 Skilled Individual Matching with Job Advertisements

This test assessed the algorithm's effectiveness in matching skilled individuals' portfolios with relevant job advertisements. Portfolios with specific skills were compared against job advertisements requiring those competencies. At thresholds 0.1 to 0.4, the algorithm achieved perfect performance across all metrics, as detailed in Table 4.3. However, at thresholds 0.5 and 0.6, accuracy decreased to 0.83, and recall dropped to 0.50, indicating some matches were missed, while precision remained at 1.00 and F1 Score was 0.67. For thresholds 0.7 and above, performance declined further, with accuracy falling to 0.67 and the F1 Score dropping to 0.00 due to no true positive matches.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00
0.4	1.00	1.00	1.00	1.00

0.5	0.83	1.00	0.50	0.67
0.6	0.83	1.00	0.50	0.67
0.7	0.67	0.00	0.00	0.00
0.8	0.67	0.00	0.00	0.00
0.9	0.67	0.00	0.00	0.00

Table 4.3: Performance Metrics for Skilled Individual Matching with Job Advertisements

4.4.2.3 Job Applications Matching with Job Advertisements

In this test, the algorithm matched job applications—including CVs and portfolios—from skilled individuals with job advertisements. At thresholds 0.1 to 0.5, the algorithm achieved perfect scores, as shown in Table 4.4. At threshold 0.6, accuracy decreased to 0.75 and recall to 0.50. Thresholds 0.7 and above saw performance decline sharply, with accuracy dropping to 0.50 and F1 Score falling to 0.00 due to missed matches.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00
0.4	1.00	1.00	1.00	1.00
0.5	1.00	1.00	1.00	1.00
0.6	0.75	1.00	0.50	0.67
0.7	0.50	0.00	0.00	0.00
0.8	0.50	0.00	0.00	0.00
0.9	0.50	0.00	0.00	0.00

Table 4.4: Performance Metrics for Job Applications Matching with Job Advertisements

4.4.2.4 Complex Matching Scenarios

The algorithm was evaluated with detailed portfolios and job advertisements from both perspectives: individuals matching job advertisements and job applications matching job advertisements. At thresholds 0.1 to 0.5, the algorithm maintained perfect scores across all metrics, as evidenced in Table 4.5 and Table 4.6. At threshold 0.6, accuracy decreased to 0.75 and recall to 0.50, with an F1 Score of 0.67. Thresholds 0.7 and above resulted in significant drops, with the algorithm failing to identify matches.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00

0.4	1.00	1.00	1.00	1.00
0.5	1.00	1.00	1.00	1.00
0.6	0.75	1.00	0.50	0.67
0.7	0.50	0.00	0.00	0.00
0.8	0.50	0.00	0.00	0.00
0.9	0.50	0.00	0.00	0.00

Table 4.5: Performance Metrics for Complex Portfolio Matching Job Advertisements

Threshold	Accuracy	Precision	Recall	F1 Score
0.1	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00
0.4	1.00	1.00	1.00	1.00
0.5	1.00	1.00	1.00	1.00
0.6	0.75	1.00	0.50	0.67
0.7	0.50	0.00	0.00	0.00
0.8	0.50	0.00	0.00	0.00
0.9	0.50	0.00	0.00	0.00

Table 4.6: Performance Metrics for Complex Job Applications Matching Job Advertisements

4.4.2.5 Edge Cases

An edge case involved portfolios completely unrelated to job advertisements. As shown in Table 4.7 below, the algorithm achieved perfect accuracy (1.00) across thresholds 0.1 to 0.9, but precision, recall, and F1 Score were all 0.00, as expected since there were no true matches.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1 to 0.9	1.00	0.00	0.00	0.00

Table 4.7: Testing Irrelevant Portfolio Items

Another edge case tested mismatches between CV content and portfolio—when the CV aligned with the job advertisement but the portfolio did not. At thresholds 0.1 and 0.2, the algorithm incorrectly matched based on CV alignment, resulting in an accuracy of 0.00. From threshold 0.3 onwards, accuracy improved to 1.00, but precision, recall, and F1 Score remained at 0.00 due to no true positive matches. Detailed metrics are shown in Table 4.8.

Threshold	Accuracy	Precision	Recall	F1 Score
0.1 to 0.2	0.00	0.00	0.00	0.00
0.3 to 0.9	1.00	0.00	0.00	0.00

Table 4.8: Testing Mismatched CV and Portfolio Content

4.4.3 Final Threshold Selection

After these tests, a final threshold of 0.3 was selected. At this threshold, the algorithm consistently achieved perfect scores across various scenarios, effectively identifying relevant matches with minimal errors. This threshold balanced sensitivity and specificity, providing reliable matching results across different scenarios.

4.4.4 Integration into the Platform

After the standalone testing and threshold optimization, the matching algorithm was integrated into the platform's Matching Management module. The MatchesController and MatchService are central to the matching functionalities. The controller handles API requests, as shown in Listing 4.13, while the service layer, detailed in Listing 4.14, contains the business logic for calculating similarity scores and retrieving top matches. The MatchService utilizes various attributes from job advertisements—such as job title, job description, required skills, key technologies, industry, experience, education, and startup description—along with portfolio and CV text from applicants. It calculates similarity scores, orders the results by descending scores, and returns only the top N results (currently set to 10).

```
[HttpGet("founder/matched-applicants")]
[Authorize(Roles = RoleConstants.StartupFounder)]
public async Task<IActionResult> GetTopMatchedApplicants([FromQuery] Guid jobAdId)
{
    int topN = 10;

    var userId = ClaimsHelper.GetUserId(User);

    if (userId == null)
    {
        return BadRequest(new ApiResponse<object>()
        {
            Success = false,
            Message = "User ID is missing or invalid."
        });
    }

    var matchedApplicants =
        await _matchService.GetTopMatchedApplicantsForJob(jobAdId, topN);

    if (!matchedApplicants.Any())
    {
        return NotFound(new ApiResponse<object>()
        {
            Success = false,
            Message = "No matched applicants found."
        });
    }

    return Ok(new ApiResponse<object>()
    {
        Data = matchedApplicants
    });
}
```

Listing 4.13: MatchesController Implementation

```
public async Task<IEnumerable<JobApplicationDto>> GetTopMatchedApplicantsForJob(Guid
jobAdId, int topN)
{
    var jobAd =
        await _jobManagementDbContext.JobAdvertisements.FindAsync(jobAdId);

    if (jobAd == null)
    {
        return new List<JobApplicationDto>();
    }

    var jobApplications =
        await _jobManagementDbContext.JobApplications
            .Include(ja => ja.JobAdvertisement)
            .Where(ja => ja.JobAdvertisement!.UserId == jobAd.UserId)
            .Where(ja => ja.JobAdvertisementId == jobAd.Id)
            .ToListAsync();

    var portfolios = new Dictionary<Guid, Portfolio?>();
    var threshold = 0.3;
    var predictions = new List<(JobApplicationDto applicant, double score)>();

    // Calculate similarity for each applicant and fetch portfolio data
    foreach (var jobApplication in jobApplications)
    {
        portfolios[jobApplication.UserId] =
            await _portfolioManagementDbContext.Portfolios
                .Include(p => p.PortfolioItems)
                .FirstOrDefaultAsync(p => p.UserId == jobApplication.UserId);

        var portfolio = portfolios[jobApplication.UserId];

        string portfolioText = string.Empty;

        if (portfolio != null)
        {
            portfolioText = string.Join(" ",
                portfolio.PortfolioItems.Select(
         item => $"{item.Title} {item.Description} {item.Skills}
         {item.Technologies}  {item.Industry}"));
        }

        var jobAppText = $"{jobApplication.CVTextContent} {portfolioText}";

        var jobAdText = $"{jobAd.JobTitle} {jobAd.JobDescription}
         {jobAd.RequiredSkills} {jobAd.KeyTechnologies} {jobAd.Industry}
         {jobAd.Experience} {jobAd.Education} {jobAd.StartupDescription}";

         var score = _similarityCalculator.CalculateAverageMatchingScore(
         jobAppText, jobAdText);

        if (score > threshold)
        {
            // Fetch the individual for each job application
            var individual = await _userService.FindByIdAsync(jobApplication.UserId);

            predictions.Add((new JobApplicationDto
            {
                Id = jobApplication.Id,
                StartupName = jobApplication.JobAdvertisement!.StartupName,
                JobTitle = jobApplication.JobAdvertisement.JobTitle,
                JobLocation = jobApplication.JobAdvertisement.JobLocation
```

```
                        ?? nameof(JobLocationType.Remote),
                ApplicationDate = jobApplication.ApplicationDate,
                Status = jobApplication.Status.ToFriendlyString(),
                IndividualFullName = individual == null ?
                    string.Empty : $"{individual.FirstName} {individual.LastName}",
                Score = score
            }, score));
        }
    }

    // Return top N matched applicants sorted by score
    return predictions
        .OrderByDescending(p => p.score)
        .Take(topN)
        .Select(p => p.applicant);
}
```

Listing 4.14: MatchService Implementation

Finally, the Matching Management is registered within the application's dependency injection container to enable its usage via the central API, as shown in Listing 4.15.

```
public static IServiceCollection AddMatchingManagement(
        this IServiceCollection services)
{
    services.AddScoped<TextSimilarityCalculator>();
    services.AddScoped<IMatchService, MatchService>();

    return services;
}
```

Listing 4.15: Dependency Injection Configuration for Matching Management

4.5 Deployment Phase Results

The deployment on Microsoft Azure was completed, ensuring accessibility, reliability, and scalability. Figure 4.17 illustrates the Kubernetes cluster configuration, demonstrating the management of the frontend and backend components. Azure Kubernetes Service (AKS) enabled automated scaling and load balancing. The Docker containers for the frontend and backend were stored in Azure Container Registry (ACR) within separate repositories, as shown in Figure 4.18.

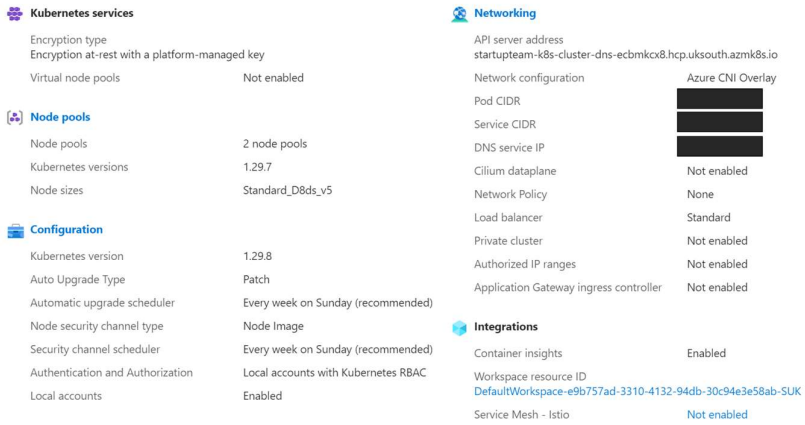


Figure 4.17: Azure Kubernetes Service Cluster

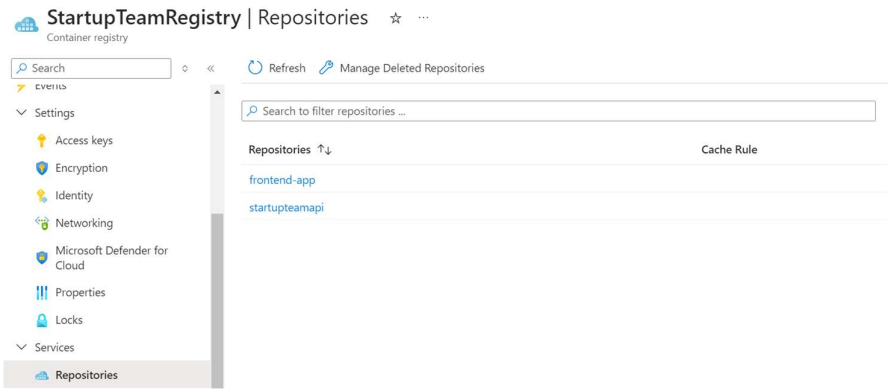


Figure 4.18: Azure Container Registry

The status of Kubernetes pods, services, and nodes, retrieved using kubectl commands, confirmed that the backend and frontend were running without issues on separate pods, ensuring modularity and ease of maintenance, as depicted in Figure 4.19. Azure SQL Database was utilized for secure and scalable data storage, with transparent encryption and automated backups maintaining data integrity, as shown in Figure 4.20.

```
[redacted] kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE     IP            NODE
 NOMINATED NODE   READINESS GATES
backend-app-57df75d5d6-zf92p    1/1     Running   0          3d10h   [redacted]    aks-userpool-38432996-vmss00000l
 <none>           <none>
frontend-app-7cb46fcd89-x7qpw   1/1     Running   0          3d10h   [redacted]    aks-userpool-38432996-vmss00000l
 <none>           <none>
[redacted] kubectl get svc -o wide
NAME               TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)                      AGE   SELECTOR
backend-service    LoadBalancer   10.0.129.139   51.142.219.150   80:30951/TCP,81:31071/TCP    14d   app=backend-app
frontend-service   LoadBalancer   10.0.194.248   20.162.179.59    80:31940/TCP                 14d   app=frontend-app
kubernetes         ClusterIP      10.0.0.1       <none>           443/TCP                      15d   <none>
[redacted] kubectl get nodes -o wide
NAME                                STATUS   ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
  KERNEL-VERSION      CONTAINER-RUNTIME
aks-agentpool-38432996-vmss00000k   Ready    <none>   69m   v1.29.7   [redacted]    <none>        Ubuntu 22.04.4 LTS
  5.15.0-1071-azure   containerd://1.7.20-1
aks-agentpool-38432996-vmss00000l   Ready    <none>   69m   v1.29.7   [redacted]    <none>        Ubuntu 22.04.4 LTS
  5.15.0-1071-azure   containerd://1.7.20-1
aks-userpool-38432996-vmss00000k    Ready    <none>   69m   v1.29.7   [redacted]    <none>        Ubuntu 22.04.4 LTS
  5.15.0-1071-azure   containerd://1.7.20-1
aks-userpool-38432996-vmss00000l    Ready    <none>   69m   v1.29.7   [redacted]    <none>        Ubuntu 22.04.4 LTS
  5.15.0-1071-azure   containerd://1.7.20-1
```

Figure 4.19: Kubernetes Pods, Services, and Nodes Status

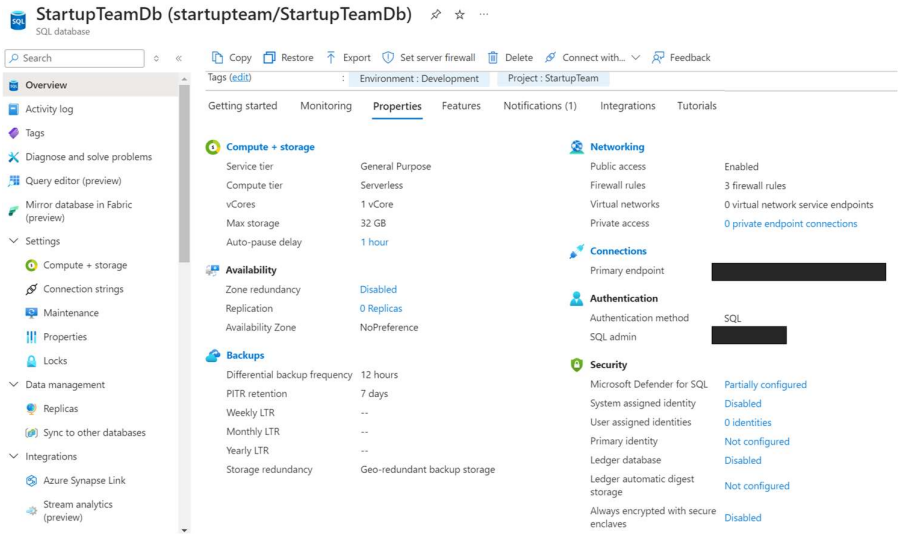


Figure 4.20: Azure SQL Database

File management was achieved by organizing files into three separate blob containers for job application attachments, portfolio attachments, and profile pictures, each file uniquely identified using GUIDs, as illustrated in Figure 4.21. This organization streamlined file retrieval and reduced server load.

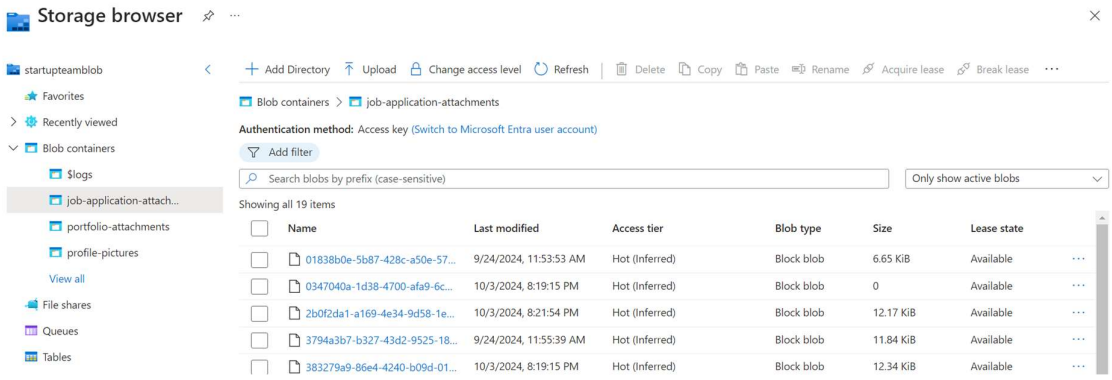


Figure 4.21: Azure Blob Storage

During the deployment phase, the system operated stably, and basic functionality tests indicated that complex queries across multiple modules ran without noticeable delays, highlighting the platform's efficiency in handling typical workloads. Performance monitoring via Container Insights provided real-time data on resource usage, showing stable performance with average CPU utilization at 1.57% and memory utilization at 4.44%, as demonstrated in Figure 4.22.

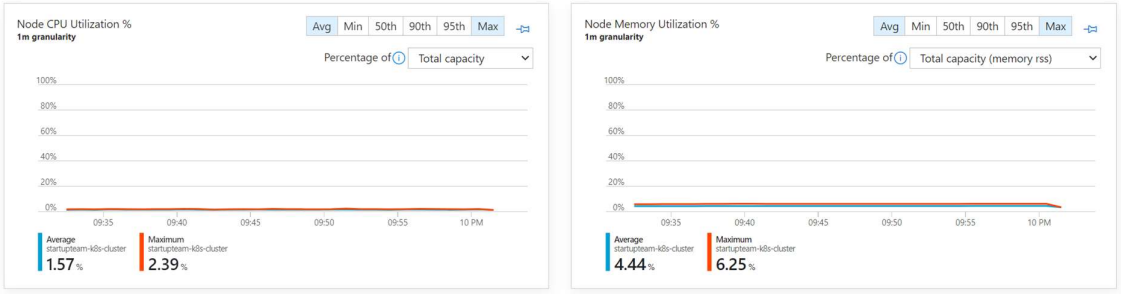


Figure 4.22: Performance Monitoring

Security measures were implemented, with a Virtual Network (VNet) configured to manage internal communications within a secure environment, as shown in Figure 4.23. Additionally, a Network Security Group (NSG) was established to regulate traffic, with defined inbound and outbound rules controlling access to the platform through specific ports and protocols, as depicted in Figure 4.24.

Network configuration

Address space	[redacted]
Subnets	1
DNS servers	Azure provided DNS service
Virtual network ID	[redacted]

Connectivity

BGP virtual network community	Configure
BGP regional community	-
Connected devices	4
Flow timeout	Configure
Peerings	Add peerings

Figure 4.23: Virtual Network (VNet) Configuration Overview

Priority	Name	Port	Protocol	Source	Destination	Action
Inbound Security Rules						
500	k8s-azure-lb_allow_IPv...	80	Tcp	Internet	[redacted]	Allow
501	k8s-azure-lb_allow_IPv...	80,81	Tcp	Internet	[redacted]	Allow
502	k8s-azure-lb_allow_IPv...	443,80	Tcp	Internet	[redacted]	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalanc...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny
Outbound Security Rules						
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

Figure 4.24: Network Security Group (NSG) Inbound and Outbound Rules

4.6 Incremental Automated Testing Results

All 272 backend unit tests passed successfully, confirming the functionality of controllers, services, and validation logic across all modules, as shown in Figure 4.25. This validated that the backend components operated as expected, ensuring robustness and stability. For instance, Listing 4.16 demonstrates a unit test that validates the successful update of a team. The test ensures that the `UpdateTeam` method returns an `OkObjectResult` when the team is updated correctly, confirming that the backend logic works as expected when handling team updates.

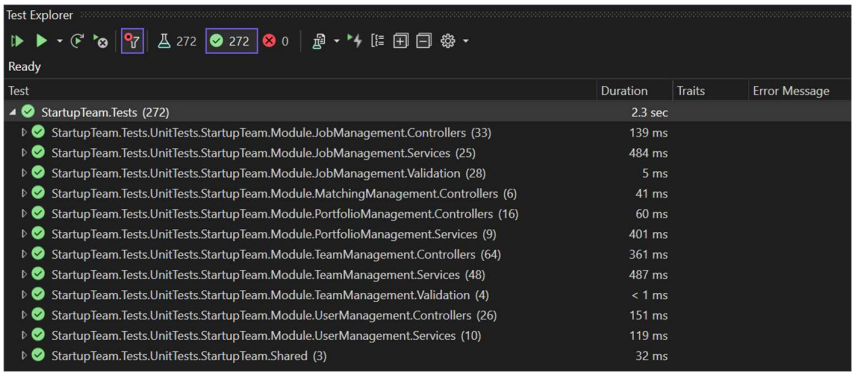


Figure 4.25: Test Explorer showing the successful passing of 272 backend unit tests

```
[Fact]
public async Task UpdateTeam_ShouldReturnOk_WhenTeamUpdatedSuccessfully()
{
    // Arrange
    var teamFormDto = new TeamFormDto { Id = Guid.NewGuid() };
    MockUser(_controller, Guid.NewGuid(), true);

    _teamAuthorizationServiceMock
        .Setup(service => service.AuthorizeTeamAction(
                teamFormDto.Id!.Value, It.IsAny<ClaimsPrincipal>()))
        .ReturnsAsync(new OkResult());

    _teamServiceMock.Setup(service => service.UpdateTeamAsync(teamFormDto))
        .ReturnsAsync(true); // Team updated successfully

    // Act
    var result = await _controller.UpdateTeam(teamFormDto.Id!.Value, teamFormDto);

    // Assert
    var okResult = Assert.IsType<OkObjectResult>(result);
    var apiResponse = Assert.IsType<ApiResponse<object>>(okResult.Value);
    Assert.Equal("Team updated successfully.", apiResponse.Message);
}
```

Listing 4.16: Backend Unit Test for Successful Team Update

A total of 149 frontend component tests confirmed that individual UI elements functioned as expected, validating interactions and rendering to ensure the frontend's reliability and user experience. Figure 4.26 illustrates the successful passing of these tests in the test explorer. An illustrative example is shown

in Listing 4.17, where a test case simulates the submission of a job application update form. This test verifies that the form correctly submits the appropriate data, triggers an API call with the expected parameters, displays a success message, and reloads the page upon successful submission.

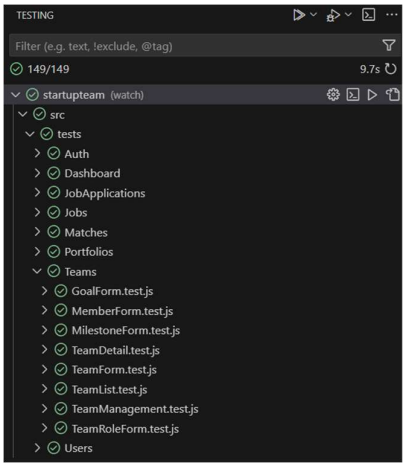


Figure 4.26: Test Explorer displaying the successful passing of 149 frontend component tests

```
test('handles successful form submission and displays success message', async () => {
        // Mock API responses
        fetchData
            .mockResolvedValueOnce({data: { id: '123', /* ...other data... */ }})
            .mockResolvedValueOnce({data: { jobTitle: 'Software Engineer'}});
        fetchData.mockResolvedValueOnce({});
       // Mock window.alert and window.location.reload
        global.alert = jest.fn();
        const originalLocation = window.location;
        delete window.location;
        window.location = {...originalLocation, reload: jest.fn()};
        render(
            <MemoryRouter>
                <JobApplicationForm />
            </MemoryRouter>
        );
        await waitFor(() => {
            expect(screen.getByLabelText(/New Application Status/i)).toBeInTheDocument();
        });
        // Fill in the form fields
        fireEvent.change(screen.getByLabelText(/New Application Status/i), {
            target: { value: 'InterviewScheduled' },
        });
        fireEvent.change(screen.getByLabelText(/Interview Date/i), {
            target: { value: '2099-09-20' },
```

```
        });
        // Submit the form
        fireEvent.click(screen.getByRole('button', { name: /Update Application/i }));
        // Wait for fetchData to be called
        await waitFor(() => {
            expect(fetchData).toHaveBeenCalledWith(
                '/jobapplications/123',
                expect.objectContaining({
                    method: 'PUT',
                    body: expect.objectContaining({
                        status: 'InterviewScheduled',
                        interviewDate: '2099-09-20',
                    }),
                })
            );
        });
        // Assert success alert and page reload
        expect(global.alert).toHaveBeenCalledWith('Job application updated successfully!');
        expect(window.location.reload).toHaveBeenCalled();
        // Restore mocks
        global.alert.mockRestore();
        window.location = originalLocation;
});
```

Listing 4.17: Frontend Test for Job Application Form Submission

4.7 Usability Testing Results

4.7.1 Participant Profiles

The usability testing involved five participants with diverse backgrounds. Participant 1 is experienced in managing technical systems and founding startups, offering insights into technical and leadership challenges. Participant 2 has a background in cybersecurity projects and team leadership within startup environments. Participant 3 is a social media manager specializing in connecting talents and enhancing the online presence of startups. Participant 4 brings expertise in hybrid cloud solutions and co-founding startups, responsible for technology implementations and team management. Participant 5 is a software engineer and founder specializing in cloud technologies, full-stack development, and automation systems.

4.7.2 Job Posting and Application Process (Task 1)

Participants frequently confused the “Job Listings” and “Manage Jobs” sections in the startup founder panel when attempting to manage job advertisements. Specifically, Participants 1, 3, and 5 initially accessed “Job Listings” instead of “Manage Jobs”. Participant 1 found the “View My Job Postings” button unintuitive and initially experienced confusion when trying to locate the “Job Applications”

section, while Participant 5 had difficulty finding the “Log Out” button. User interface issues, such as unclear labels and navigation elements, contributed to these challenges. For instance, Participant 2 mistakenly perceived non-clickable labels like “CV” and “Cover Letter” as hyperlinks. Additionally, Participant 1 expected the “Job Details” section to precede “Startup Information” when viewing job advertisements and preferred startup details to be stored in the profile to avoid repetitive data entry. Participant 4 anticipated creating the startup profile separately during account setup.

Workflow and team management processes posed challenges for participants, as many attempted to add team members without first defining roles or accepting job applications, leading to confusion. Participants realized during the process that defining roles and accepting applicants were necessary steps before adding team members. For instance, Participant 1 initially thought they could add members while creating the team but discovered that members must be added separately. Similarly, Participant 3 faced difficulties adding members post-application acceptance without defined roles. Participant 5 also encountered difficulties adding members after accepting offers, as the process still required predefined roles.

Issues with unsupported file formats during job application were reported in error handling and feedback. Participants 1 and 2 encountered errors when uploading files in unsupported formats and suggested clearly labelling acceptable file types to prevent confusion. Despite these challenges, participants provided positive feedback on certain aspects of the platform. Participants appreciated the minimal requirements for job applications and found certain features user-friendly. Participants 1 and 2 valued the message indicating a job had already been applied for and the inclusion of detailed information on the job application detail page. Participant 3 received appropriate prompts when correcting an invalid founding year entry, as did Participant 1 for the team size entry. Moreover, participant 2 valued the minimal details required for job applications and appreciated the date picker input for the “Application Deadline” field.

Survey results corroborated these findings, with participants rating the ease of posting a job and the clarity and effectiveness of the candidate review process highly, both averaging 4.6 out of 5. Additionally, all participants confirmed they were able to find and apply for jobs without difficulties (see Appendix P.1 and Appendix Q.3).

4.7.3 Portfolio Creation and Application Review (Task 2)

Participants were generally able to create and manage portfolios, though some experienced initial confusion, particularly regarding user categorization and the portfolio process. Participant 1, for instance, was unclear about the purpose of a portfolio item and found adding a portfolio after applying for a job confusing, suggesting prompts to guide users in uploading portfolios before applying and a

checkbox for founders to make portfolios mandatory. Participant 2 initially confused the portfolio section with the profile section but managed to add a portfolio item without issues thereafter. Participant 3 created a portfolio item successfully and accessed it from various sections, including “Team”, “Users”, and “Job Applications”. Participant 4 completed the portfolio creation process without difficulty and suggested guiding candidates to create multiple portfolio items with diverse examples, even if not directly related to job postings. Participant 5 found the portfolio creation process user-friendly but suggested enhancing the visibility of portfolio links within job applications.

Participants expected the “Users” section to display only company-related individuals, such as team members or applicants, rather than all platform users. Participants 1, 2, 3, and 4 found the inclusion of unrelated users, especially in the skilled individual panel, unnecessary. Despite this, all participants acknowledged the importance of reviewing portfolios during application evaluations and successfully accessed candidate portfolios. Participant 1 highlighted that portfolios provided valuable insights into applicants’ histories, while Participants 2 and 3 found portfolios useful for decision-making. Participant 4 viewed portfolios directly from the application section without issues, while Participant 5 initially faced difficulty locating the portfolio link in job applications but resolved the issue.

Survey insights revealed that participants rated the ease of updating or adding portfolio items highly, with an average score of 4.8 out of 5, and the portfolio feature was considered useful for hiring decisions by startup founders, averaging 4.4 out of 5. Overall, participants found the ability to review and integrate candidate portfolios a valuable component of the platform (see Appendices P.2, Q.4, and R.3).

4.7.4 Interacting with the Matching Algorithm (Task 3)

Participants recognized that job matches were influenced by their portfolios and observed that updating portfolio content impacted match quality. However, some did not initially understand the inclusion of CV content in candidate matching from the startup founder panel. Participant 1 tested changes to their portfolio, noted improvements in the matching score, and emphasized the importance of portfolio content. Similarly, Participant 2 updated their portfolio with technologies similar to those in the job advertisement but did not receive the expected match, while Participant 3 added an irrelevant item to their portfolio, observing a drop in match quality. Participant 4 confirmed that updating the portfolio improved job matches and Participant 5 observed better alignment with job advertisements after modifying their portfolio. User interface challenges included confusion with the select box for job advertisements in the matching feature, particularly for founders. Participants initially overlooked the select box and were unsure how to view candidate matches, though they understood it after some interaction. Despite these challenges, participants valued the accuracy of job matches.

Participants expressed a desire for greater control over matching criteria, including the ability to prioritize the CV, portfolio, or both, and granularly choose fields such as skills or years of experience. Participant 1 wanted the ability to choose whether the CV, portfolio, or both should be used for matching and to specify which fields, such as skills or years of experience, to consider. Participant 2 desired options to select matching criteria to avoid skewed results and believed that overlapping content in the CV and portfolio should strengthen the matching score. Participant 3 preferred to prioritize matching criteria rather than using both the CV and portfolio by default and desired the ability to upload CVs both separately and during job applications. Participant 4 expected user-controlled settings for matching criteria, favouring default matching based on either the CV or portfolio. Participant 5 believed the CV might be more reliable for matching and sought transparency in how matches are calculated, a desire shared by other participants.

Survey results supported these observations, with the accuracy of job matches based on skills and portfolio updates receiving an average rating of 4.6 out of 5, and the alignment of candidate recommendations with job postings also averaging 4.6 out of 5. Participants confirmed that updating their portfolios improved the accuracy of job matches, though some expressed a need for more control and transparency in the matching process (see Appendices P.3, Q.5, and R.4).

4.7.5 Roles, Goals, and Milestones Setup (Task 4)

Participants generally found the process of creating goals and milestones straightforward and encountered no major issues. They understood the concept of breaking down goals into milestones. Participant 1 noted that this hierarchy was logical and effective for managing startup teams. Participant 2 suggested that accountability could be further assigned to goals and milestones, with individuals responsible for each, and perceived value in the current structure. In terms of role assignment, most participants did not encounter significant issues. Participant 1 expected that only one role could be assigned per team member. However, Participants 3 and 4 desired the ability to assign multiple roles to a single team member but were unsure how to do so. Particularly, Participant 4 was unaware that assigning multiple roles was possible. Participant 1 also suggested that roles should precede the members section in the team view.

Regarding user interface and navigation, the process was found to be user-friendly, with no major navigation issues reported. Participant 1 found the “View Profile” option from the members table useful. Participant 5 expected a “Teams” button on the dashboard for easier access and experienced some confusion when trying to find teams from the skilled individual panel, but found the process straightforward after initial familiarization. Participants also discussed the hierarchy of goals and milestones. While most accepted the current structure, Participant 4 expected milestones to contain

goals, suggesting an inverse hierarchy compared to the current design. They proposed a more detailed hierarchy to better manage complex projects and team responsibilities, suggesting that milestones should contain goals, which in turn could be broken down into objectives and tasks. Participant 3 initially misunderstood that milestones could be defined independently of goals but later realized they could be established separately.

Survey insights showed high ratings for team management features, with the ease of creating a new team scoring an average of 5.0 out of 5, while adding new members and assigning roles each averaged 4.8 out of 5. The ease of setting goals and milestones, along with their clarity in presentation, received slightly lower averages of 4.4 out of 5 (see Appendix P.4 and Appendix Q.6).

4.7.6 Additional Feedback and Suggestions

Participants provided several suggestions to enhance the platform's functionality and user experience:

- **Job Posting Enhancements:** Participant 1 suggested adding hiring person details to job postings and integrating startup profiles within user profiles to streamline information entry.
- **Portfolio Features:** Participant 1 recommended making portfolio details mandatory for applicants to streamline the review process. Participant 3 suggested adding a cover letter option to the portfolio to enhance applications. Participant 4 emphasized guiding candidates to create multiple diverse portfolio items.
- **Matching Algorithm Improvements:** Participants recommended allowing users to prioritize matching criteria (CV, portfolio, or both) and to specify which fields to consider. Transparency in how matching scores are calculated and the factors influencing them was also desired.
- **Team Management Enhancements:** Participants suggested integrating task management features to track progress, clarifying the existing ability to assign multiple roles to team members, and implementing a more granular hierarchy for goals and milestones, such as breaking goals into objectives and tasks with progress indicators. Additionally, improving the visibility of the "Teams" button and enhancing communication features within team management were recommended.

Overall, participants highly appreciated the integration of team management and recruitment features, valuing the platform's simplicity, user-friendliness, and comprehensive functionality. They emphasized the importance of intuitive design and suggested enhancements to improve the design and workflow (see Appendix R.8).

4.8 Security Considerations Results

Post-deployment, configuring HTTPS was impossible due to the absence of a registered domain name for the Kubernetes cluster, which used a public IP address. This restricted the platform to HTTP connections after deployment, as depicted in Figure 4.27. Additionally, Google Authentication failed post-deployment because Google restricts authentication requests from public IPs without a domain name, as shown in Figure 4.28.

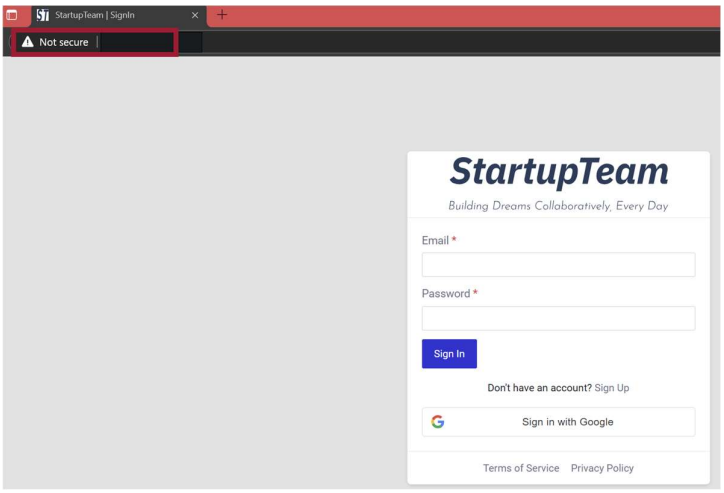


Figure 4.27: HTTPS Configuration Failure Due to Public IP Limitations

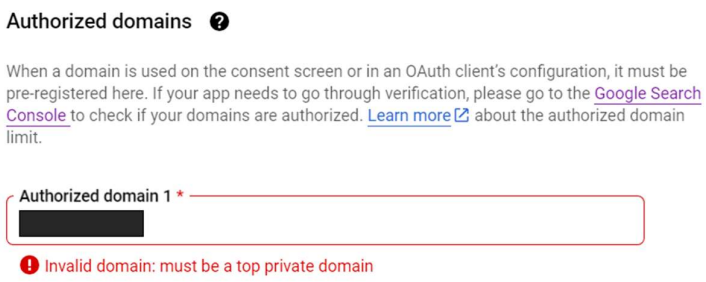


Figure 4.28: Google Authentication Failure Due to Public IP Restrictions

Despite these issues, other security measures were successfully implemented. JSON Web Tokens (JWTs) were used for secure session management and role-based access control. JWTs were generated in the backend, embedding user claims and roles, as demonstrated in Listing 4.18. On the frontend, JWTs were decoded to maintain user sessions and enforce role-based access, as shown in Listing 4.19. Role-based authorization ensured that only users with specific roles could perform certain actions, such as team creation restricted to the "StartupFounder" role, illustrated in Listing 4.20. Moreover, Azure SQL Server with Transparent Data Encryption (TDE) safeguarded sensitive data.

```
public class JwtHelper : IJwtHelper
{
    private readonly JwtSettings _jwtSettings;

    public JwtHelper(JwtSettings jwtSettings)
    {
        _jwtSettings = jwtSettings;
    }

    public string GenerateToken(User user, IEnumerable<string> roles)
    {
        var claims = new List<Claim>
        {
            new Claim(JwtRegisteredClaimNames.Sub, user.Id.ToString()),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            new Claim(JwtRegisteredClaimNames.Iat,
                 DateTimeOffset.UtcNow.ToUnixTimeSeconds().ToString()),
            new Claim(ClaimTypes.GivenName, user.FirstName),
            new Claim(ClaimTypes.Name, user.UserName!),
            new Claim(ClaimTypes.Email, user.Email!)
        };

        foreach (var role in roles)
        {
            claims.Add(new Claim(ClaimTypes.Role, role));
        }

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(
                _jwtSettings.SecretKey!));
        var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: _jwtSettings.Issuer,
            audience: _jwtSettings.Audience,
            claims: claims,
            expires: DateTime.UtcNow.AddHours(_jwtSettings.ExpiresInHours),
            signingCredentials: credentials);

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

Figure 4.18: Backend JWT Generation in ASP.NET Core

```
import React, { createContext, useContext, useState, useEffect } from 'react';
import { jwtDecode } from 'jwt-decode';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
    const [authState, setAuthState] = useState({
       token: null, role: '', userFirstName: '' });
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        const token = localStorage.getItem('jwtToken');
        if (token) {
            try {
```

```
                const decodedToken = jwtDecode(token);
                const role = decodedToken[
        'http://schemas.microsoft.com/ws/2008/06/identity/claims/role'] || '';
                const userfirstName = decodedToken[
        'http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname'] || '';
                setAuthState({ token, role, userfirstName });
            } catch (error) {
                console.error('Invalid token', error);
            }
        }
        setLoading(false);
    }, []);

    const hasRole = (roleToCheck) => authState.role === roleToCheck;

    return (
        <AuthContext.Provider value={{ authState, hasRole, loading }}>
            {children}
        </AuthContext.Provider>
    );
};
export const useAuth = () => useContext(AuthContext);
```

Listing 4.19: Frontend JWT Decoding and Role-Based Access Control

```
[HttpPost]
[Authorize(Roles = RoleConstants.StartupFounder)]
public async Task<IActionResult> CreateTeam(TeamFormDto teamFormDto)
{
        // Action logic ...
}
```

Listing 4.20: Role-Based Authorization for Team Creation

5 Discussion

This chapter analyses the key findings, evaluating the extent to which the objectives were achieved. It examines the impact of software engineering decisions, highlights their implications, and compares the results with relevant industry tools and studies. Broader implications for team organization in technology-based startups are also explored.

5.1 Review of Objectives

The project set out four key objectives to provide technology-based startups with a team organization solution by developing a tailored platform. The platform integrated features such as job posting, portfolio management, a matching algorithm, and team management functionalities. The usability testing results indicated that objectives were partially achieved, with outcomes aligning with user expectations despite certain shortcomings. The following review assesses the extent to which each objective was accomplished based on the outcomes and usability testing results.

Objective 1: To enable startup founders to organize startup teams by hiring skilled individuals.

Outcome: This objective was partially achieved. Key features implemented to support this goal included the ability for startup founders to create and manage job advertisements, review job applications and accept or reject candidates, form teams by adding skilled individuals as members based on accepted job applications, and for skilled individuals to apply for job postings with their profiles and supporting materials such as CVs and cover letters. Usability testing, particularly Task 1: Job Posting and Application Process, evaluated these functionalities. This involved creating job advertisements, applying for jobs, reviewing applications, and forming teams. While participants were generally satisfied, some found the process confusing initially.

Reasoning:

Usability testing revealed several areas for improvement in terminology and navigation. Most users struggled to differentiate between the “Job Listings” and “Manage Jobs” sections on the startup founder panel. After overcoming initial confusion, users could create job postings, though the organization of these sections lacked intuitiveness. In the job advertisement form, Participants 1 and 2 expected dropdowns or multi-select fields for fields such as “Industry”, “Required Skills”, and “Key Technologies”. Additionally, the UI failed to clearly distinguish between links and highlighted text in the job application form, causing Participant 2 to mistake the highlighted CV and Cover Letter fields for clickable links. Another usability issue identified was the need to enter startup details for each job posting. Participants 1 and 4 found this redundant, especially when managing a single startup. Although this feature was intended to accommodate founders who might post jobs for multiple startups, it was

perceived as unnecessary duplication in scenarios where only one startup was being managed. During job applications, Participants 1 and 2 encountered issues with unsupported CV file formats, suggesting a need for clearer instructions on acceptable formats to prevent user frustration.

The team formation process presented further challenges. Users often deviated from the intended workflow, which required accepting an application, awaiting the applicant's acceptance of the offer, and defining roles before adding the individual to a team. This sequential process was not enforced on the platform, leading to confusion. Additionally, the platform did not require roles to be created before job postings, and roles within the team were managed separately from job advertisements. This separation meant that applicants were applying to roles that were not predefined or linked to specific job postings, causing uncertainty about the responsibilities associated with each position. Participants suggested that the platform could benefit from linking role creation with job postings to enforce the correct procedure, ensuring that applicants know exactly what roles they are applying for. Clear guidance and feedback were particularly lacking when users encountered issues during the team member addition process. Participants attempted to add team members without first accepting applications or creating roles, indicating that the need to create roles before adding members was not immediately apparent. Moreover, the ability to assign multiple roles to a team member, although available on the platform, was not intuitive, as Participant 4 reported confusion when attempting to assign multiple roles to a single team member. This highlighted the need for clearer instructions or a more intuitive interface to support multi-role assignments.

Despite these challenges, participants appreciated the integration of job posting and team formation within the platform. The platform's design, based on the specific needs of startups, allowed for team organization once users understood the process.

Objective 2: To enable skilled individuals to showcase their portfolios and join startup teams.

Outcome: This objective was largely achieved. Key features implemented to support this goal included the ability for skilled individuals to create and manage portfolios, apply for jobs with portfolio integration, and enable startup founders to view portfolios during the job application review process. Usability testing, particularly Task 2: Portfolio Creation and Application Review, evaluated these functionalities. This task involved creating and managing portfolios and allowing startup founders to review portfolios during the application process. Overall, participants were generally satisfied, although some found aspects of portfolio creation and management confusing initially.

Reasoning: The portfolio management feature was designed based on the identified gaps during the requirement analysis phase. Usability feedback indicated that most participants appreciated the

interface, finding it straightforward to add and display portfolio items. Participants reported that reviewing applicant portfolios during the application review process was valuable, as it allowed founders to examine the candidate's experience more thoroughly. However, certain usability issues were identified. Participant 2 initially mistook the portfolio section for the profile section but eventually managed to navigate to the correct area. Participant 1 expressed uncertainty about when to add portfolio items after submitting an application and proposed incorporating prompts or a checkbox within the job application form to ensure that portfolios are made mandatory if required. Although the platform was designed to allow skilled individuals to add their portfolios at any point in time, including an option to make portfolios mandatory during the job application process could further streamline and simplify the workflow. Furthermore, Participant 3 suggested integrating an option to attach cover letters alongside portfolios to improve the application process. Participant 4 highlighted the value of encouraging candidates to create multiple, varied portfolio items, even if not directly linked to specific job postings, to better showcase their range of skills and experiences.

Several participants expected the "Users" section to show only startup team members rather than all platform users, especially from the skilled individual panel. Participants suggested categorizing users into distinct groups such as company/team members and all users/applicants to make it easier for individuals to distinguish between their team and the broader platform user base. Although team members are already displayed within the team view for each specific team, adding a filtering option to the Users section of the platform would enhance clarity and further improve the user experience by aligning the interface with participants' expectations. Furthermore, while most participants successfully viewed portfolios during the application review process, Participant 5 had difficulty locating the portfolio within the job application interface. This indicates a need to improve the visibility and accessibility of portfolio links within job applications. Participants highlighted that reviewing applicant portfolios provided valuable insights into candidates' experiences and histories, thereby aiding in more informed hiring decisions. Enhancing the interface to make portfolio access more intuitive would further support this functionality.

Despite these challenges, participants were generally satisfied with the platform's integrated approach, which facilitated portfolio showcasing and team recruitment in a single platform, enhancing the overall team organization experience.

Objective 3: To match suitable skilled individuals to startup job postings and assist founders in finding the right talent.

Outcome: This objective was partially achieved. The platform's matching algorithm, based on TF-IDF and cosine similarity, generally provided accurate and relevant pairings that helped startup founders

identify suitable candidates and enabled skilled individuals to find pertinent job postings. The algorithm was tested during the development phase using simulated data to refine its functionality and further validated through usability testing with real users. Usability testing, particularly Task 3: Interacting with the Matching Algorithm, evaluated these functionalities. This task involved using the matching feature to find relevant job postings or candidates based on CVs and portfolios. Participants were generally satisfied with the accuracy of job matches, though some desired more control and transparency over the matching criteria and experienced confusion with certain interface elements.

Reasoning: Participants understood the matching process and appreciated the relevance of suggested matches, which enhanced team organization and the hiring process. Nevertheless, some usability challenges and improvement areas emerged. Many, including Participant 1 and Participant 3, desired greater control over the matching criteria, seeking options to prioritize CVs, portfolios, or both, and to manually specify fields such as skills, years of experience, and key technologies. For instance, Participant 2 found that the algorithm penalized candidates with strong portfolios but less relevant CVs, expecting matches if either aligned with the job posting. Similarly, Participant 3 noticed a drop in match scores after adding irrelevant portfolio items, highlighting the need for more nuanced control over portfolio prioritization. Participants suggested several improvements, such as adjusting the matching threshold and manually selecting match criteria. Participant 4 recommended prioritizing matches if either the CV or portfolio aligns with job requirements, while Participant 2 believed that overlapping skills between CVs and portfolios should enhance match scores. Participant 5 preferred the CV as a more reliable matching factor and sought greater transparency in how matches are calculated.

The current automated matching algorithm, which aligns CVs and portfolios with comprehensive job advertisement fields—including job title, job description, required skills, key technologies, industry, experience, education, and startup description—and corresponding CV and portfolio fields such as title, description, skills, technologies, and industry, as the matching parameters were not suitable for all users. This led to inefficiencies and frustrations, as the algorithm did not cater to the diverse needs and expectations of different users. Additionally, some participants noted that multi-word phrases sometimes led to less relevant matches, suggesting refinements to the algorithm's dictionary or the incorporation of machine learning to improve context sensitivity and accuracy. These feedback points highlight the need for customizable preferences to prevent unintended filtering and ensure a more balanced and effective matching outcome.

User interface issues were also evident, particularly with the select box in the job matching feature within the startup founder panel. Participants, including Participants 1 and 3, specifically suggested that matches should display by default without requiring the additional step of selecting a job advertisement,

as they found this process inefficient. However, since candidate matches must align with a specific job posting, selecting a job advertisement is an integral part of the algorithm's behaviour. Removing this step could result in an overwhelming display of matches across all advertisements, making it difficult for users to distinguish between them. To address these concerns, the select box design should be improved with bolder styling or clearer labelling to make it visually distinct from a text box, reducing user confusion. Moreover, while users were satisfied with the overall accuracy of the matching algorithm, they desired more transparency regarding how matches were calculated. Participants requested clearer indications of the factors influencing matching scores and a detailed explanation of the algorithm's decision-making process. This desire for transparency reflects a need for users to trust and understand the algorithm's logic, thereby increasing their confidence in the matching results.

Overall, participants rated the matching feature positively, appreciating its utility in facilitating candidate-job pairings. However, feedback indicated that additional customization options, enhanced transparency, and user interface adjustments are necessary to further improve user experience and better align the platform with the needs of both startup founders and skilled individuals.

Objective 4: To facilitate startup team management through goal setting, milestone tracking, role definitions, and task management.

Outcome: This objective was partially achieved. Key features implemented to support this goal included role assignment, goal setting, and milestone tracking, all of which were evaluated through Task 4: Roles, Goals, and Milestones Setup during usability testing. Participants found these features generally intuitive and effective for managing teams. However, the absence of task management, which would enable breaking milestones into assignable tasks with progress tracking, and identified usability issues limited the platform's ability to fully meet this objective.

Reasoning: The platform provided a structured approach to team management, with clear definitions for roles, goals, and milestones. Role assignments and the hierarchy of goals and milestones were particularly appreciated. For instance, while most participants found the relationship between goals and milestones logical and easy to understand, Participant 4 proposed a more detailed hierarchy. They suggested organizing milestones to contain goals, which could be further broken down into objectives and tasks, each with progress indicators. This feedback highlights varying perceptions of the hierarchy between "goals" and "milestones". The platform should offer clearer guidance to ensure consistent understanding and support granular tracking for complex team workflows. Furthermore, the absence of task management was a recurring concern. Participant 1 emphasized the need to break goals into assignable tasks for better tracking and accountability, while Participant 2 proposed assigning specific

team members to goals and milestones to enhance ownership. Participant 4 suggested integrating status tracking at all hierarchy levels to measure progress effectively.

User interface and navigation were generally smooth, with most participants finding the features easy to use. However, some enhancements were suggested. Several participants highlighted the need for enhanced clarity and flexibility in role assignments. While the platform supports assigning multiple roles to a single individual, Participants 3 and 4 found it unclear how to do so. In contrast, Participant 1 expected that each team member would be limited to a single role, reflecting differing user expectations and underscoring the need for better guidance on the role assignment feature. Moreover, Participant 5 recommended adding a direct "Teams" button on the dashboard for quicker navigation. Participant 1 also recommended organizing the team view so that the roles section appears before the members section, improving clarity and design.

Despite the lack of a task management feature, the platform's modular monolith architecture supports the potential integration of this feature in future iterations without significant changes, ensuring scalability and reliability. Incorporating task assignments or accountability features and addressing usability shortcomings would respond to users' feedback and enable the platform to fully realize this objective, offering a comprehensive startup team organization solution.

5.2 Comparison with Existing Solutions

The platform developed during this project potentially addresses the unique team organization needs of technology-based startups, as identified in the literature and confirmed by project results. Literature highlights existing tools such as LinkedIn (2024), Indeed (2024), Upwork (2024), Microsoft Teams (Microsoft, 2024), Asana (2024), Trello (2024), and Jira (Atlassian, 2024) are well-suited for general recruitment, team management, and project management but fall short of supporting the specialized requirements of startup team organization. None of these platforms are fully equipped to handle the nuances of the team organization in technology-based startups, lacking integrated features for talent acquisition, accessing a talent pool of skilled individuals, portfolio showcase, team formation, ongoing role reassignments and changes to the team composition, complementary skill sets, and a shared vision within team members all of which are critical for organizing technology-based startup teams (Seppänen et al., 2017; Patzelt et al., 2021). This project bridges these gaps by integrating recruitment and team management features within a single environment.

Requirement-focused platforms such as LinkedIn, Upwork, and Freelancer (2024) provide broad professional networking and job-posting features but lack targeted support for startup-specific needs. These platforms do not support detailed job postings that convey startup-specific information, such as the stage, mission, or technology stack, nor do they offer team formation and portfolio integration

features. While networking feature of LinkedIn and freelancers pool on Upwork and Freelancer can be advantageous, they primarily rely on keyword-based matching, which does not account for nuanced skills and terminologies unique to startup environments. In contrast, the developed platform employs a custom matching algorithm that incorporates a dictionary of startup-specific terms and N-grams, enhancing the relevance and accuracy of job-candidate matches. This algorithm assists both startup founders and skilled individuals by ensuring that matches consider comprehensive candidate profiles, including both CVs and portfolio information—features that are often missing in other platforms like LinkedIn. Usability testing confirmed that users appreciated the algorithm's ability to target relevant skills more effectively, although some participants expressed a desire for more control over matching criteria, such as customizable algorithm parameters.

Furthermore, project management tools like Asana, Trello, and Jira excel in managing tasks and tracking milestones but lack integration with recruitment and team management functionalities essential for startups. These platforms do not facilitate dynamic role assignments, skill matching, or integrated portfolio features alongside recruitment processes, which are critical as startups evolve. For example, Jira's focus on agile issue tracking is beneficial for development projects but lacks direct support for team reformation, a recurring requirement in fast-paced startups. The developed platform addresses these limitations by integrating team management features such as goal and milestone tracking and role assignment and recruitment features like skill matching, adaptive team restructuring, and portfolio showcases, all necessary for technology-based startups. Usability testing confirmed that users found this integration helpful, potentially reducing the need for multiple, separate tools. Participants particularly appreciated the ability to direct team member addition from job applicants; however, they suggested the inclusion of more guidance and enforcement of the correct flow for this process within the platform.

The results also respond to gaps highlighted in studies by Cantamessa et al. (2018) and Noui and Dehane (2023), which underline the necessity of adaptive solutions for team dynamics and role management in startups. The developed platform's design aligns with key theories of team dynamics as well, including Tuckman's Stages of Group Development, Hackman's Model of Team Effectiveness, and Belbin's Team Roles Theory (Tuckman, 1965; Hackman, 2002; Belbin, 2011). These frameworks emphasize the importance of clear roles, task alignment with team member strengths, and a shared vision among team members, which are challenging to maintain in a startup environment using existing fragmented tools. The platform's features, such as role creation, with the ability to designate multiple roles to a member, goal/milestone tracking, and the matching algorithm, excluding task management, directly support these theoretical needs, enhancing team cohesion and adaptability, as highlighted in usability testing results.

The platform consolidates recruitment and team management features into a single tool, eliminating the fragmentation seen in traditional platforms and addressing the gap identified in technology-based startup team organization, although certain shortcomings remain. The platform offers an integrated approach to hiring skilled individuals directly into startup teams and assigning them roles, which are essential for startups dealing with changing team composition. The goal and milestone tracking features further assist in providing a solution for managing the startup team and bringing clarity to team activities. Additionally, the matching algorithm adds a level of sophistication to the recruitment process that existing solutions often lack by considering the specific keywords from the technology-based startup environment. Unlike LinkedIn and Upwork, where users often need to manually link profiles or portfolios, the platform simplifies these processes by integrating them into the application and recruitment workflows, reducing manual steps. Usability testing showed that participants were able to complete tasks like job posting, role assignment, and team management, though faced some challenges as they required time to adjust to the integrated interface and as the platform failed to provide enough guidelines to them on certain processes such as team member addition process.

Usability testing also demonstrated that the platform could handle typical startup tasks, such as posting jobs and matching candidates, with stable performance and minimal delays. Although formal stress testing was not conducted, usability testing under typical conditions indicated that the platform's responsiveness was sufficient to support key startup tasks, such as job posting and candidate matching, without noticeable delays. One limitation observed was the inability to configure HTTPS due to the use of a public IP, resulting in HTTP-only connections post-deployment. While this differs from established platforms like LinkedIn and Upwork, which typically offer HTTPS on registered domains, the platform mitigates security risks with features such as JSON Web Tokens (JWT) for authentication and role-based access controls to protect user data. Future deployment enhancements, such as acquiring a registered domain and enabling HTTPS, will further align the platform's security measures with industry standards.

Overall, the developed platform integrates several specialized features to provide an adaptive approach to team organization in technology-based startups. By aligning with existing literature and addressing specific market needs, the platform positions itself as a specialized integrated solution over the current generic tools.

5.3 Impact of System Design Choices

The wireframes provided a balance between detail and flexibility, allowing for adjustments during frontend development. Considering more detail in the wireframes could have improved the clarity of platform pages, particularly the matching algorithm interface, which was less intuitive than others. This

lack of detail likely contributed to issues with navigation between job posting, team creation pages, and adding new team members, where users needed clearer guidance, even though the platform's visual design was generally well-received. The simplified Cockburn Use Case Template ensured that the essential elements of each use case were captured without adding unnecessary complexity. The resulting use cases provided a clear roadmap for the platform's functionality, ensuring that all user interactions were systematically addressed during development. The Requirements Traceability Matrix (RTM) was helpful in tracing functional requirements, aiding backend and frontend development, while non-functional requirements were addressed through architectural and coding best practices. Prioritization of requirements proved critical, especially when the Task Management feature was deprioritized due to its high cost and low value to the overall platform's objectives.

The modular monolith architecture proved effective, supporting the hybrid software development life cycle model and incremental feature development. It minimized module interruptions, allowed task management feature deprioritization, and reduced breaking changes while maintaining flexibility and control. Although integrating cross-module code like the Matching Algorithm was moderately challenging, this approach improved code organization and maintainability. Scalability was a key advantage, enabling future transitions to distributed architectures such as microservices with minimal refactoring by replacing inter-module runtime calls with HTTP requests. The central API acted as a façade, enabling communication and integration across modules, while maintaining the independence of each module. Moreover, this architectural choice separated the database schema for each module which may reduce the efficiency of complex queries requiring traditional joins across multiple schemas. To mitigate this, the design embraced clarity in queries, accepting the trade-off of requiring multiple queries in exchange for improved maintainability and scalability. The architecture provides future growth alongside the requirements it serves, accommodating additional features and modules. The breakdown of code inside each module using layered architecture helped organize the code further on the module level.

ASP.NET Core provided a smooth experience for API development, while React.JS required more effort for UI implementation, particularly with server interactions, routing, roles-based access, and state management, which proved more challenging than backend tasks. Nonetheless, the client-server model adhered to the Separation of Concerns (SoC) principle from SOLID, enabling independent development and scaling of components without added complexity (Martin, 2003). Backend testing with the Arrange-Act-Assert (AAA) pattern validated module functionalities in isolation, while frontend component testing ensured UI elements functioned correctly across different scenarios. Regarding security, implementing JWT and Google Authentication workflows posed challenges, particularly with Google Authentication failing during deployment due to restrictions on public IP usage, which required a

private domain address. Attempts to set up Let's Encrypt for HTTPS on Kubernetes with Cert Manager also failed for the same reason (Let's Encrypt, 2024; Cert-manager, 2024). This highlights the importance of pre-emptively planning domain registration for platforms reliant on third-party authentication. Azure Kubernetes Service (AKS) ensured a smooth deployment, offering scalability, high availability, and fault tolerance.

One of the key features of the platform is the matching algorithm, which uses a combination of Term Frequency-Inverse Document Frequency (TF-IDF) and Cosine Similarity to match job postings with skilled individuals. This hybrid approach was chosen because it offers a balance between computational efficiency and relevance in text-based matching. During usability testing, the algorithm generated accurate matches based on participants' portfolios and job descriptions, validating the decision to use this hybrid technique. However, while the algorithm's performance was generally acceptable, the testing revealed areas for improvement. In some cases, the algorithm did not provide favorable results to users because it considered all job posting attributes, such as required skills, key technologies, and industry (as previously detailed), rather than focusing on those users preferred to prioritize, such as required skills or key technologies, or placing greater emphasis on CV details over portfolio information. This highlighted that over-automation was not favorable to users, as they preferred more manual control in certain aspects of the matching process.

Overall, the system's design emphasized flexibility, scalability, and maintainability, with a modular architecture enabling future adaptability. The matching algorithm performed well but revealed user preference for manual control in certain aspects. Moreover, security challenges highlighted the importance of preemptive planning, while robust testing ensured functionality and stability.

5.4 Recommendations for Improving Platform Usability

Based on the usability testing results, several key improvements are recommended to enhance the platform's usability. Firstly, navigation should be clarified by renaming sections such as "Job Listings" and "Manage Jobs" with more descriptive titles and grouping related functions together. Adding prominent buttons, like "Create Job Posting" and "Teams," on the dashboard will streamline access to frequently used features. Form design should be enhanced by incorporating dropdowns, multi-select fields, and real-time file format validation to reduce user errors and improve data entry efficiency. Additionally, implementing a persistent company profile that auto-populates startup information can eliminate redundant data entry. To prevent user confusion, acceptable file formats should be clearly listed near upload buttons, and clickable elements should be visually distinct from non-interactive text.

Furthermore, the platform should implement guided workflows to enforce the correct sequence of actions, such as creating roles before adding team members, accompanied by clear prompts and error

messages. Enhancing the matching algorithm's usability by allowing users to customize matching criteria and providing transparency in how matching scores are calculated will align matches more closely with user preferences and build trust in the system. Differentiating user categories through dedicated sections for team members and general users will improve information relevance and simplify interactions. Additionally, integrating task management features and clearly defining the hierarchy between goals, milestones, and tasks will support comprehensive team management and accountability. Providing onboarding tutorials, real-time error feedback, and maintaining a consistent design language across the platform will further improve the user experience and ensure the platform effectively meets the needs of startup founders and skilled individuals.

5.5 Identified Limitations

The final platform, while aligning with the core objectives, has some shortcomings that present opportunities for future improvement:

- **Task Management Deprivation:** The absence of task management limited the platform's team management capabilities, as users requested. This feature could be added in future iterations to enhance the platform's comprehensiveness.
- **Security and Authentication:** Due to deployment constraints, HTTPS and Google Authentication could not be utilized, impacting the platform's robustness. Addressing these issues in future iterations by using a private domain on Kubernetes would improve security.
- **Performance Testing:** Performance metrics were only captured under idle conditions and through human participants survey, without heavy data loads. Future work should include stress testing to assess scalability under more demanding scenarios, particularly for the matching algorithm.
- **User Diversity in Usability Testing:** The usability testing was conducted with a small group of five participants, all of whom were male. This homogeneity in both size and gender representation may limit the generalizability of findings. Expanding future tests to include a larger and more diverse sample would yield broader insights.
- **Matching Algorithm Limitations:** The algorithm matches job advertisements with skilled individuals by analysing the attributes of specific job postings (e.g., required skills, key technologies, industry) against candidate profiles, including their CVs and portfolios. It only processes candidates who have applied for a given job, ensuring relevance to the role. While the algorithm performed well, its reliance on a predefined keyword dictionary for multi-word matching could be enhanced using NLP techniques like word embeddings to better capture the contextual meaning of attributes in job descriptions and candidate portfolios (Manning et al.,

2008). Customization options, such as allowing users to prioritize specific attributes (e.g., skills over experience or portfolio over CV), would provide more tailored results, improving the algorithm's flexibility and alignment with user preferences. Lastly, evaluating the algorithm using alternative ranking metrics, such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) (Manning et al., 2008), would provide a more holistic view of its performance under varied data loads as the platform's user base grows.

5.6 Generalizability of Results

The platform developed is specifically tailored to address the unique challenges faced by technology-based startups by combining recruitment and team organization within a single platform. Therefore, the results and findings are most applicable to the startup ecosystem, requiring flexible and integrated solutions for team management. However, the platform's core functionalities, such as the matching algorithm and modular monolith architecture, suggest broader potential due to their scalability and adaptability. The matching algorithm, designed to analyse job attributes against candidate profiles, can be adapted for other domains requiring resource matching, such as healthcare recruitment or academic admissions, by configuring its customizable attributes and ranking techniques to suit specific needs. Similarly, the modular monolith architecture's scalability and flexibility make it suitable for integrating multiple functionalities in systems like customer relationship management (CRM) or enterprise resource planning (ERP), enabling customization and extension beyond the startup context.

6 Evaluation, Reflections, and Conclusions

This chapter evaluates the overall effectiveness of the platform developed for the team organization in technology-based startups. It reflects on the development process, highlights challenges encountered and their mitigation strategies, and provides conclusions drawn from the findings. Additionally, it outlines the future work that could enhance the platform further.

6.1 Project Outcome Evaluation and Insights

The platform developed during this project addresses the key challenges associated with team organization in technology-based startups. The overall design, functionality, and usability results demonstrate the ability of the platform to address the needs of startup founders and skilled individuals, potentially helping the startup community. The assessment of requirements fulfilment, features effectiveness, and testing outcomes are presented in the below sections.

6.1.1 Evaluation of Requirements Fulfilment

The project's success can be evaluated by assessing how well the platform fulfilled the functional and non-functional requirements defined during the design phase.

- **Functional Requirements:** Core features like job posting, portfolio management, team formation, and the matching algorithm were implemented, with the task management feature being dropped. Usability tests validated that users were able to utilize these features without significant difficulty, confirming the platform's functional capabilities.
- **Non-Functional Requirements:** The platform adhered to non-functional requirements, including scalability, maintainability, and security. The modular monolith architecture ensured stability, as well as maintainability and flexibility for future development. Security measures, such as JWT-based authentication, provided secure access; however, HTTPS and Google Authentication were limited due to deployment restrictions involving public IPs.

6.1.2 Functional Testing

Core features were unit tested for the backend, and components tested for the frontend, ensuring bug-free functionality. No bugs were found during usability tests, as per participants' survey responses. Moreover, the matching algorithm was tested for accuracy using generated ground data and ground truth with different threshold values. While the algorithm performed well overall, user feedback indicated a need for more customization in the matching criteria. Furthermore, features such as job management, portfolio management, and team management were tested and confirmed to work as expected, with some usability issues found.

6.1.3 Usability Testing

The incorporation of real user feedback through usability testing proved essential. While the literature review helped identify the theoretical gaps and functional requirements for the platform, it was through usability testing that the practical implementation issues became clear. Delaying stakeholder engagement until the usability phase allowed the project to focus on core functionalities during development, while still incorporating user-driven insights at a crucial point in the project's lifecycle. This approach, consistent with the incremental software development model, ensured that user feedback was gathered on a working MVP of the software, where it could most effectively evaluate the project. Usability testing validated the platform's core functionalities and highlighted the areas for improvement. The recruitment and team organization features integration within a single platform was particularly appreciated for reducing tool fragmentation.

However, key areas for improvement were identified, including enhanced guidance for navigating job postings and team member addition processes. It is a reminder that even when the functionality is robust, user experience can be hindered by insufficient instruction or overly complex interfaces. A notable issue was the over-automation of the matching algorithm, which lacked user preferences, making the interface less intuitive. Users sought matches based on skill alignment, but the algorithm's equal weighting of all attributes, such as required skills and key technologies, occasionally led to less favourable results. This highlighted the need for a balance between automation and user control through customizable criteria like prioritizing CV details or portfolio elements. Participants also highlighted the value of task management features, including task breakdowns and accountability tracking, as a necessary addition to the team management module. These insights reinforced the importance of user-centric refinements to address practical challenges and ensure the platform effectively meets the needs of its target audience.

6.1.4 Performance Testing

While extensive performance testing was not conducted, the platform demonstrated stable and responsive operation during typical user tasks and idle mode. The system handled typical tasks, such as job posting and candidate matching, without noticeable delays or performance issues. Users noted acceptable performance speeds during the usability tests, although high-load performance remains to be evaluated.

6.1.5 Compliance Testing

The platform met essential security and privacy standards, with data encryption and role-based access control. Automated tests confirmed these security measures, but limitations in HTTS and Google

Authentication impacted full compliance. Addressing these restrictions in future iterations will further enhance security.

6.1.6 Project Planning and Execution

The project planning and execution followed a hybrid model, combining the elements of structured, linear planning with iterative refinements. The project phases included preparation, design, development, and evaluation, each progressively building and validating the platform. Key steps like requirement gathering, creating a modular monolith architecture, implementing the matching algorithm, and developing user management, job management, and team management features were obtained progressively. Regular feedback loop from automated testing guided software adjustments with usability test in the final stage evaluating the project with the users' feedback. This phased and iterative approach enabled the completion of the MVP, within the project timeline, maintaining the focus on essential functionalities while accommodating user needs and platform scalability.

6.1.7 Choice of Objectives

The objectives for this project were selected to address specific challenges in organizing technology-based startup teams. These objectives were defined with a focus on practical outcomes and academic requirements, ensuring they were achievable within the project's scope and timeline. Each objective aimed to contribute to the theoretical and practical aspects of startup team organization. They were formulated to align with the program's goals, allowing for a structured investigation and development process that connects academic knowledge with real-world contexts.

6.1.8 Methods Applied

The methods applied in this project followed a structured approach, integrating theoretical and practical techniques to address the project objectives. Initially, a literature review was conducted to establish the background and identify existing approaches to team organization in technology startups. This was followed by a requirements analysis to outline essential features and functionalities based on identified needs. The project then employed a modular monolith architecture to ensure scalability and maintainability. Development involved a feedback loop from automated testing, enabling continuous validation of functionality and system reliability. Usability testing was conducted at the end of the development phase, providing insights into user experience and ensuring alignment with the project's practical goals. This structured approach allowed the project to systematically address requirements while maintaining flexibility in development.

6.2 Key Achievements and Successes

Several key achievements stand out from the project:

- The platform addressed the unique challenges of technology-based startups, such as dynamic team reformation and integrated team management, verified by users.
- The matching algorithm particularly proved effective in connecting startup founders with skilled individuals and vice versa.
- The modular monolith architecture and decoupled frontend and backend architectures combined with Azure cloud infrastructure allowed for modularity, scalability, and maintainability, ensuring the platform could grow and evolve with future updates.
- Usability testing demonstrated promising results, indicating users found the platform intuitive and aligned with their needs overall, integrating recruitment and team management, although certain shortcomings existed.

6.3 Challenges Faced and Mitigation Strategies

Some challenges emerged during the development of the platform, necessitating strategic decisions to keep the project on track. The following reflections reveal how each challenge shaped the current implementation:

- **Time Constraints and Feature Prioritization:** The limited time frame led to the prioritization of features, which ultimately affected the project's scope. Although task management was initially planned as a component of team organization, it was deprioritized to ensure the delivery of fundamental features like job posting, portfolio management, and the matching algorithm. This prioritization allowed the project to follow its core objectives, although the absence of task management limited the overall platform's functionality. This experience emphasized the need for effective time management and realistic goal-setting in software project planning.
- **Integration Complexity:** Integrating the matching algorithm with job management and portfolio management modules presented challenges, particularly in ensuring seamless data flow and communication between modules and handling inter-dependencies. These issues were addressed through modular monolith design, early planning for implementation, and testing these functionalities, though caused minor delays during development. The experience highlighted the importance of a modular software design to handle complex integrations.
- **Deployment Constraints:** Although the platform incorporated HTTPS and Google Authentication support, these features became inaccessible upon deployment. HTTPS could not be obtained due to Kubernetes' limitations with encrypting connections using public IPs, and the same public IP restriction prevented Google Authentication from functioning. This reduced the platform's security and user authentication options. Future iterations would aim to address these constraints by using a private domain setup on the Kubernetes instance.

6.4 Lessons Learned

This project provided valuable insights into both technical and project management aspects. A key takeaway was the importance of prioritizing core features and focusing on usability testing and refinement over extensive feature expansion to ensure timely delivery while meeting the platform's objectives. Were the project to begin anew, earlier integration of stakeholder feedback might be considered to refine requirements from the outset, potentially reducing the need for adjustments later in the project. Usability testing, conducted at the final stage, was invaluable in identifying areas for improvement and aligning the platform with target users' expectations. However, future projects would benefit from an iterative approach to usability testing, with usability testing integrated throughout development. This would allow user-driven refinements to be incorporated progressively, enhancing the platform's usability and minimizing end-stage modifications. Additionally, balancing performance and accuracy in the matching algorithm was a critical focus. Combining TF-IDF with Cosine Similarity and N-Grams provided a method that balanced computational efficiency and relevance in text-based matching. Although alternative approaches, such as advanced machine learning models, were not compared directly due to project constraints, the selected method supported the platform's matching goals and performed well in user testing.

The choice of a modular monolith architecture was another critical lesson. This architecture proved effective for scalability and maintainability, enabling the platform to grow and adapt without significant refactoring. It supported incremental feature development and ensured stability across modules, providing a strong foundation for future improvements. While this architecture met the project's needs, its initial complexity required additional time for understanding and implementing cross-module integration. A simpler architecture, such as Model-View-Controller (MVC), might reduce initial development complexity and improve focus on core functionality. Unlike the modular monolith, which emphasizes modular separation and scalability, MVC focuses on clear separation of concerns between the user interface, business logic, and data layers, which may streamline development. However, the modular monolith's ability to support future scalability and modularity suggests that the initial complexity was a worthwhile trade-off in this project.

In retrospect, these lessons highlight the significance of maintaining a user-focused approach, selecting architectures aligned with long-term goals, and ensuring that feature design and performance optimization remain balanced with project constraints.

6.5 Future Work and Recommendations

While the platform met most of its core objectives, areas for improvement remain. A key priority is integrating the deprioritized task management feature, which would enhance the platform by enabling

startup teams to assign and track tasks better and define clear responsibilities for goals and milestones. Refining the matching algorithm is also recommended by incorporating user-defined preferences, optimizing the threshold value through extended real-world testing, and exploring machine learning techniques to improve accuracy. In future iterations, advanced methods like natural language processing (NLP) could further enhance matching flexibility and scalability. Future work should also focus on conducting more detailed performance testing, including load and stress tests, to ensure scalability and reliability under heavier usage. Security enhancements are also necessary, particularly addressing public IP restrictions to establish HTTPS and Google Authentication in the deployment environment for more robust user protection. Further platform expansions, such as a notification system for key events and a feedback and rating system, would improve user engagement. Including a feature for sharing reports within startup teams would be a valuable improvement, as well as incorporating a subscription model for potentially commercializing the platform. Ultimately, regular maintenance activities such as bug fixing, performance monitoring, and feature updates will ensure the platform remains functional and relevant over time. Although these maintenance tasks extend beyond the current project scope, they are essential for the long-term sustainability and usability of the platform.

6.6 Personal Reflections

The project presented challenges that were greater than initially anticipated, but these obstacles became valuable learning opportunities. At various points, the complexity of the tasks and the need to meet deadlines pushed the researcher to develop stronger problem-solving and time management skills. While technical and organizational difficulties arose, they ultimately contributed to the researcher's personal and professional growth, fostering resilience and adaptability. The experience proved to be rewarding, equipping the researcher with the ability to manage multifaceted projects effectively and confidently navigate challenges in future endeavours.

6.7 Project Conclusion

Over the course of three months, this project addressed the core challenges of organizing technology-based startup teams, achieving its primary objective. An integrated platform was developed, combining features such as job management, portfolio management, a matching algorithm, and team management. The system design choices, particularly the use of a modular monolith architecture, ensured scalability and maintainability, while usability testing confirmed the platform's potential to meet the needs of technology-based startups. Despite limitations, such as the deprioritization of the task management feature and identified usability issues, like the lack of clarity in the job matching user interface, the platform's foundational elements were successfully implemented and well received by participants, making a contribution to addressing the gap in the market for a specialized startup team organization

solution. Future work will continue to build on this foundation, enhancing the platform's capabilities and expanding its scope.

References

1. Asana. (2024). Asana: Manage your team's work, projects, & tasks online. Available at: https://asana.com (Accessed: 8 December 2024).
2. Atlassian. (2024). Jira | issue & project tracking software. Available at: https://www.atlassian.com/software/jira (Accessed: 8 December 2024).
3. Auth0 (2024). What is a JSON Web Token (JWT)?. Available at: https://auth0.com/learn/json-web-tokens/ (Accessed: 8 December 2024).
4. Barney, J. (1991) 'Firm resources and sustained competitive advantage', Journal of Management, 17(1), pp. 99–120. Available at: https://doi.org/10.1177/014920639101700108.
5. Barnum, C.M. (2011) Usability testing essentials: ready, set-- test. Burlington, MA: Morgan Kaufmann Publishers.
6. Basecamp. (2024). Project management software, online collaboration. Available at: https://basecamp.com/ (Accessed: 8 December 2024).
7. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001) 'Manifesto for Agile Software Development', Agile Alliance. Available at: http://agilemanifesto.org/ (Accessed: 8 December 2024).
8. Belbin, R. M. (2011). Team Roles at Work (2nd ed., Reprinted). Amsterdam: Butterworth-Heinemann.
9. Blank, S., and Dorf, B. (2012) The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company. Pescadero: K&S Ranch Press.
10. Booch et al. (2005) The Unified Modeling Language User Guide. 2nd ed. Upper Saddle River, NJ: Addison-Wesley.
11. Cantamessa et al. (2018) 'Startups' roads to failure', Sustainability, 10(7), p. 2346. Available at: https://doi.org/10.3390/su10072346.
12. Canva (2024). Create beautiful designs with your team. Available at: https://www.canva.com/ (Accessed: 8 December 2024).
13. CB Insights. (2021). Report: "The Top 12 Reasons Startups Fail". URL: https://www.cbinsights.com/research/report/startup-failure-reasons-top. (Accessed: 8 December 2024).
14. Cert-manager (2024) cert-manager. Available at: https://cert-manager.io/ (Accessed: 8 December 2024).
15. Cockburn, A. (2001) Writing effective use cases. Boston: Addison-Wesley (The Crystal series for software development).

16. Cosine similarity (2024) Wikipedia. Available at: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1238947356 (Accessed: 8 December 2024).
17. Diakanastasi et al. (2018) 'Entrepreneurial team dynamics and new venture creation process: an exploratory study within a start-up incubator', SAGE Open, 8(2), p. 215824401878144. Available at: https://doi.org/10.1177/2158244018781446.
18. Docker. (2022). Accelerated container application development. Available at: https://www.docker.com/ (Accessed: 8 December 2024).
19. Freelancer. (2024). Hire freelancers & find freelance jobs online. Available at: https://www.freelancer.com/ (Accessed: 8 December 2024).
20. GeeksforGeeks. (2024) What is a modular monolith? Available at: https://www.geeksforgeeks.org/what-is-a-modular-monolith/ (Accessed: 8 December 2024).
21. Genome, S. (2024) Startup genome, Startup Genome. Available at: https://startupgenome.com/article/the-state-of-the-global-startup-economy (Accessed: 8 December 2024).
22. Ghannoum, H. (2024) 'Modular monolith architecture in .Net', Code Maze, 29 May. Available at: https://code-maze.com/dotnet-modular-monolith/ (Accessed: 8 December 2024).
23. GitHub (2024) 'GitHub: Where the world builds software', GitHub. Available at: https://github.com (Accessed: 8 December 2024).
24. Google. (2024). Google Identity Platform Documentation. Available at: https://developers.google.com/identity (Accessed: 8 December 2024).
25. Hackman, J. R. (2002). Leading Teams: Setting the Stage for Great Performances. Boston, MA: Harvard Business School Press.
26. IBM Rational Software Architect (2023). Available at: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-deployment (Accessed: 8 December 2024).
27. Indeed. (2024). Indeed. Available at: https://www.indeed.com/ (Accessed: 8 December 2024).
28. Jest (2024) 'Jest: Delightful JavaScript Testing', Jest. Available at: https://jestjs.io (Accessed: 8 December 2024).
29. Jovanović, M. (2023) Modular monolith data isolation. Available at: https://www.milanjovanovic.tech/blog/modular-monolith-data-isolation (Accessed: 8 December 2024).
30. Kalra et al. (2022) 'Generation of domain-specific vocabulary set and classification of documents: weight-inclusion approach', International Journal of Information Technology, 14(1), pp. 275–285. Available at: https://doi.org/10.1007/s41870-021-00830-8.

31. Karabiber, F. (2024) Cosine similarity. Available at: https://www.learndatasci.com/glossary/cosine-similarity/ (Accessed: 8 December 2024).
32. Katzenbach, J. R., and Smith, D. K. (2015) The Wisdom of Teams: Creating the High-Performance Organization. 3rd ed. Boston: Harvard Business Review Press.
33. Kemell, K.-K. et al. (2020) 'Business model canvas should pay more attention to the software startup team', in 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Portoroz, Slovenia: IEEE, pp. 342–345. Available at: https://doi.org/10.1109/SEAA51224.2020.00063.
34. Kingsley, M.S. (2024) 'Comparing aws, azure, and gcp', in Kingsley, M. S., Cloud Technologies and Services. Cham: Springer International Publishing, pp. 381–393. Available at: https://doi.org/10.1007/978-3-031-33669-0_12.
35. Kirkman, B.L. et al. (2002) 'Five challenges to virtual team success: Lessons from Sabre, Inc.', Academy of Management Perspectives, 16(3), pp. 67–79. Available at: https://doi.org/10.5465/ame.2002.8540322.
36. Klada, T.-V. (2018) 'Unraveling entrepreneurial team formation: a qualitative study among funded ventures', SAGE Open, 8(2), p. 215824401877670. Available at: https://doi.org/10.1177/2158244018776700.
37. Kubernetes. (2024). Production-grade container orchestration. Available at: https://kubernetes.io/ (Accessed: 8 December 2024).
38. Labelf.ai (2024) 'What is accuracy, precision, recall and F1 score?', Labelf AI Blog. Available at: https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score (Accessed: 8 December 2024).
39. Let's Encrypt (2024). Available at: https://letsencrypt.org/ (Accessed: 8 December 2024).
40. LinkedIn. (2024). LinkedIn. Available at: https://www.linkedin.com (Accessed: 8 December 2024).
41. Manning et al. (2008) Introduction to information retrieval. New York: Cambridge University Press.
42. Martin, R.C. and Martin, R.C. (2003) Agile software development: principles, patterns, and practices. Upper Saddle River, NJ: Prentice Hall/Pearson Education (Alan Apt series).
43. MDN Web Docs (2024) 'Using the Fetch API', Mozilla Developer Network. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (Accessed: 8 December 2024).
44. Melegati, J. and Kon, F. (2020) 'Early-stage software startups: main challenges and possible answers', in Nguyen-Duc, A. et al. (eds) Fundamentals of Software Startups. Cham: Springer International Publishing, pp. 129–143. Available at: https://doi.org/10.1007/978-3-030-35983-6_8.
45. Microsoft (2024) 'Azure Container Registry', Microsoft Azure. Available at: https://azure.microsoft.com/en-us/products/container-registry (Accessed: 8 December 2024).

46. Microsoft (2024) 'Entity Framework Core documentation', Microsoft Docs. Available at: https://learn.microsoft.com/en-us/ef/core/ (Accessed: 8 December 2024).
47. Microsoft (2024) 'Managed Kubernetes Service (AKS)', Microsoft Azure. Available at: https://azure.microsoft.com/en-us/products/kubernetes-service (Accessed: 8 December 2024).
48. Microsoft (2024) 'Visual Studio Code documentation', Microsoft Docs. Available at: https://code.visualstudio.com (Accessed: 8 December 2024).
49. Microsoft (2024) 'Visual Studio documentation', Microsoft Docs. Available at: https://visualstudio.microsoft.com (Accessed: 8 December 2024).
50. Microsoft (2024). ASP.NET Core Identity. Available at: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity (Accessed: 8 December 2024).
51. Microsoft. (2024). ASP.NET Core | open-source web framework for .NET. Available at: https://dotnet.microsoft.com/en-us/apps/aspnet (Accessed: 8 December 2024).
52. Microsoft. (2024). Azure Blob Storage. Available at: https://azure.microsoft.com/en-gb/products/storage/blobs (Accessed: 8 December 2024).
53. Microsoft. (2024). Azure SQL Database overview. Available at: https://docs.microsoft.com/en-us/azure/azure-sql/database/ (Accessed: 8 December 2024).
54. Microsoft. (2024). Create your Azure free account today | Microsoft Azure. Available at: https://azure.microsoft.com (Accessed: 8 December 2024).
55. Microsoft. (2024). Microsoft 365. Available at: https://www.office.com/ (Accessed: 8 December 2024).
56. Microsoft. (2024). Microsoft Teams. Available at: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software (Accessed: 8 December 2024).
57. MIT, 2024. Reading 12: user testing. Available at: https://web.mit.edu/6.813/www/sp16/classes/12-user-testing/ (Accessed: 8 December 2024).
58. Monday.com. (2024). Monday.com | your go-to work platform. Available at: https://monday.com/ (Accessed: 8 December 2024).
59. Moq (2024) 'Moq: The open source mocking framework for .NET', Moq. Available at: https://github.com/moq/moq4 (Accessed: 8 December 2024).
60. Murphy, C., 2018. A comprehensive guide to user testing. Smashing Magazine. Available at: https://www.smashingmagazine.com/2018/03/guide-user-testing/ (Accessed: 8 December 2024).
61. Murugan, M. (2021) Modular architecture in ASP.NET Core - building better monoliths - CodeWithMukesh. Available at: https://codewithmukesh.com/blog/modular-architecture-in-aspnet-core/ (Accessed: 8 December 2024).

62. N-gram (2024) Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/w/index.php?title=N-gram&oldid=1229559008 (Accessed: 8 December 2024).
63. Nielsen, J. (2000) 'Why you only need to test with 5 users', Nielsen Norman Group, 18 March. Available at: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (Accessed: 8 December 2024).
64. Nithyashree (2021) 'What are n-grams and how to implement them in python?', Analytics Vidhya, 13 September. Available at: https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/ (Accessed: 8 December 2024).
65. Noui, M.E.A. and Dehane, M. (2023). Analysing startups failure factors: Evidence from CB Insights Tech Market Intelligence Platform. Journal of Economic Growth and Entrepreneurship (JEGE), 6(1), 10-30.
66. npm (2024). About npm. Available at: https://www.npmjs.com/about (Accessed: 8 December 2024).
67. Oates, B.J. (2006) Researching information systems and computing. SAGE, London.
68. OpenAI. (2024). ChatGPT. Available at: https://www.openai.com (Accessed: 8 December 2024).
69. Patzelt et al. (2021) 'Understanding the life cycles of entrepreneurial teams and their ventures: an agenda for future research', Entrepreneurship Theory and Practice, 45(5), pp. 1119–1153. Available at: https://doi.org/10.1177/1042258720978386.
70. Plesk, 2024. MySQL vs MSSQL: Comparing Similarities and Differences. Available at: https://www.plesk.com/blog/various/mysql-vs-mssql/ (Accessed: 8 December 2024).
71. Qualtrics (2024) Qualtrics XM Platform. Available at: https://www.qualtrics.com (Accessed: 8 December 2024).
72. Rathinam, S. (2023) 'Analysis and comparison of different frontend frameworks', in S. Prabhu, S.R. Pokhrel, and G. Li (eds) Applications and Techniques in Information Security. Singapore: Springer Nature Singapore, pp. 243–257. Available at: https://doi.org/10.1007/978-981-99-2264-2_19.
73. React Hook Form (2024). React Hook Form Documentation. Available at: https://www.react-hook-form.com/ (Accessed: 8 December 2024).
74. React. (2024). React – A JavaScript library for building user interfaces. Available at: https://react.dev/ (Accessed: 8 December 2024).
75. Richards, M. (2022) Software architecture patterns. Second edition. Sebastopol, CA: O'Reilly Media.
76. Ries, E. (2011). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business.

77. Sahatqija et al. (2018) 'Comparison between relational and NOSQL databases', in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0216–0221. Available at: https://doi.org/10.23919/MIPRO.2018.8400041.
78. Salamzadeh, A. and Kawamorita Kesim, H. (2015) 'Startup companies: life cycle and challenges'. Rochester, NY. Available at: https://doi.org/10.2139/ssrn.2628861.
79. Santisteban, J. et al. (2023) 'Failure of tech startups: a systematic literature review', in Garcia, M. V. and Gordón-Gallegos, C. (eds) CSEI: International Conference on Computer Science, Electronics and Industrial Engineering (CSEI). Cham: Springer Nature Switzerland, pp. 111–126. Available at: https://doi.org/10.1007/978-3-031-30592-4_9.
80. Seppänen et al. (2017) 'Little big team: acquiring human capital in software startups', in M. Felderer et al. (eds) Product-Focused Software Process Improvement. Cham: Springer International Publishing, pp. 280–296. Available at: https://doi.org/10.1007/978-3-319-69926-4_20.
81. Seppänen, P. (2020) 'Yes, we can! Building a capable initial team for a software startup', in A. Nguyen-Duc et al. (eds) Fundamentals of Software Startups. Cham: Springer International Publishing, pp. 45–59. Available at: https://doi.org/10.1007/978-3-030-35983-6_3.
82. Sire, T. (2024) 'What is requirements traceability matrix (Rtm) - a comprehensive guide with examples', Requiment, 22 January. Available at: https://www.requiment.com/requirements-traceability-matrix-rtm-guide/ (Accessed: 8 December 2024).
83. Skala, A. (2019) 'The startup as a result of innovative entrepreneurship', in Skala, A., Digital Startups in Transition Economies. Cham: Springer International Publishing, pp. 1–40. Available at: https://doi.org/10.1007/978-3-030-01500-8_1.
84. Slack. (2024). Slack: Slack is your productivity platform. Available at: https://slack.com (Accessed: 8 December 2024).
85. Sommerville, I. (2016). Software Engineering. 10th ed., Global edition. Boston: Pearson (Always Learning).
86. Swagger, 2024. Swagger: API Documentation & Testing Tool. Available at: https://swagger.io (Accessed: 8 December 2024).
87. Telerik (2024) Arrange Act Assert. Available at: https://docs.telerik.com/devtools/justmock/basic-usage/arrange-act-assert (Accessed: 8 December 2024).
88. Thirasak, V., 2019. Building an effective startup team. In International Conference on Advances in Business and Law (ICABL) (Vol. 3, No. 1, pp. 18-27).
89. Thoughtworks. (2024) When (modular) monolith is the better way to build software. Available at: https://www.thoughtworks.com/en-gb/insights/blog/microservices/modular-monolith-better-way-build-software (Accessed: 8 December 2024)

90. Trello. (2024). Trello: Manage your team's projects from anywhere. Available at: https://trello.com/ (Accessed: 8 December 2024).
91. Tuckman, B.W. (1965) 'Developmental sequence in small groups.', Psychological Bulletin, 63(6), pp. 384–399. Available at: https://doi.org/10.1037/h0022100.
92. Turing. Word embeddings in NLP: A complete guide (2024). Available at: https://www.turing.com/kb/guide-on-word-embeddings-in-nlp (Accessed: 8 December 2024).
93. Upwork. (2024). Upwork. Available at: https://www.upwork.com/ (Accessed: 8 December 2024).
94. Visual Paradigm (2024). Ideal modeling & diagramming tool for agile team collaboration. Available at: https://www.visual-paradigm.com/ (Accessed: 8 December 2024).
95. xUnit (2024) 'xUnit.net: A free, open-source, community-focused unit testing tool for the .NET Framework', xUnit. Available at: https://xunit.net (Accessed: 8 December 2024).
96. Yocco, V., 2020. Ethical considerations in UX research: the need for training and review. Smashing Magazine. Available at: https://www.smashingmagazine.com/2020/12/ethical-considerations-ux-research/ (Accessed: 8 December 2024).

Appendices

Appendix A: Original Project Proposal

THIS PAGE INTENTIONALLY LEFT BLANK

Project Proposal:

Technology-based Startup Team Organization Platform

By: Ali Momenzadeh Kholenjani

Supervisor: Dr. Christos Kloukinas

1 Introduction

1.1 Purpose of the Work

In the ever-evolving landscape of technology-based startups, several factors contribute to the success of which efficient team organization plays a significant role. Skala (2019) notes that technology-based startups are characterized by their short-lived enterprise, novelty, rapid growth, and operation within specific industries, typically the technology market. An individual with a remarkable entrepreneurial notion occasionally possesses the required technical and organizational knowledge to bring their idea to fruition. Therefore, there is a demand to seamlessly discover skilled and reliable individuals to assemble a competitive team, a challenging task that requires a careful approach.

Based on CB Insights (2021) report, a driven startup team where responsibilities are appropriately shared seems important to maintain passion and reduce burnout. The presence of disharmony between members and the inadequacy of diverse skills within the team are also enumerated as prominent failure factors in startups, as highlighted in the same report. Moreover, the report mentions a startup's struggle to recruit key talent, particularly skilled individuals, during the early stages of the venture. Noui and Dehane (2023) further emphasize that ineffective team formation stands out as the top failure factor from an organizational standpoint, persisting throughout different startup phases.

Existing software solutions facilitating team organization offer a generic range of features for teamwork or provide professional networking opportunities. However, these solutions often fail to address the specific needs of technology startup teams, such as hiring technical experts or adapting to the fast-changing environment that impacts teamwork dynamics. Consequently, a significant gap remains in the market for a software-based solution tailored specifically to meet the unique requirements of team organization in the tech startup environment. Recognizing this shortcoming, the proposed solution aims to streamline the process of team organization for technology-based startups through a dedicated online platform.

1.2 Aims and Objectives

The primary aim of this platform is to provide a centralized hub for the tech startup community, where startup founders can successfully organize high-performing teams. Moreover, the skilled individuals will have the opportunity to showcase their skills and portfolio (to demonstrate their capabilities) and collaborate with startup teams.

More specifically, the breakdown of objectives into sub-objectives is as follows (→ shows the evaluation criteria):

1. Enabling startup founders to post job advertisements and add talented individuals to their team → *Usability tests to gather feedback on the ease of job posting process and effectiveness of adding individuals to teams*

2. Providing skilled individuals, the capability to showcase their portfolio → *Usability tests to gather feedback from skilled individuals on the ease of portfolio sharing and from founders on how this feature assisted in team formation decisions*

3. Implementing a sophisticated matching algorithm to help founders identify experienced individuals and help individuals discover the right startup projects to invest their time in → *Measuring the accuracy of matches by comparing the skills and preferences of individuals with the requirements of job postings, along with usability tests to assess user satisfaction with the matching process.*

4. Enabling goal, milestone, and role definitions in startup teams to bring more clarity to teams → *Usability tests to evaluate how these features help team members understand project objectives and their individual responsibilities.*

5. Enabling task management so that startup founders can designate tasks to team members → *Usability tests to evaluate the effectiveness of task delegation by founders and the ease with which team members manage their assigned tasks.*

6. Providing a facility to share written or file reports within the group to inform members of different concerns → *Usability tests to assess the ease of use and effectiveness of the report-sharing feature in addressing team concerns.*

7. Establishing feedback and rating mechanisms to allow users to provide valuable feedback and rate team members → *Usability tests to assess how easy the feedback and rating features are to use and their impact on team dynamics and individual performance.*

1.3 Research Question

The central question of this project is "How can a unified platform be designed and implemented to help organize efficient tech startup teams seamlessly by matching the relevant experts to the right startup founders?"

1.4 Products of the Work

The primary project deliverable is an online platform that enables startup founders and individual professionals to collaborate on startup initiatives. Additionality, architectural diagrams and documentation produced during development, and usability and performance reports generated during evaluation are the secondary products of this work.

1.5 Project Beneficiaries

1.5.1 Startup Founders

Startup founders seeking a simplified method to organize a startup team and bring their startup ideas to life will be the primary beneficiaries.

1.5.2 Skilled Individuals

Professional individuals, especially in the information technology sector, will also be the primary beneficiaries as they are offered the opportunity to collaborate in startup initiatives.

1.5.3 Investors and Accelerators

By equipping startups with streamlined tools to organize their teams and track development, the platform will indirectly assist investors and accelerators in identifying promising startups for investment.

1.5.4 Platform Operators

Platform operators will benefit through a subscription model that provides exclusive functionalities, targeted to provide additional value to the users.

1.6 Project Scope

The scope of the project encompasses the design, implementation, testing, and deployment of an online platform that enables the organization of tech startup teams. It does not involve the hiring process for team members, nor does it encompass the development of team communication capabilities provided by social media platforms. Additionally, factors beyond team organization that contribute to startup management are not within the scope.

2 Critical Context

Exploring team organization within tech-based startups presents a promising avenue for research. Although there may be a noticeable gap in the literature specific to this area, there exists ample research on closely related topics such as startup dynamics, talent acquisition practices for entrepreneurial teams, and general team dynamics. These established areas of research offer valuable insights that help demonstrate the importance of the organization of a well-rounded startup team. Consequently, the literature review will synthesize these relevant concerns, ultimately justifying the necessity for a tailored software solution.

2.2 Talent Acquisition in Entrepreneurial Ventures

A closer inspection of the startup life cycle by Patzelt et al. (2021) reveals that team creation initially occurs in the first phase of the startup when founders need to find their teammates to start the venture. While their study proposes some research questions about the formation of such teams, it does not provide any further answers to them. Another research endeavor examines different aspects of acquiring human resources in software startups, and their results highlight the importance of founding a team where complimentary skill sets of members derive innovation (Seppänen et al., 2017). These members are believed to be typically hired by strategically assessing team strength (initially the founder) and identifying skill gaps.

Seppänen (2020) while identifying the role of the founder as a person with a brilliant idea, highlights the importance of extra human resources to complement their skills, even in cases where the founder is technical. This study sheds more light on the hiring practices of technology startups, revealing a diverse range of hires from less experienced ones to more seasoned technical individuals who are either subcontracted, contracted, or provided by external companies. This decision was identified to be mainly based on the limited resources in startups that also constraint keeping the team size small.

2.3 Team Organization in Technology Startups

Building upon the insights gained from talent acquisition, the organization of teams within technology startups emerges as a critical concern. Diakanastasi et al. (2018) research on startup team dynamics identifies several factors that impact teamwork performance. Initially, the study highlights the importance of sharing close visions between members and the founder, followed by a clear set of expectations, which are influential factors for maintaining the integrity of the startup team. Then, the significance of acquiring a relevant and complementary set of skills

within the team, similar to Patzelt et al. (2021) findings, was realized to play a vital role in team dynamics. Subsequently, Diakanastasi et al. identify the importance of clear communication and the significance of a well-defined set of roles and the designation of tasks. Moreover, the number of members, their contribution to the team, the founder's characteristics, and cases when there are couples in the team can affect team dynamics, as described in their findings.

Besides, the dynamic nature of startups can necessities changes in team composition over time. As Patzelt et al. (2021) suggested, team members might decide to leave the entrepreneurial venture, or the founder may realize more competencies and resources are required to address a specific need. These circumstances can result in a situation where the startup needs to hire new members during the development, which comes with the reallocation of tasks and roles within the team. Additionally, Melegati and Kon (2020) underscore onboarding of new members as a challenge encountered by startup teams. Although the article does not delve into specifics, another study points to insufficient strategy in task and role assignments to members, along with inadequate transparency regarding teamwork activities that results in the disorganization of startups, which can be the reason for onboarding challenges (Cantamessa et al., 2018).

2.4 Necessity for a Tailored Solution

The review of existing literature revealed a conspicuous gap in research on team organization within technology startups. Despite significant attention being devoted to understanding startup dynamics and challenges, there remains a notable scarcity and ambiguity on team organization aspects which is one of the substantial requirements of technology startups. Unlike the relatively static nature of one-time events like weddings, tech startups require continuous adaptation, dynamic role assignments, and team coordination. This entails the need for a solution to address the ongoing organizational challenges of these startups such as strategic talent acquisition, dynamic team formation, clear task assignment, transparent goal setting, and seamless collaboration among team members.

The comparative analysis of current solutions for team organization in Table 1 reveals that while platforms like Microsoft Teams and Slack offer robust collaboration tools, they lack essential features such as job posting capabilities and advanced matching algorithms crucial for efficient team formation (Microsoft, 2024; Slack, 2024). Similarly, LinkedIn, despite its strengths in professional networking, falls short to provide team organization and task management functionalities (LinkedIn, 2024). Even dedicated project management tools like Asana and Trello, while proficient in certain aspects, fail to integrate essential features like job posting and effective member matching (Asana, 2024; Trello, 2024). Drawing from these insights, none of the tools adequately facilitate the creation of teams from existing platform members by matching them with relevant job postings, while providing teamwork functionalities. Hence, a tailored solution is essential to provide the matching capability while addressing the unique organizational needs of tech startups mentioned earlier.

Feature	Microsoft Teams	Slack	LinkedIn	Asana	Trello	Proposed Solution
Team Collaboration	Yes	Yes	No	Limited	Limited	Yes
Professional Networking	No	No	Yes	No	No	Yes
Job Posting	No	No	Yes	No	No	Yes
Matching Algorithm	No	No	No	No	No	Yes
Portfolio Showcase	No	No	Yes	No	No	Yes

Goal, Milestone, and Role Definitions	Limited	Limited	No	Yes	Yes	Yes
Task Management	Yes	Yes	No	Yes	Yes	Yes
Reporting Mechanism	Yes	Yes	No	Yes	Yes	Yes
Feedback and Rating Mechanisms	No	No	No	Yes	No	Yes
Subscription Model	Yes	Yes	Yes	Yes	Yes	Yes

Table 1: Comparison of Features Across Team Organization Tools (Microsoft, 2024; Slack, 2024; LinkedIn, 2024; Asana, 2024; Trello, 2024)

3 Approaches: Methods & Tools for Design, Analysis & Evaluation

Considering the solo development of the project, the time constraints, the absence of external parties for reviewing the design and development phases, and the structured timeline of the project, the Waterfall development model is appropriate to keep the project progress manageable (Akinsola et al., 2020). The sub-sections below describe each stage of the project's progress.

3.1 Preparation

The project will be initiated with a meticulous requirements analysis, informed by a comprehensive literature review and analysis of prevailing industry trends, including an examination of relevant industry tools. This is to inform the subsequent design and development processes effectively. Additionally, essential coding environments and tools will be set up, and the initial configuration of the cloud environment will be established.

3.2 Design

The design decisions will be informed by the insights gained from the requirements analysis as there is no user feedback at this stage. In addition to producing detailed use case definitions from the requirements, architectural diagrams will be created using Unified Modeling Language (UML) (Object Management Group, 2024) to provide a clear visualization of the system's structure and its components. Moreover, the skeleton of the front end will be prototyped to provide a basic structure for user interaction and interface design. Simultaneously, the structure of the solution for the backend is being scaffolded. This ensures that necessary components and architectural decisions are in place for seamless development. The blueprint of the platform designed during this phase allows a smooth transition into actual coding and development.

Since the architecture can significantly influence the platform's scalability, performance, and maintainability, it should be carefully considered. Richards (2022, Chapter 1) suggests that smaller systems would benefit better from a monolith architecture, despite scalability limitations. Furthermore, according to Richards, distributed alternatives such as microservices pose complexity in transaction management and data consistency. Given the project's small scale, a monolith architecture would be more suitable. To enhance maintainability and scalability, a newer implementation of the monolith architecture, known as the modular monolith that is illustrated by Richards and explained by GeeksforGeeks (2024), will be adopted. Additionally, this architecture can serve as a stepping stone toward transitioning to microservices architecture if the project grows in complexity in the future (Thoughtworks, 2024).

3.3 Development and Deployment

Regarding the technology choices, the .NET software development ecosystem, particularly ASP.NET Core (Microsoft, 2024), will be utilized for the backend development, due to previous familiarity and widespread usage for robust backend software development. For the frontend development, React.js is chosen to ensure user-friendliness provided its moderate learning curve, strong community, and performance optimization based on the utilization of virtual DOM (React, 2024; Rathinam, 2023). Version control will be managed through Git, with GitHub serving as the repository for tracking changes, code reviews, and regular backups. Microsoft SQL Server (Microsoft, 2024) will be used as the relational database to ensure ACID properties (Sahatqija et al., 2018), particularly due to its robust integration with the .NET technology stack (Plesk, 2024).

After the development phase, containerization via Docker will be implemented to ensure portability and consistent deployment in the cloud environment (Docker, 2022). Both frontend and backend components will be containerized individually allowing for separate deployment. Subsequently, Kubernetes (Kubernetes, 2024) will be utilized within the cloud to effectively manage the scaling requirements of the application. Although the backend architecture is implemented as a modular monolith, this deployment approach where the backend and frontend are deployed separately seems to offer relative scalability benefits akin to microservices. Finally, Microsoft Azure will be used as the cloud platform due to seamless integration with .NET technologies and its more reliable support of Microsoft SQL Server compared with the alternatives (Microsoft, 2024; Kingsley, 2024).

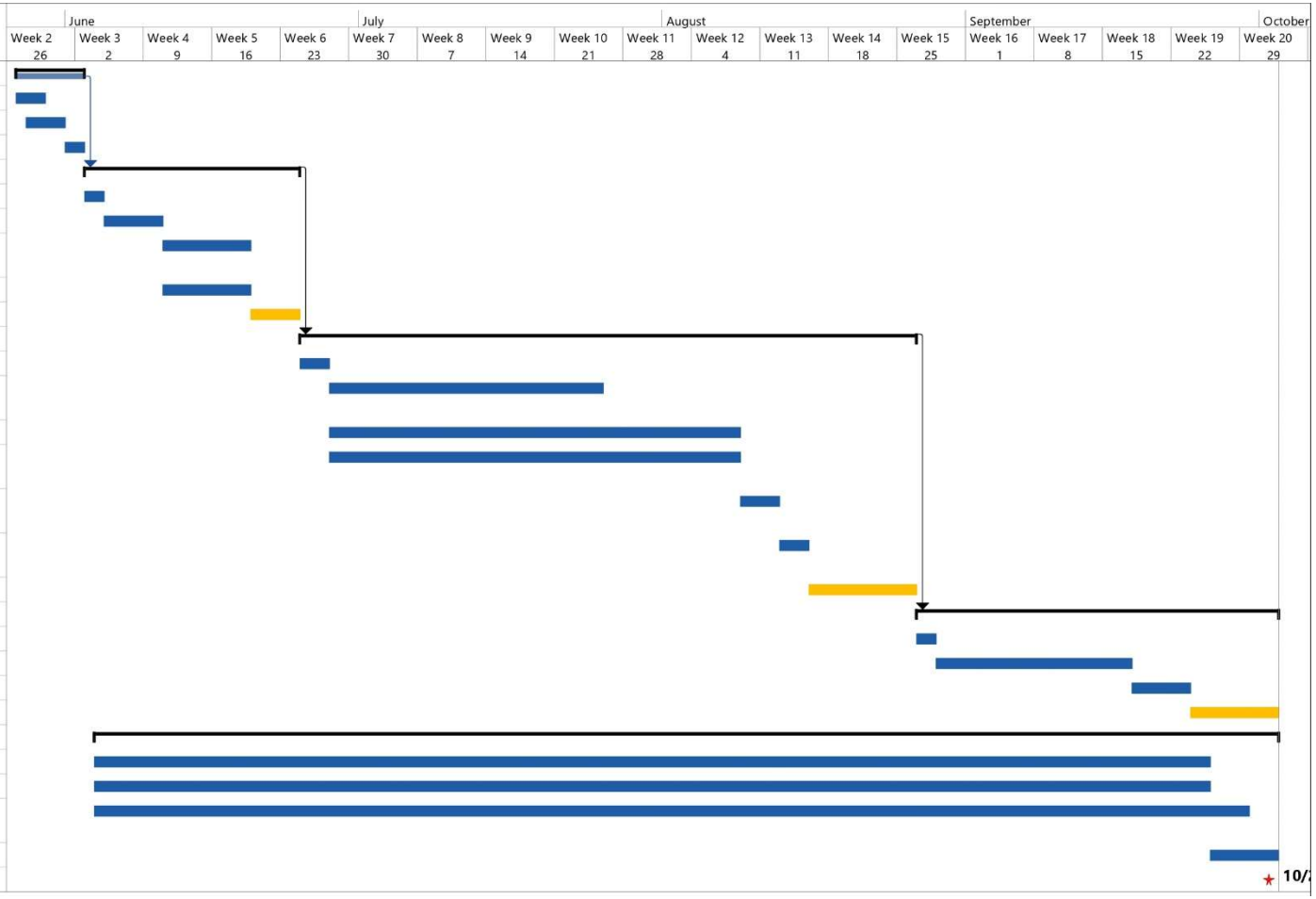
3.4 Analysis and Evaluation

Comprehensive unit and integration testing guides quality software development for which test scripts will be developed to validate expected outcomes and identify any inconsistencies in the software during the development. Also, performance metrics like response time and loading speeds will be measured in varying conditions to ensure the platform meets expected benchmarks and delivers a satisfactory user experience. Following the usability tests, which are guided by established usability principles and involve individual participants, the effectiveness of the platform will be assessed. Subsequently, users' feedback and comments, combined with the earlier performance tests, will contribute to evaluating the overall effectiveness of the platform.

4 Work Plan

Below is a detailed Gantt chart of the work plan, spanning 18 weeks from 27/05/2024 to 02/10/2024, accounting for approximately 600 hours, including some buffer time. Report writing will be conducted concurrently with the project’s progression, and regular meetings with the supervisor will be held to discuss progress. The project plan is flexible and may be adjusted at each stage based on new insights and developments.

ID	Task Name	Duration	Start	Finish
1	**1. Preparation**	**6 days**	**Mon 5/27/24**	**Sun 6/2/24**
2	Conduct Literature Review	3 days	Mon 5/27/24	Wed 5/29/24
3	Perform Requirements Analysis	4 days	Tue 5/28/24	Fri 5/31/24
4	Set Up Development Environme	2 days	Sat 6/1/24	Sun 6/2/24
5	**2. Design**	**16 days**	**Mon 6/3/24**	**Mon 6/24/24**
6	Define and Write Use Cases	2 days	Mon 6/3/24	Tue 6/4/24
7	Create UML Architectural Diagra	4 days	Wed 6/5/24	Mon 6/10/24
8	Prototype Front-end Template (without React.js)	7 days	Tue 6/11/24	Wed 6/19/24
9	Scaffold the Backend Architectu	7 days	Tue 6/11/24	Wed 6/19/24
10	Buffer Time	3 days	Thu 6/20/24	Mon 6/24/24
11	**3. Development and Deployment**	**45 days**	**Tue 6/25/24**	**Mon 8/26/24**
12	Review React.js Documentation	3 days	Tue 6/25/24	Thu 6/27/24
13	Develop Front-end (Convert Template to React.js)	20 days	Fri 6/28/24	Thu 7/25/24
14	Develop Backend	30 days	Fri 6/28/24	Thu 8/8/24
15	Conduct Unit and Integration Testing	30 days	Fri 6/28/24	Thu 8/8/24
16	Containerize Front-end and Backend	2 days	Fri 8/9/24	Mon 8/12/24
17	Deploy on Kubernetes Engine on Azure	3 days	Tue 8/13/24	Thu 8/15/24
18	Buffer Time	7 days	Fri 8/16/24	Mon 8/26/24
19	**5. Analysis and Evaluation**	**27 days**	**Tue 8/27/24**	**Wed 10/2/24**
20	Measure Performance Metrics	2 days	Tue 8/27/24	Wed 8/28/24
21	Conduct Usability Testing	14 days	Thu 8/29/24	Tue 9/17/24
22	Analyze Results	4 days	Wed 9/18/24	Mon 9/23/24
23	Buffer Time	7 days	Tue 9/24/24	Wed 10/2/24
24	**5. Documentation**	**87 days**	**Tue 6/4/24**	**Wed 10/2/24**
25	Write Report	**82 days**	Tue 6/4/24	Wed 9/25/24
26	Meet with Supervisor	83 days	Tue 6/4/24	Thu 9/26/24
27	Make Adjustments Based on Supervisor's Guidance	85 days	Tue 6/4/24	Sun 9/29/24
28	Final Review and Editing	5 days	Thu 9/26/24	Wed 10/2/24
29	Submit Final Report	0 days	Wed 10/2/24	Wed 10/2/24



5 Risks

Possible risks that could affect the successful delivery of the project are listed in the table below.

#	Risk Description	Likelihood of failure (L) (1-5)	Impact of failure (I) (1-5)	Severity (S = L × I) (1-25)	Mitigation Strategy
1	Software/Hardware failures resulting in loss of project data	1	5	5	Using GitHub to maintain a remote backup of the code, in addition to saving regular backups on an external SDD drive.
2	Breaking changes in the libraries and frameworks	2	5	10	Utilizing package management tools such as NuGet and npm. Also, incorporating Automated tests.
3	Technical complexity of implementation exceeding expertise	2	5	10	Reevaluating the project scope with the supervisor's guidance in the design phase.
4	Failure to deliver sufficient results	2	4	8	Regular discussions with the supervisor and reconsidering the project scope.
5	Insufficient user feedback to test the project properly	2	5	10	Automated testing of the system against requirements
6	Security breaches and data leaks	2	4	8	Implementing Two-factor authentication as well as anonymizing the users' data
7	Supervisor unavailability	2	3	6	Using online communication
8	Budget overruns for cloud services	3	4	12	Careful monitoring of resources and checking of cloud credits

Table 2: Risks Table

1: Low 5, 25: High

6 Ethical Considerations

The ethics checklist attached shows that the project requires the involvement of human participants to conduct usability tests. This procedure will be addressed carefully to ensure the protection and well-being of the individuals. Moreover, two-factor authentication, data encryption, and providing transparent terms of services and policies of usage to the users when utilizing the platform ensures data protection.

References

1. Akinsola et al. (2020) 'Comparative analysis of software development life cycle models(Sdlc)', in R. Silhavy (ed.) Intelligent Algorithms in Software Engineering. Cham: Springer International Publishing, pp. 310–322. Available at: https://doi.org/10.1007/978-3-030-51965-0_27.
2. Asana. (2024). Asana: Manage your team's work, projects, & tasks online. Available at: https://asana.com (Accessed: 18 May 2024).
3. CB Insights. (2021). Report: "The Top 12 Reasons Startups Fail". URL: https://www.cbinsights.com/research/report/startup-failure-reasons-top. (accessed: 05.05.2024).
4. Cantamessa et al. (2018) 'Startups' roads to failure', Sustainability, 10(7), p. 2346. Available at: https://doi.org/10.3390/su10072346.
5. Diakanastasi et al. (2018) 'Entrepreneurial team dynamics and new venture creation process: an exploratory study within a start-up incubator', SAGE Open, 8(2), p. 215824401878144. Available at: https://doi.org/10.1177/2158244018781446.
6. Docker. (2022). Accelerated container application development. Available at: https://www.docker.com/ (Accessed: 18 May 2024).
7. Noui, M.E.A. and Dehane, M. (2023). Analysing startups failure factors: Evidence from CB Insights Tech Market Intelligence Platform. Journal of Economic Growth and Entrepreneurship (JEGE), 6(1), 10-30.
8. GeeksforGeeks. (2024) What is a modular monolith? Available at: https://www.geeksforgeeks.org/what-is-a-modular-monolith/ (Accessed: 18 May 2024).
9. Kingsley, M.S. (2024) 'Comparing aws, azure, and gcp', in Kingsley, M. S., Cloud Technologies and Services. Cham: Springer International Publishing, pp. 381–393. Available at: https://doi.org/10.1007/978-3-031-33669-0_12.
10. Kubernetes. (2024). Production-grade container orchestration. Available at: https://kubernetes.io/ (Accessed: 18 May 2024).
11. LinkedIn. (2024). LinkedIn. Available at: https://www.linkedin.com (Accessed: 18 May 2024).
12. Melegati, J. and Kon, F. (2020). Early-Stage Software Startups: Main Challenges and Possible Answers. In: Nguyen-Duc, A., Münch, J., Prikladnicki, R., Wang, X., Abrahamsson, P. (eds) Fundamentals of Software Startups. Springer, Cham. https://doi.org/10.1007/978-3-030-35983-6_8
13. Microsoft. (2024). ASP.NET Core | open-source web framework for .NET. Available at: https://dotnet.microsoft.com/en-us/apps/aspnet (Accessed: 18 May 2024).
14. Microsoft. (2024). Create your Azure free account today | Microsoft Azure. Available at: https://azure.microsoft.com (Accessed: 18 May 2024).
15. Microsoft. (2024). SQL Server Downloads | Microsoft. Available at: https://www.microsoft.com/en-gb/sql-server/sql-server-downloads (Accessed: 18 May 2024).
16. Microsoft. (2024). Microsoft Teams. Available at: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software (Accessed: 18 May 2024).
17. Object Management Group (OMG). (2024). Unified Modeling Language (UML), Version 2.5.1. Available at: https://www.omg.org/spec/UML/2.5.1/About-UML (Accessed: 18 May 2024).
18. Patzelt et. al. (2021) 'Understanding the life cycles of entrepreneurial teams and their ventures: an agenda for future research', Entrepreneurship Theory and Practice, 45(5), pp. 1119–1153. Available at: https://doi.org/10.1177/1042258720978386.
19. Plesk, 2024. MySQL vs MSSQL: Comparing Similarities and Differences. Available at: https://www.plesk.com/blog/various/mysql-vs-mssql/ (Accessed: 18 May 2024).
20. React. (2024). Available at: https://react.dev/ (Accessed: 18 May 2024).
21. Rathinam, S. (2023) 'Analysis and comparison of different frontend frameworks', in S. Prabhu, S.R. Pokhrel, and G. Li (eds) Applications and Techniques in Information Security. Singapore: Springer Nature Singapore, pp. 243–257. Available at: https://doi.org/10.1007/978-981-99-2264-2_19.
22. Richards, M. (2022) Software architecture patterns. Second edition. Sebastopol, CA: O'Reilly Media.
23. Sahatqija et al. (2018) 'Comparison between relational and NOSQL databases', in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0216–0221. Available at: https://doi.org/10.23919/MIPRO.2018.8400041.
24. Seppänen et al. (2017) 'Little big team: acquiring human capital in software startups', in M. Felderer et al. (eds) Product-Focused Software Process Improvement. Cham: Springer International Publishing, pp. 280–296. Available at: https://doi.org/10.1007/978-3-319-69926-4_20.
25. Seppänen, P. (2020) 'Yes, we can! Building a capable initial team for a software startup', in A. Nguyen-Duc et al. (eds) Fundamentals of Software Startups. Cham: Springer International Publishing, pp. 45–59. Available at: https://doi.org/10.1007/978-3-030-35983-6_3.
26. Skala, A. (2019) 'The startup as a result of innovative entrepreneurship', in Skala, A., Digital Startups in Transition Economies. Cham: Springer International Publishing, pp. 1–40. Available at: https://doi.org/10.1007/978-3-030-01500-8_1.
27. Slack. (2024). Slack: Slack is your productivity platform. Available at: https://slack.com (Accessed: 18 May 2024).
28. Thoughtworks. (2024) When (modular) monolith is the better way to build software. Available at: https://www.thoughtworks.com/en-gb/insights/blog/microservices/modular-monolith-better-way-build-software (Accessed: 18 May 2024)
29. Trello. (2024). Trello: Manage your team's projects from anywhere. Available at: https://trello.com/ (Accessed: 18 May 2024).

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.
The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk.
The approval may be ***provisional –*** *identifying the planned research as* likely to involve MINIMAL RISK.
In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

	A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/	*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
	A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/	*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**

2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**YES**

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be ***provisional***. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with ***full approval*** of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions must be answered fully. All grey instructions must be removed.		*Delete as appropriate*
1.1.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES**
1.2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**YES**
1.3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES**
1.4	Will consent be obtained from the participants in your project? Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc.	**YES**
1.5	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	**YES**

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? *If **YES**, you must provide details of how your safety will be ensured.*	**NO**

B.3 Attachments **ALL of the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**	***YES***	***NO***	***Not Applicable***

Details on how safety will be assured in any non-University location, including risk assessment if required (see B2)			✓
Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) *Any personal data must be acquired, stored and made accessible in ways that are GDPR compliant.*	✓		
Full protocol for any workshops or interviews**			✓
Participant information sheet(s)**	✓		
Consent form(s)**	✓		
Questionnaire(s)** *sharing a Qualtrics survey with your supervisor is recommended.*	✓		
Topic guide(s) for interviews and focus groups**			✓
Permission from external organisations or Head of Department** *e.g. for recruitment of participants*			✓

***If these items are not available at the time of submitting your project proposal, then **provisional approval** can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. **All** such items **must** be seen and approved by your supervisor before the activity for which they are needed begins. Written evidence of **final approval** of your planned activity must be acquired from your supervisor before you commence.*

Changes

If your plans change and any aspects of your research that are documented in the approval process change as a consequence, then any approval acquired is invalid. If issues addressed in Part A (the checklist) are affected, then you must complete the approval process again and establish the kind of approval that is required. If issues addressed in Part B are affected, then you must forward updated documentation to your supervisor and have received written confirmation of approval of the revised activity before proceeding.

Templates for Consent and Information

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. You **must** adapt them according to the needs of your project before you submit them for consideration.

Participant Information Sheets, Consent Forms and Protocols must be consistent. Please ensure that this is the case prior to seeking approval. Failure to do so will slow down the approval process.

We strongly recommend using Qualtrics to produce digital information sheets and consent forms.

Further Information

http://www.city.ac.uk/department-computer-science/research-ethics

https://www.city.ac.uk/research/ethics/how-to-apply/participant-recruitment

https://www.city.ac.uk/research/ethics

Appendix B: Requirements

B.1 Functional Requirements

The table below outlines the platform requirements, categorized by key functional areas such as User Management, Job Management, Portfolio Management, Team Management, Matching Algorithm, and Task Management. These requirements are essential features needed to support the primary functionalities of the platform and are prioritized and incrementally implemented for better organization and clarity. Although Task Management is part of Team Management, it is listed separately for clearer organization and emphasis.

Category	Requirement	Priority	Increment
User Management	1.1 Users should be able to create an account using their email address and password. 1.2 Users should also be able to register using their Google account credentials. 1.3 Email verification should be implemented for security. 1.4 Users can select their role as a startup founder or skilled individual during registration.	High	First
	1.5 Users should be able to update their profile. 1.6 Users should be able to view the profiles of all users registered on the platform.	High	First
	1.7 Startup founders should have access to post job advertisements, review job applications, and manage their teams. 1.8 Skilled individuals should have access to manage their portfolio and apply for jobs, as well as viewing their team.	High	First
Job Management	2.1 Startup founders should be able to create, update, and delete job advertisements. 2.2 Job advertisements should include details of the startup like startup name, stage, industry, and job details like title, description, required skills, and application deadline. 2.3 Job advertisements should be visible to all users.	High	First
	2.4 Skilled individuals should be able to apply for jobs. 2.5 Startup founders should be able to review and accept or decline job applications.	High	First
Portfolio Management	3.1 Skilled individuals should be able to create, update, and delete portfolio items. 3.2 Portfolio items should include title, description, skills, industry, and other relevant details. 3.3 Portfolios should be visible to all users.	High	First
Team Management	4.1 Startup founders should be able to create, update, and delete teams.	High	Second
	4.2 Startup founders should be able to add skilled individuals to their team as members based on accepted job applications. 4.3 Startup founder should be able to manage member roles. 4.4 Skilled individuals should be able to view their assigned teams.	High	Second
	4.5 Startup founders should be able to manage team goals and milestones.	Medium	Third

Matching Algorithm	5.1 Startup founders should be able to find experienced individuals based on job applications. 5.2 Skilled individuals should be able to find suitable job advertisements based on their portfolio. 5.3 The algorithm should consider factors such as the individual's skills, experience, and curriculum vitae.	High	Second
Task Management	6.1 Startup founders should be able to create, assign, update, and delete tasks. 6.2 Team members should be able to view and update their assigned tasks.	Low	Third

B.2 Nonfunctional Requirements

Aspect	Requirement	Priority
Scalability	1. The system should be designed to scale efficiently to accommodate increasing numbers of users and data volumes. 2. The system should support flexible resource allocation to adapt to varying loads and maintain consistent performance.	High
Security	3. User data should be encrypted to protect sensitive information. 4. Access to sensitive features should be restricted using role-based access control.	High
Compliance	5. Compliance with relevant data protection regulations, such as General Data Protection Regulation (GDPR), should be ensured. 6. Clear terms of service and a privacy policy should be provided to users.	High
Data Storage	7. Application data, including user information, job advertisements, portfolios, etc., should be stored securely in a relational database. 8. Documents and images should be stored in a scalable storage solution to handle large volumes of data efficiently and ensure scalability.	High
Maintainability	9. The codebase should be modular and well-documented to facilitate maintenance. 10. Automated tests, including unit tests and integration tests, should be implemented to ensure code quality and detect issues early.	High
Usability	11. The user interface should be intuitive, easy to navigate, and provide a user-friendly environment with a smooth user experience.	High
Accessibility	12. The system should be accessible globally to users with a reliable internet connection.	High
Reliability	13. The system should have an uptime of 99.9% to achieve high availability and reliability. 14. Regular backups should be configured to ensure data integrity and availability.	Medium
Extensibility	15. The system should be designed to allow easy addition of new features in the future. 16. A modular architecture should be utilized to support scalability and flexibility.	Medium
Performance	17. The system should handle up to 1000 concurrent users without performance degradation. 18. Page load times should not exceed 2 seconds under normal conditions.	Low
Compatibility	19. The system should be compatible with modern web browsers (Chrome, Firefox, Safari, Edge). 20. Responsive design should be ensured to support various devices (desktops, tablets, smartphones).	Low

Appendix C: Use Case Definitions

This section details the interactions between users and the system, describing each use case and how the system enables user actions and meets functional requirements.

Title: User Account Registration
ID: 01
Description: Allows new users to create an account.
Actor: Visitor
Preconditions: Visitor must have access to the registration page.
Flow of Events: 1. Visitor navigates to the registration page. 2. Visitor selects the registration method either email/password or Google Sign-up. 3. **IF** Google Sign-up is selected **THEN** 3.1 Visitor is redirected to Google's authentication page. 3.2 Visitor authenticates with Google. 3.3 System retrieves user details from Google (email, name, profile picture). 3.4 Visitor is prompted to select their role as "Startup Founder" or "Skilled Individual". 4. **IF** Email/Password Sign-Up is selected **THEN** 4.1 Visitor fills in the registration form with the necessary details (e.g., email, name, password) and selects their role as "Startup Founder" or "Skilled Individual". 4.2 Visitor submits the registration form. 4.3 System validates the input data. 4.4 **IF** the information is valid **THEN** 4.4.1 System creates a new user account. 4.4.2 System sends a confirmation email to the visitor including a verification link. 4.4.3 Visitor confirms their account. 4.4.4 System verifies the confirmation. 4.4.5 System signs in the user and redirects them to their dashboard. 4.5 **ELSE** 4.5.1 System prompts the visitor to correct the information. 4.5.2 Visitor revises and resubmits the form. 4.5.3 Go back to 4.3.
Postconditions: User account is created and verified, and the user is signed into their dashboard.

Title: User Account Sign-in
ID: 02
Description: Allows existing users to log into their account.
Actor: User
Preconditions: User must have a registered account.
Flow of Events: 1. User navigates to the sign-in page. 2. User enters their email and password or uses Google credentials. 3. System validates credentials. 4. **IF** credentials are valid **THEN** 4.1 System signs in the user and redirects them to their dashboard. 5. **ELSE** 5.1 System prompts the user to correct the information. 5.2 User revises and retry login. 5.3 Go back to 3.
Postconditions: User is signed in and redirected to the dashboard.

Title: User Account Sign-out
ID: 03
Description: Allows users to log out of their account.
Actor: User
Preconditions: User must be signed in.
Flow of Events: 1. User clicks the sign-out button. 2. System terminates the user's session. 3. User is redirected to the sign-in page.
Postconditions: User is logged out and must sign in again to access their account.

Title: User Profile Management
ID: 04
Description: Allows users to view and update their profile information.
Actor: User
Preconditions: User must be signed in.

Flow of Events:

1. User navigates to the profile page.
2. User views their profile information.
3. User updates their profile information (e.g., name, email, profile picture).
4. User submits the updated information.
5. System validates and updates the user's profile.

Postconditions: User's profile information is updated.

Title: Viewing User Public Profiles

ID: 05

Description: Allows all users to view the public profiles of registered startup founders and skilled individuals.

Actor: User

Preconditions: User must be signed in.

Flow of Events:

1. User navigates to the public profiles section.
2. User selects a profile to view.
3. System displays the public profile details

3.1 For Startup Founders

3.1.1 System displays the startup founder's information, including their name, startup details, and list of job advertisements.

3.2 For Skilled Individuals

3.2.1 System displays the skilled individual's information, including their name, skills, and list of portfolio items.

Postconditions: Public profiles of startup founders and skilled individuals are visible to all users.

Title: Managing Job Advertisements

ID: 06

Description: Allows startup founders to create, update, and delete job advertisements.

Actor: Startup Founder

Preconditions: Startup founder must be signed in.

Flow of Events:

6. Startup founder navigates to the job advertisements page.

7. Startup founder selects the create, edit, or delete option.
8. **IF** creating a new job advertisement **THEN**
   8.1 Startup founder enters startup details (e.g., startup name, stage, industry) and job details (e.g., title, description, required skills, and application deadline).
   8.2 System saves the new job advertisement and makes it visible to all users.
9. **IF** updating an existing job advertisement **THEN**
   9.1 Startup founder selects the job advertisement to update.
   9.2 Startup founder modifies the job advertisement details as needed.
   9.3 System saves the updated advertisement.
10. **IF** deleting a job advertisement **THEN**
   10.1 Startup founder selects the job advertisement to delete.
   10.2 System prompts for confirmation.
   10.3 Startup founder confirms the deletion.
   10.4 System deletes the job advertisement.

Postconditions: Job advertisement is created, updated, or deleted as per the startup founder's action.

Title: Submitting Job Applications

ID: 07

Description: Allows skilled individuals to apply for job advertisements.

Actor: Skilled Individual

Preconditions: Skilled individual must be signed in and have a suitable portfolio.

Flow of Events:

1. Skilled individual navigates to the job listing page.
2. Skilled individual selects a job advertisement to apply for.
3. Skilled individual submits an application with necessary details (e.g., resume, cover letter).
4. System saves the application and makes it visible to the startup founder.

Postconditions: Application is submitted.

Title: Managing Job Applications

ID: 08

Description: Allows startup founders to review, accept, or reject job applications, and allows skilled individuals to accept or reject job offers.

Actor: Startup Founder, Skilled Individual

Preconditions:

Startup founder must be signed in and have received job applications.

Skilled individual must be signed in and have submitted a job application.

Flow of Events:

1. Startup founder navigates to the received job applications page.
2. Startup founder selects an application to review.
3. Startup founder updates the application status.
4. **IF** the status is updated to "Interview Scheduled" **THEN**
   - 4.1 Startup founder schedules a date for the interview.
   - 4.2 System records the interview details.
   - 4.3 System displays the interview schedule to the skilled individual in their dashboard.
5. **IF** the status is updated to "Accepted" **THEN**
   - 5.1 System saves the changes.
   - 5.2 Skilled individual can choose to either accept or reject the offer.
6. **IF** the status is updated to "Rejected" **THEN**
   - 6.1 The application status is updated to "Rejected" in the system.
   - 6.2 No further action is required from the skilled individual.
7. **IF** skilled individual updates application status to "Withdrawn" **THEN**
   - 7.1 System checks the previous application status.
   - 7.2 **IF** previous status was a final offer status (e.g., Accepted, Reject) **THEN**
     - 7.2.1 System disallows the withdrawal.
   - 7.3 **ELSE**
     - 7.3.1 Application is withdrawn, and the system saves the changes.

Postconditions:

Application status is updated to reflect the latest action.

Skilled individual's response to an offer is recorded if applicable.

The interview is scheduled if applicable.

Title: Managing Portfolio Items
ID: 09
Description: Allows skilled individuals to create, update, and delete their portfolios.
Actor: Skilled Individual
Preconditions: Skilled individual must be signed in.

Flow of Events:

1. Skilled individual navigates to the portfolio management page.
2. Skilled individual selects the add, update, or delete option.
3. **IF** adding a new item to the portfolio **THEN**
   - 3.1 Skilled individual enters details of the new item (e.g., title, description, skills used, etc.).
   - 3.2 System saves the new item to their portfolio and makes it visible to all users.
4. **IF** updating an existing item in the portfolio **THEN**
   - 4.1 Skilled individual selects the item to update.
   - 4.2 Skilled individual modifies the details as needed.
   - 4.3 System saves the updated item.
5. **IF** removing an item from the portfolio **THEN**
   - 5.1 Skilled individual selects the item to delete.
   - 5.2 System prompts for confirmation.
   - 5.3 Skilled individual confirms the deletion.
   - 5.4 System removes the item from the portfolio.

Postconditions: Items are added, updated, or deleted from the skilled individual's portfolio as per their action.

Title: Finding Matches Using Matching Algorithm

ID: 10

Description: Allows startup founders and skilled individuals to find suitable matches based on job requirements and portfolio details.

Actor: Startup Founder, Skilled Individual

Preconditions: Actor must be signed in and have relevant data defined (job requirements for startup founders, portfolio for skilled individuals).

Flow of Events:

1. **For Startup Founders Finding Skilled Individuals**
   - 1.1 Startup founder selects a job advertisement and clicks "Find Matches".
   - 1.2 System applies the matching algorithm to identify suitable skilled individuals.
   - 1.3 System displays a list of matched skilled individuals to the startup founder, sorted in descending order based on their score.
2. **For Skilled Individuals Finding Job Advertisements**
   - 2.1 Skilled individual has completed their portfolio and navigates to the job matches page.

2.2 System applies the matching algorithm to find relevant job advertisements.

2.3 System presents a list of matched job advertisements to the skilled individual, sorted in descending order based on their score.

Postconditions:

Startup founders receive a list of suitable skilled individuals.

Skilled individuals receive a list of relevant job advertisements.

Title: Managing Startup Teams

ID: 11

Description: Allows startup founders to create, update, and delete teams.

Actor: Startup Founder

Preconditions: Startup founder must be signed in.

Flow of Events:

1. Startup founder navigates to the team management page.
2. Startup founder selects the add, update, or delete option.
3. **IF** creating a team **THEN**
   - 3.1 Startup founder selects the option to create a team.
   - 3.2 Startup founder enters team details (e.g., title, description).
   - 3.3 System saves the new team.
4. **IF** updating an existing team **THEN**
   - 4.1 Startup founder selects the team to update.
   - 4.2 Startup founder modifies the team details as needed.
   - 4.3 System saves the updated team.
5. **IF** deleting a team **THEN**
   - 5.1 Startup founder selects the team to delete.
   - 5.2 System prompts for confirmation.
   - 5.3 Startup founder confirms the deletion.
   - 5.4 System deletes the team.

Postconditions: Teams are created, updated, or deleted as per the startup founder's actions.

Title: Managing Team Members and Viewing Assigned Teams

ID: 12

Description: Allows startup founders to add or remove team members based on accepted job applications and allows skilled individuals to view their assigned teams.

Actor: Startup Founder, Skilled Individual

Preconditions: Startup founder must be signed in and have created a team. Skilled individual must be signed in and assigned to a team.

Flow of Events:

1. **For Startup Founders Managing Members**
   - 1.1 Startup founder navigates to the member management section for the selected team.
   - 1.2 Startup founder selects the add or remove option.
   - 1.3 **IF** adding a member **THEN**
     - 1.3.1 Startup founder selects the option to add a member.
     - 1.3.2 Startup founder selects a skilled individual based on accepted job applications and assign them a role.
     - 1.3.3 System adds the selected individual to the team.
   - 1.4 **IF** removing a member **THEN**
     - 1.4.1 Startup founder selects the team member to remove.
     - 1.4.2 System prompts for confirmation.
     - 1.4.3 Startup founder confirms the removal.
     - 1.4.4 System removes the member from the team.
2. **For Skilled Individuals Viewing Assigned Teams**
   - 2.1 Skilled individual navigates to the "My Teams" section.
   - 2.2 System displays a list of teams the skilled individual is assigned to, along with relevant team details (e.g., team name, role, goals).

Postconditions:

Team members are added or removed as per the startup founder's actions.

Skilled individuals can view their assigned teams.

Title: Managing Team Roles

ID: 13

Description: Allows startup founders to create, update, and delete roles within the team.

Actor: Startup Founder

Preconditions: Startup founder must be signed in and have created a team.

Flow of Events:

1. Startup founder navigates to the role management section for the selected team.
2. Startup founder select the add, update, or delete option.
3. **IF** creating a role **THEN**
   3.1 Startup founder selects the option to create a role.
   3.2 Startup founder enters the role details (e.g., role name, description).
   3.3 System saves the new role.
4. **IF** updating an existing role **THEN**
   4.1 Startup founder selects the role to update.
   4.2 Startup founder modifies the role details as needed.
   4.3 System saves the updated role.
5. **IF** deleting a role **THEN**
   5.1 Startup founder selects the role to delete.
   5.2 System prompts for confirmation.
   5.3 Startup founder confirms the deletion.
   5.4 System deletes the role.

Postconditions: Team roles are created, updated, or deleted as per the startup founder's actions.

Title: Managing Team Goals
ID: 14
Description: Allows startup founders to create, update, and delete team goals.
Actor: Startup Founder
Preconditions: Startup founder must be signed in and have created a team.

Flow of Events:

1. Startup founder navigates to the goal management section for the selected team.
2. Startup founder selects the add, update, or delete option.
3. **IF** creating a goal **THEN**
   3.1 Startup founder selects the option to create a goal.
   3.2 Startup founder enters the goal details (e.g., goal title, description).
   3.3 System saves the new goal.
4. **IF** updating an existing goal **THEN**
   4.1 Startup founder selects the goal to update.
   4.2 Startup founder modifies the goal details as needed.
   4.3 System saves the updated goal.

5. **IF** deleting a goal **THEN**
   5.1 Startup founder selects the goal to delete.
   5.2 System prompts for confirmation.
   5.3 Startup founder confirms the deletion.
   5.4 System deletes the goal.

Postconditions: Team goals are created, updated, or deleted as per the startup founder's actions.

Title: Managing Team Milestones

ID: 15

Description: Allows startup founders to create, update, and delete team milestones.

Actor: Startup Founder

Preconditions: Startup founder must be signed in and have created a team.

Flow of Events:

1. Startup founder navigates to the milestone management section for the selected team.
2. Startup founder selects the add, update, or delete option.
3. **IF** creating a milestone **THEN**
   3.1 Startup founder selects the option to create a milestone.
   3.2 Startup founder enters the milestone details (e.g., milestone title, description, related goal).
   3.3 System saves the new milestone.
4. **IF** updating an existing milestone **THEN**
   4.1 Startup founder selects the milestone to update.
   4.2 Startup founder modifies the milestone details as needed.
   4.3 System saves the updated milestone.
5. **IF** deleting a milestone **THEN**
   5.1 Startup founder selects the milestone to delete.
   5.2 System prompts for confirmation.
   5.3 Startup founder confirms the deletion.
   5.4 System deletes the milestone.

Postconditions: Team milestones are created, updated, or deleted as per the startup founder's actions.

Title: Managing Team Tasks

ID: 16

Description: Allows startup founders to create, assign, update, and delete tasks, and enable team members to view and update their assigned tasks.

Actor: Startup Founder, Team Member

Preconditions: Actor must be signed in.

Flow of Events:

1. **For Startup Founders Managing Tasks**
   - 1.1 Startup founder navigates to the task management section for the selected team.
   - 1.2 Startup founder selects the create, assign, update, or delete option.
   - 1.3 **IF** creating a task **THEN**
     - 1.3.1 Startup founder selects the option to create a task.
     - 1.3.2 Startup founder enters task details (e.g., title, description, due date).
     - 1.3.3 Startup founder assigns the task to one or more team members.
     - 1.3.4 System saves the task and updates the task list for the team members.
   - 1.4 **IF** updating an existing task **THEN**
     - 1.4.1 Startup founder selects the task to update.
     - 1.4.2 Startup founder modifies the task details as needed.
     - 1.4.3 System saves the updates and updates the task list.
   - 1.5 **IF** deleting a task **THEN**
     - 1.5.1 Startup founder selects the task to delete.
     - 1.5.2 System prompts for confirmation.
     - 1.5.3 Startup founder confirms the deletion.
     - 1.5.4 System deletes the task and updates the task list.
2. **For Team Members Viewing and Updating Tasks**
   - 2.1 Team member navigates to the “My Tasks” section.
   - 2.2 Team member views assigned tasks.
   - 2.3 Team member updates task progress or details as needed.
   - 2.4 System saves updates and reflects changes in the task list.

Postconditions:

Tasks are created, updated, or deleted as per the startup founder’s or team member’s actions.

Team members’ task lists are updated to reflect the changes.

Appendix D: Requirements Traceability Matrix (RTM)

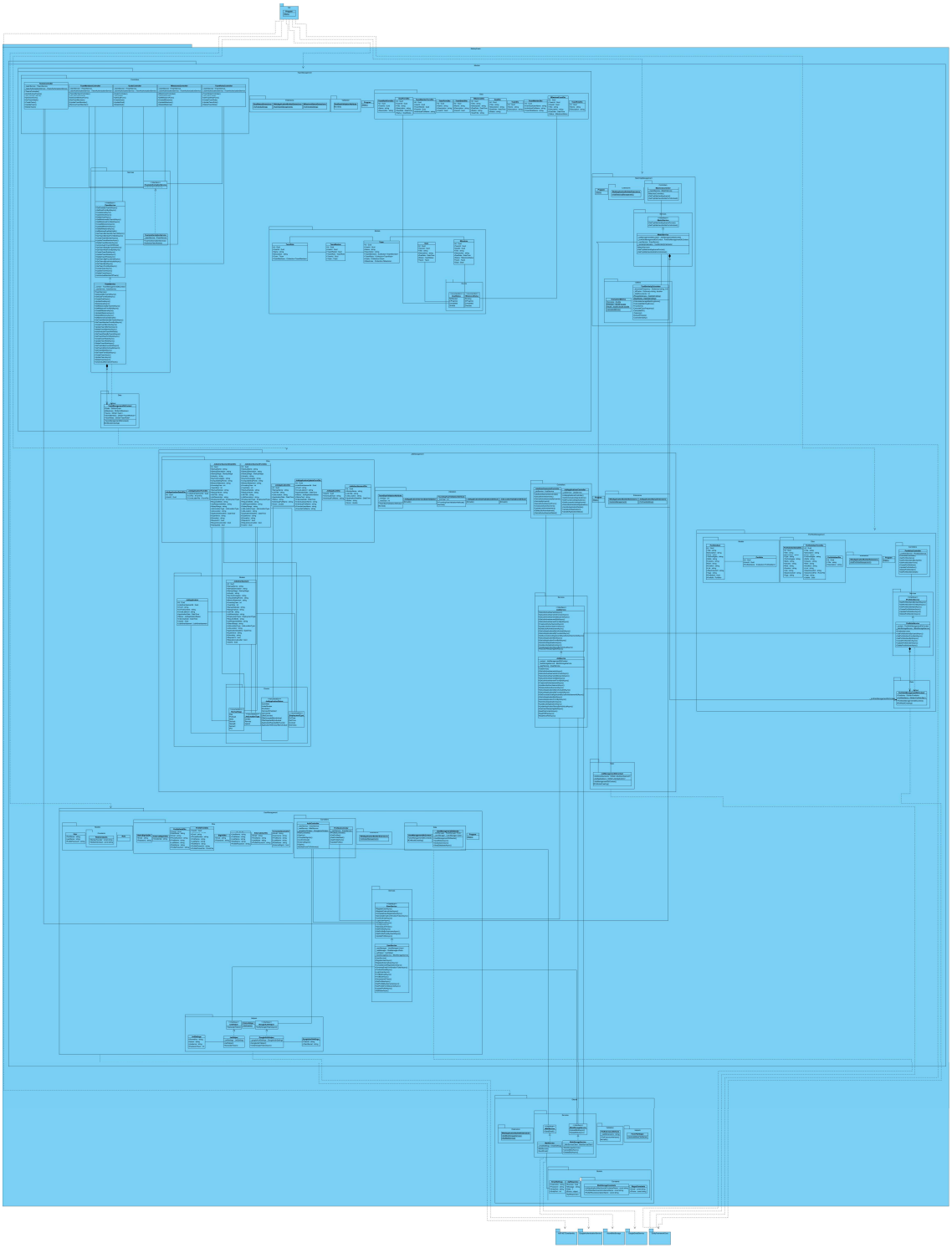
The root directory for the backend tests is located at / backend/ StartupTeam/ Tests/ StartupTeam.Tests/ UnitTests, while the root directory for the frontend tests can be found at / frontend/ startupteam/ src/ tests.

Requirement ID	Use Case ID	Backend Test Folder Reference	Frontend Test Folder Reference	Test Status
1.1	01	/StartupTeam.Module.UserManagement	/Auth	Passed
1.2	01	/StartupTeam.Module.UserManagement	/Auth	Passed
1.3	01	/StartupTeam.Module.UserManagement	/Auth	Passed
1.4	01	/StartupTeam.Module.UserManagement	/Auth	Passed
1.5	04	/StartupTeam.Module.UserManagement	/Users	Passed
1.6	05	/StartupTeam.Module.UserManagement	/Users	Passed
1.7	06, 08, 11	Covered Elsewhere	Covered Elsewhere	N/A
1.8	07, 09, 12	Covered Elsewhere	Covered Elsewhere	N/A
2.1	06	/StartupTeam.Module.JobManagement	/Jobs	Passed
2.2	06	/StartupTeam.Module.JobManagement	/Jobs	Passed
2.3	05, 06	Covered Elsewhere	Covered Elsewhere	N/A
2.4	07	/StartupTeam.Module.JobManagement	/JobApplications	Passed
2.5	08	/StartupTeam.Module.JobManagement	/JobApplications	Passed
3.1	09	/StartupTeam.Module.PortfolioManagement	/Portfolios	Passed
3.2	09	/StartupTeam.Module.PortfolioManagement	/Portfolios	Passed
3.3	05, 09	Covered Elsewhere	Covered Elsewhere	N/A
4.1	11	/StartupTeam.Module.TeamManagement	/Teams	Passed
4.2	12	/StartupTeam.Module.TeamManagement	/Teams	Passed
4.3	13	/StartupTeam.Module.TeamManagement	/Teams	Passed
4.4	12	/StartupTeam.Module.TeamManagement	/Teams	Passed
4.5	14, 15	/StartupTeam.Module.TeamManagement	/Teams	Passed
5.1	10	/StartupTeam.Module.MatchingManagement	/Matches	Passed

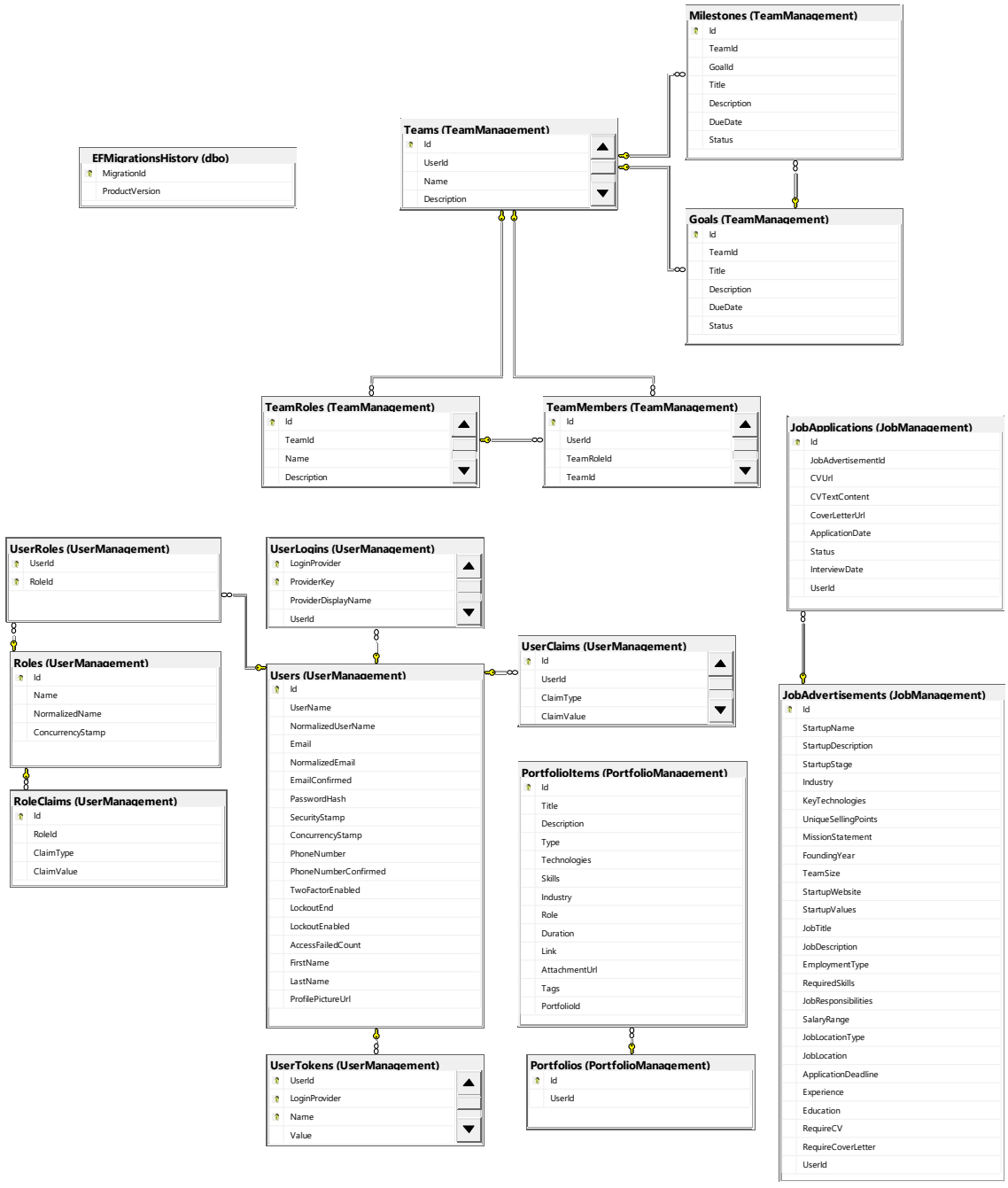
5.2	10	/StartupTeam.Module.MatchingManagement	/Matches	Passed
5.3	10	/StartupTeam.Module.MatchingManagement	/Matches	Passed
6.1	16	Dropped	Dropped	N/A
6.2	16	Dropped	Dropped	N/A

Appendix E: Use Case Diagram

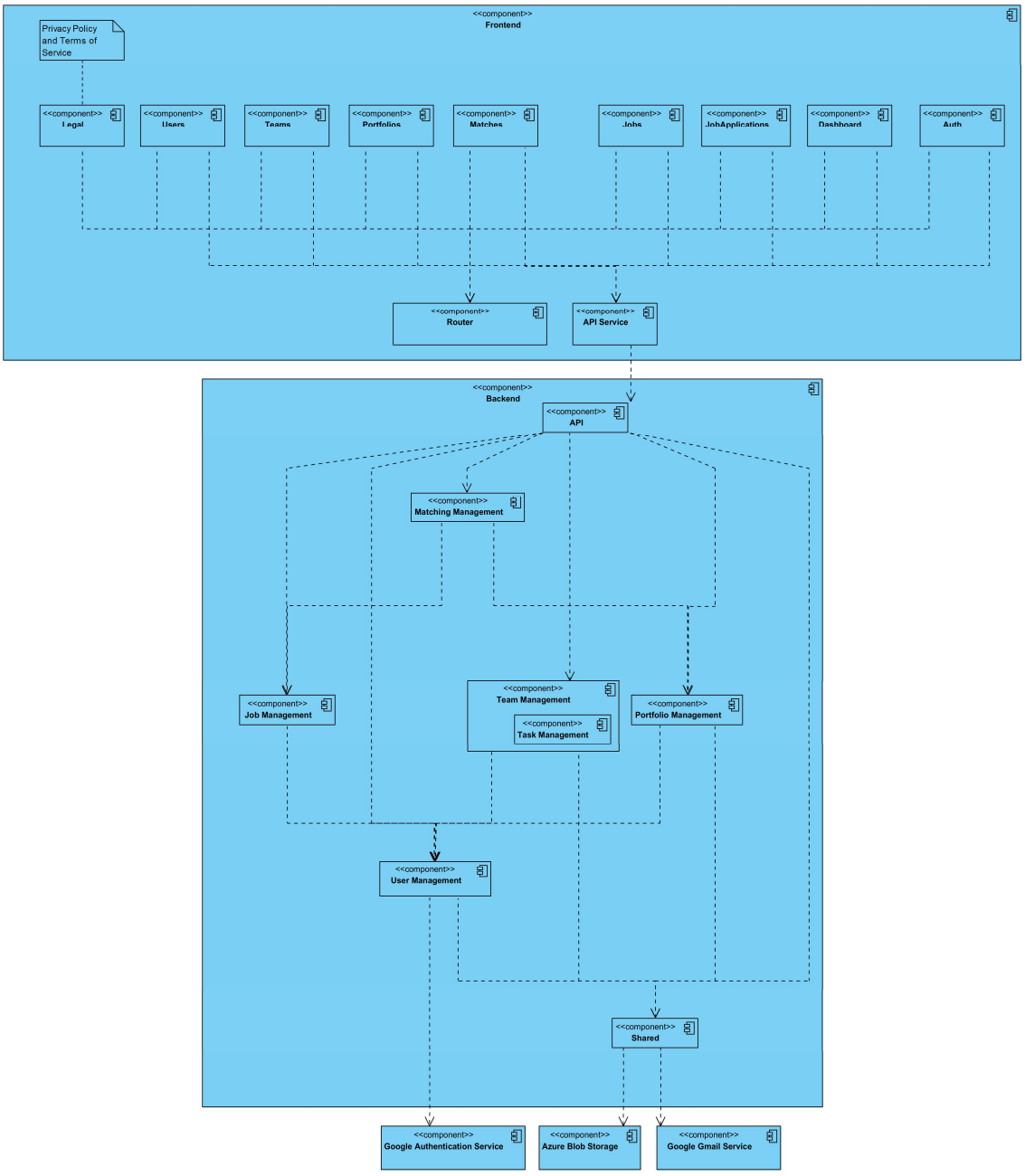
Appendix F: Class Diagram



Appendix G: Data Model Diagram

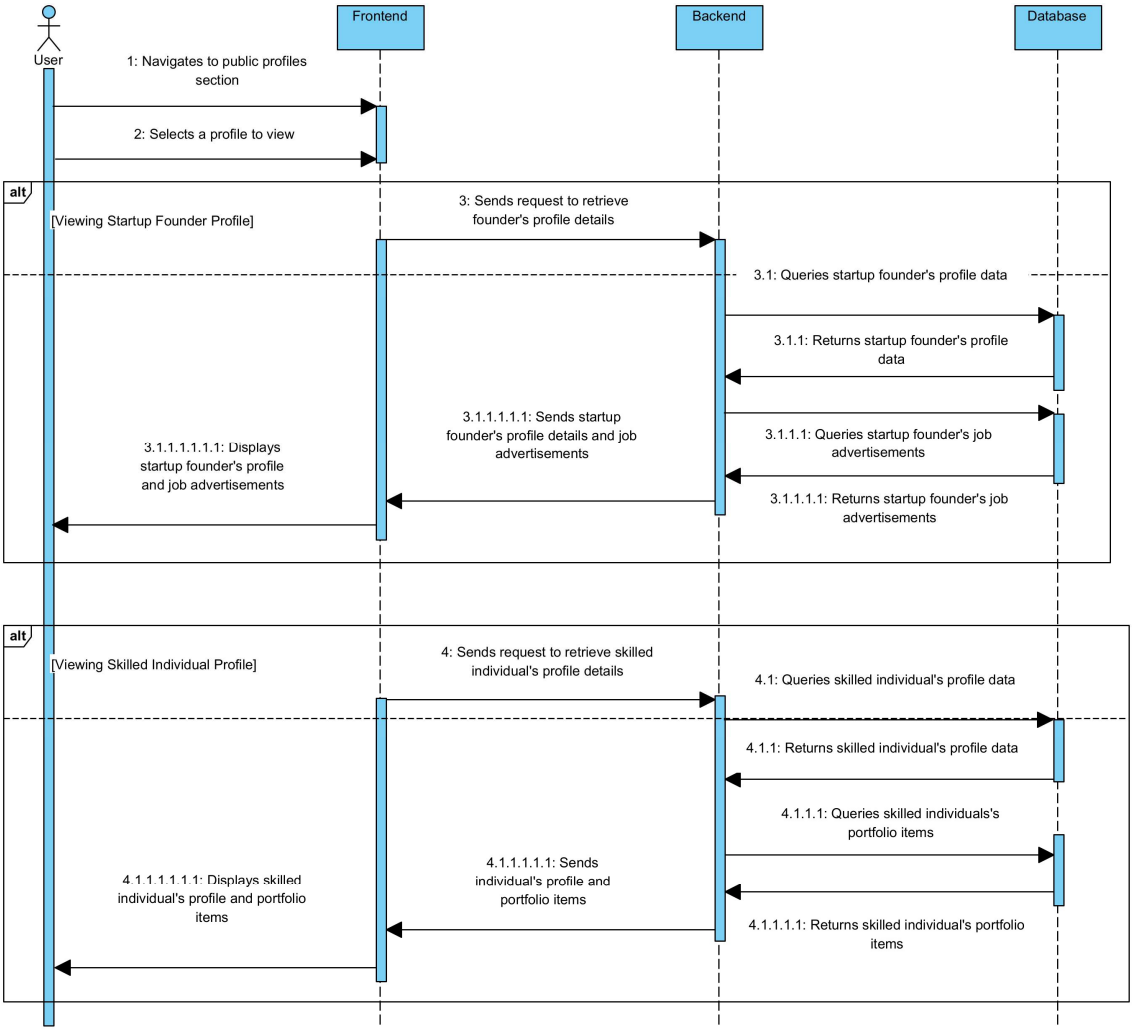


Appendix H: Component Diagram

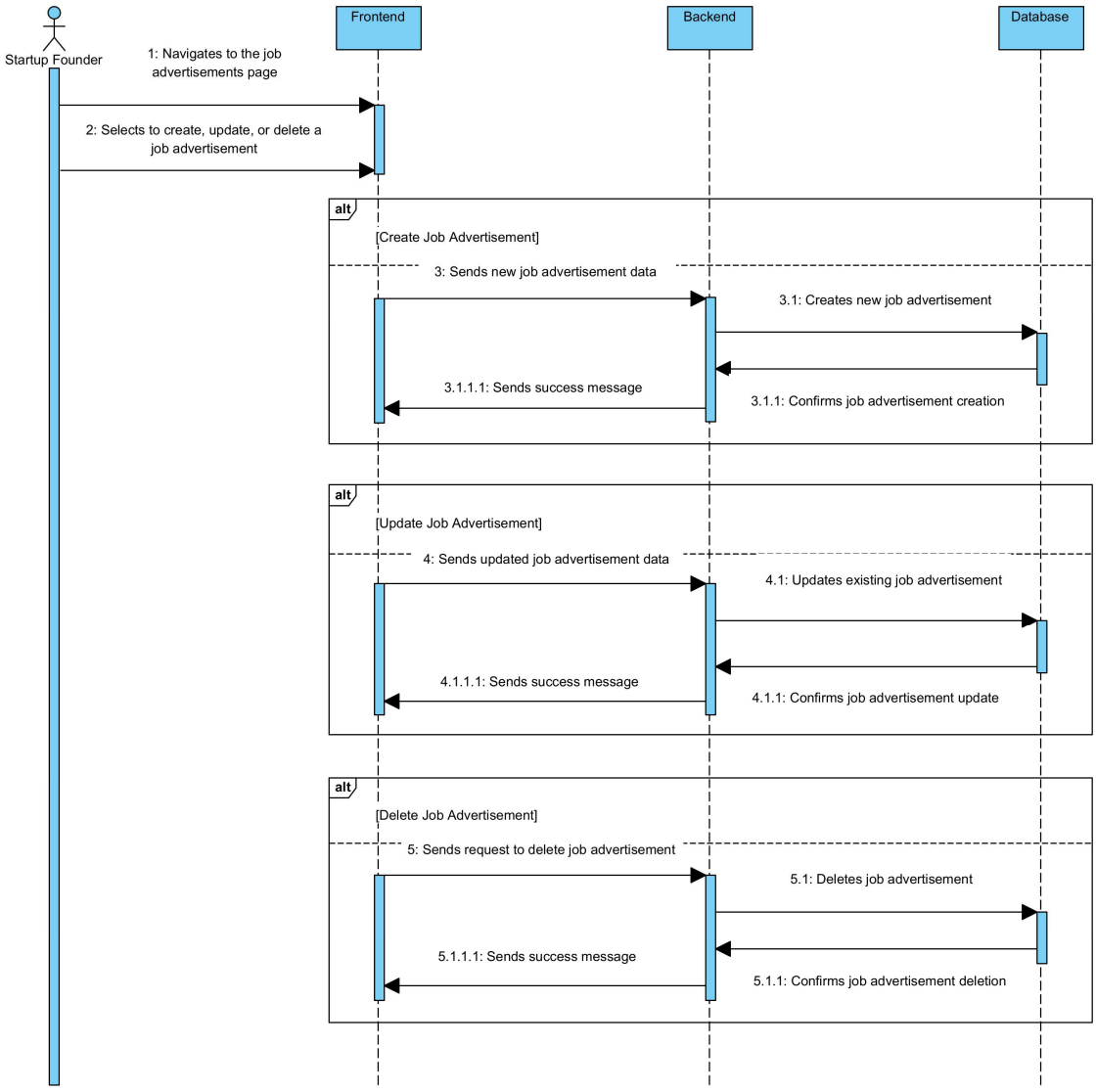


Appendix I: Sequence Diagrams

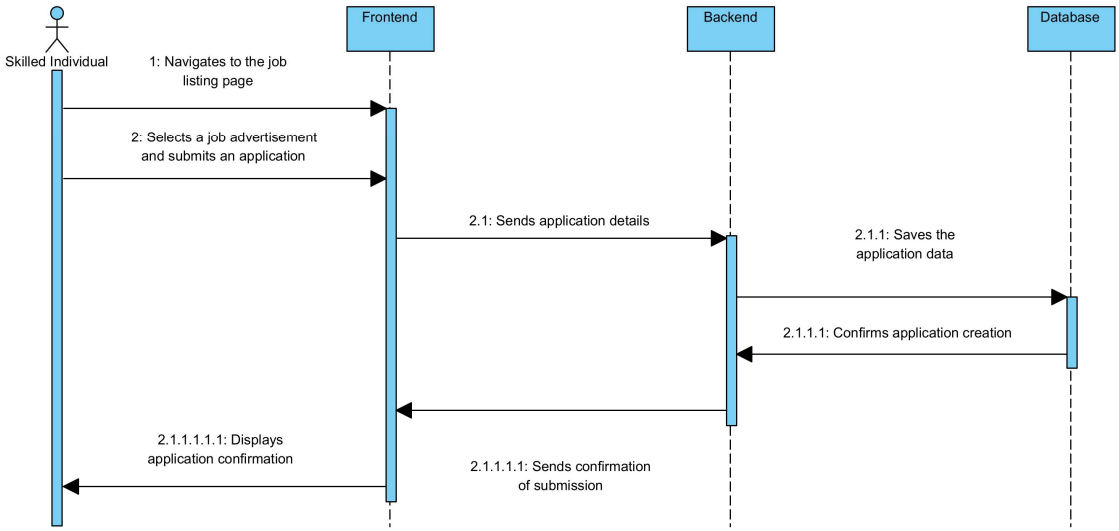
I.1 Sequence Diagram for Viewing User Public Profiles



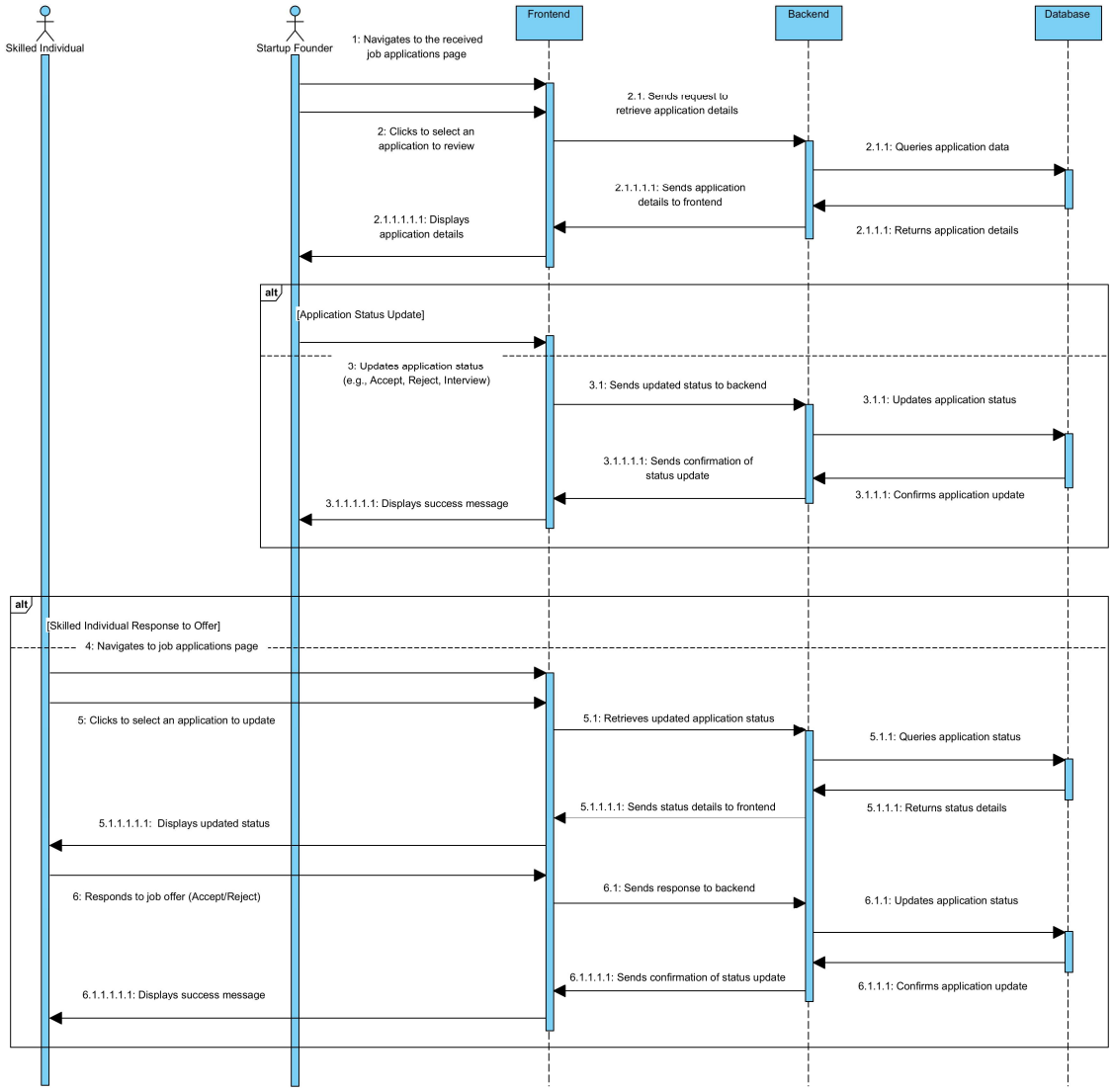
I.2 Sequence Diagram for Managing Job Advertisements



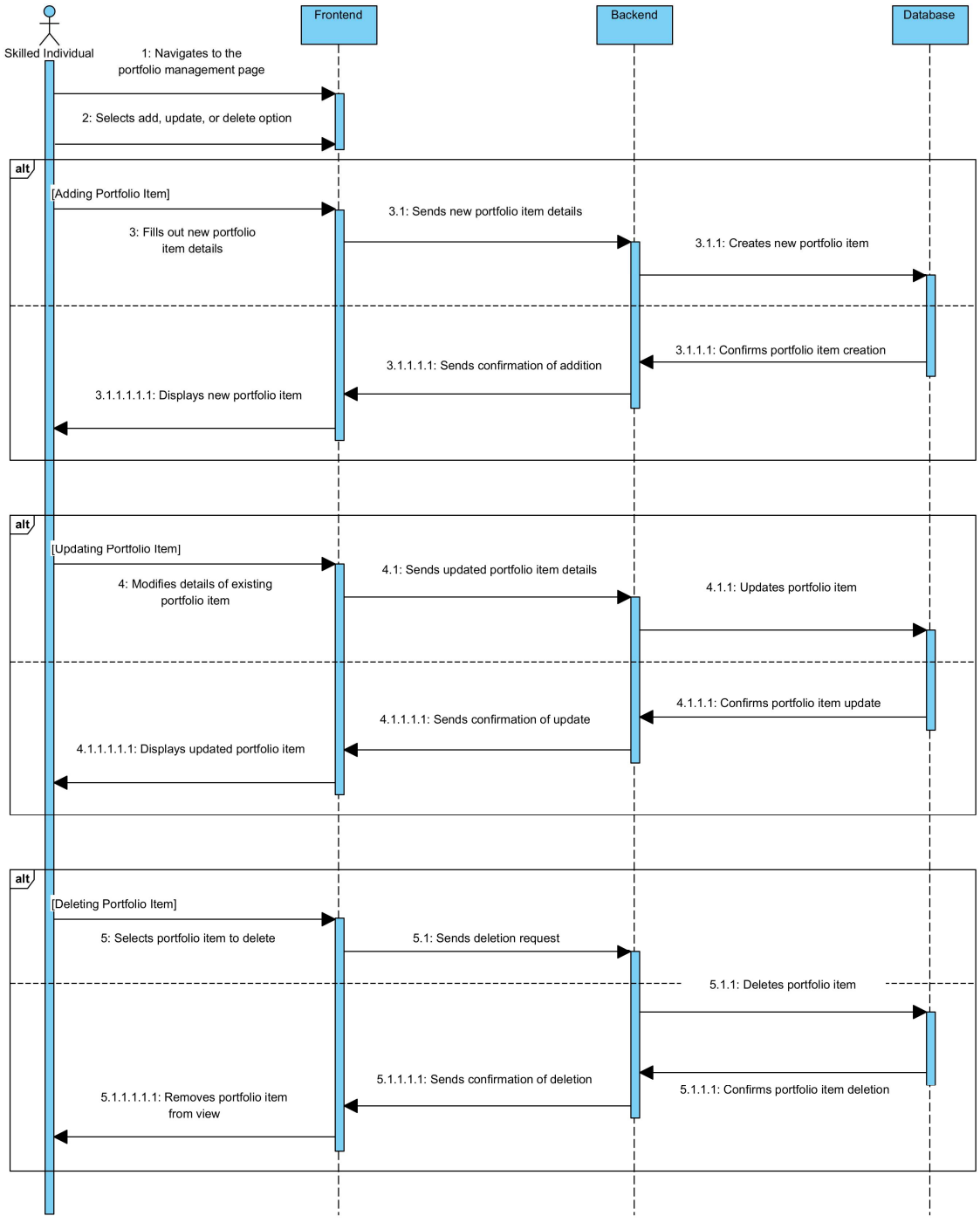
I.3 Sequence Diagram for Submitting Job Applications



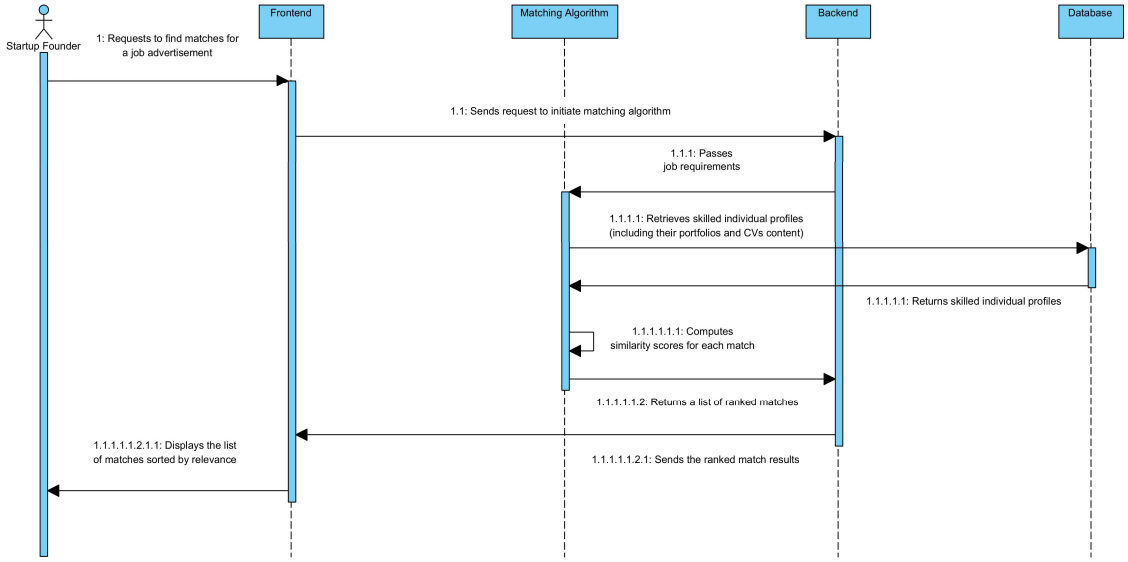
I.4 Sequence Diagram for Managing Job Applications



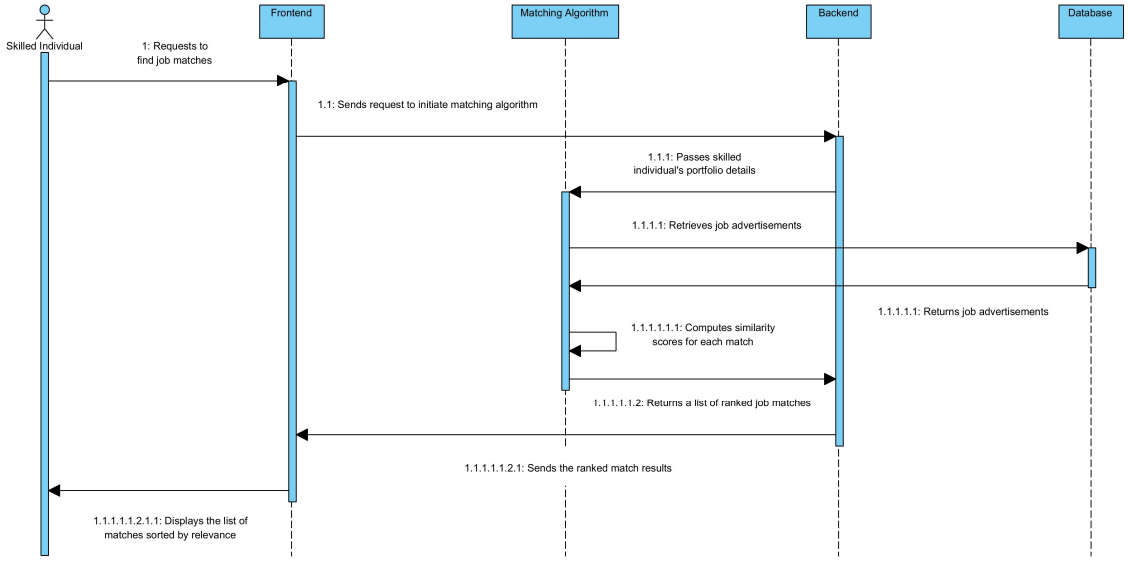
I.5 Sequence Diagram for Managing Portfolio Items



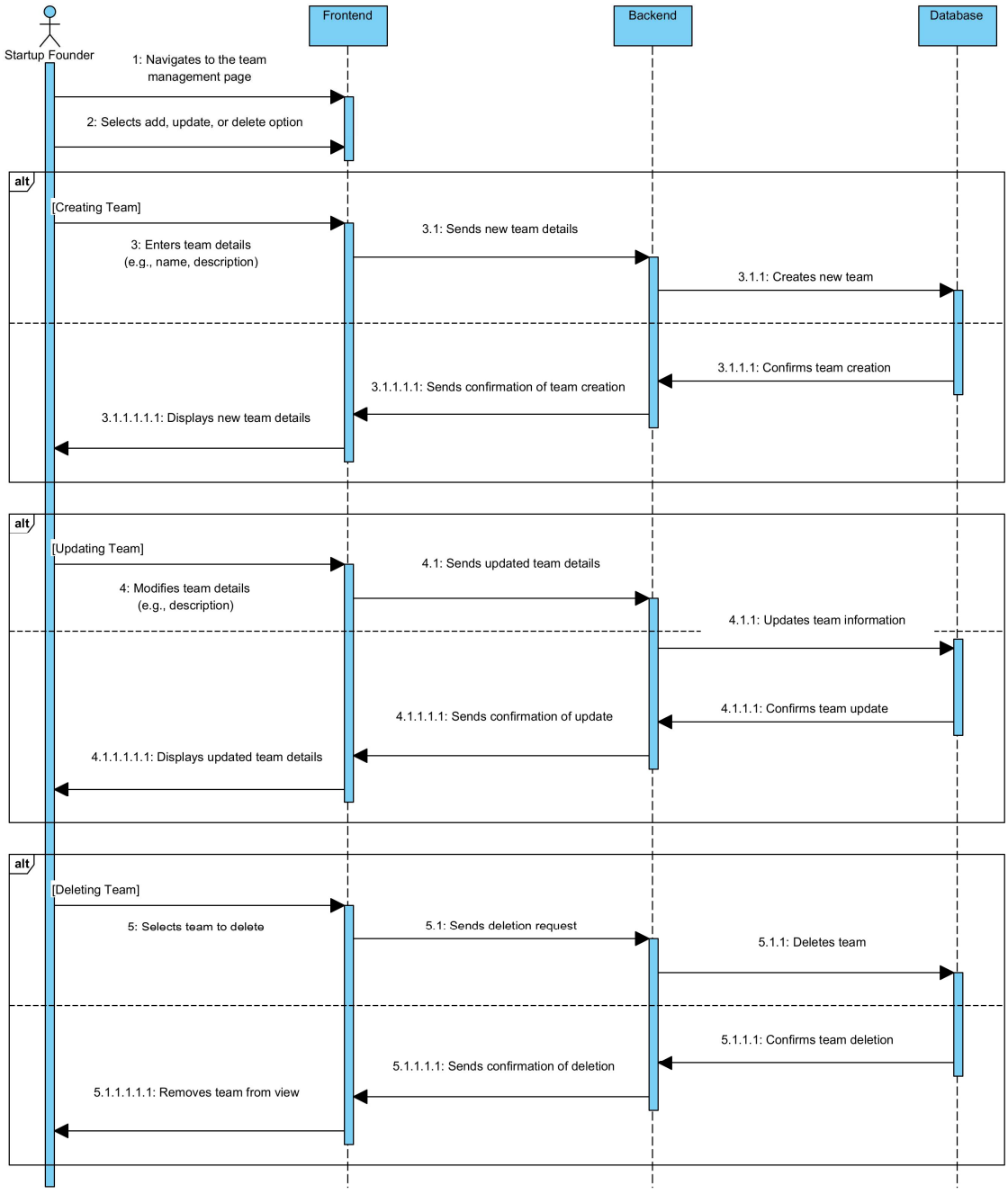
I.6 Sequence Diagram for Finding Matches Using Matching Algorithm (Startup Founder)



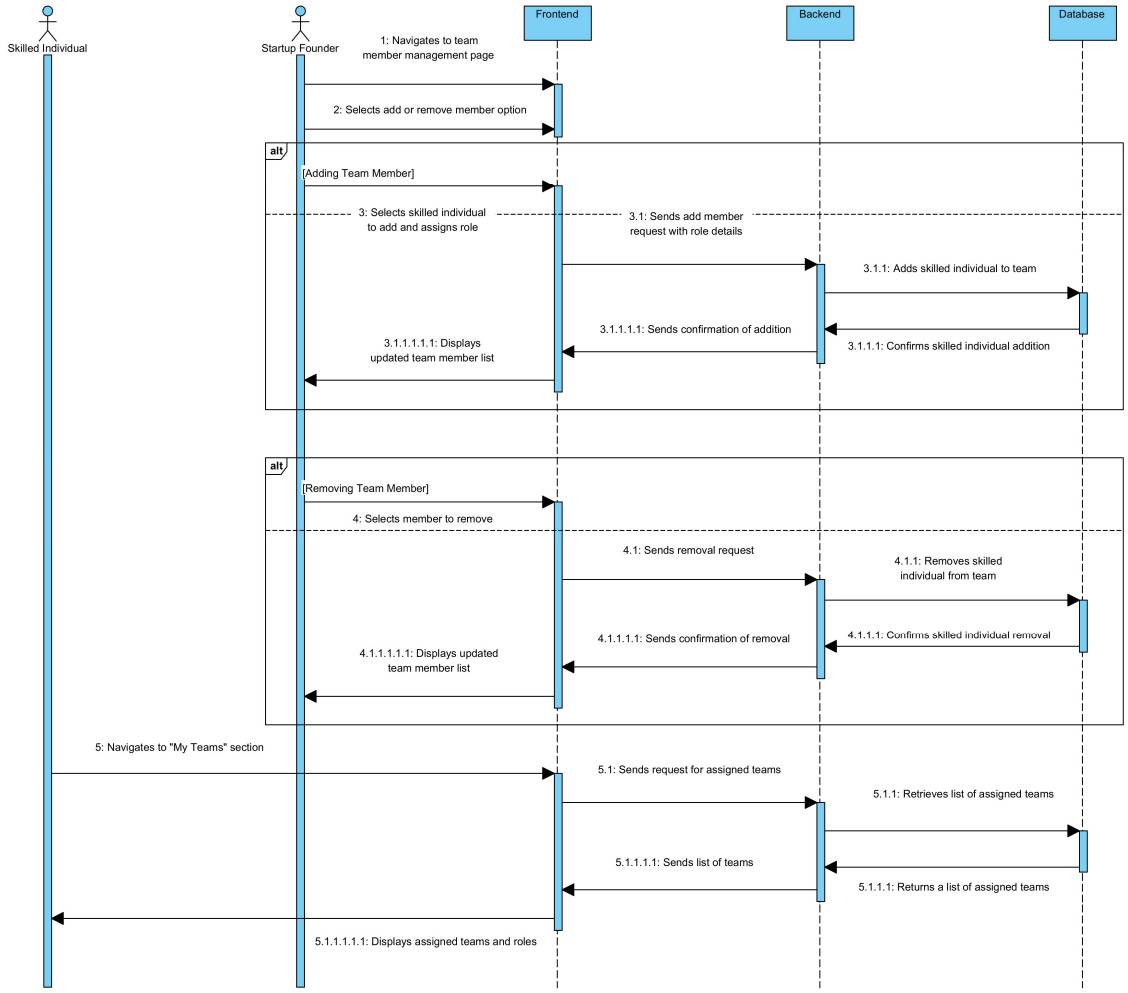
I.7 Sequence Diagram for Finding Matches Using Matching Algorithm (Skilled Individual)



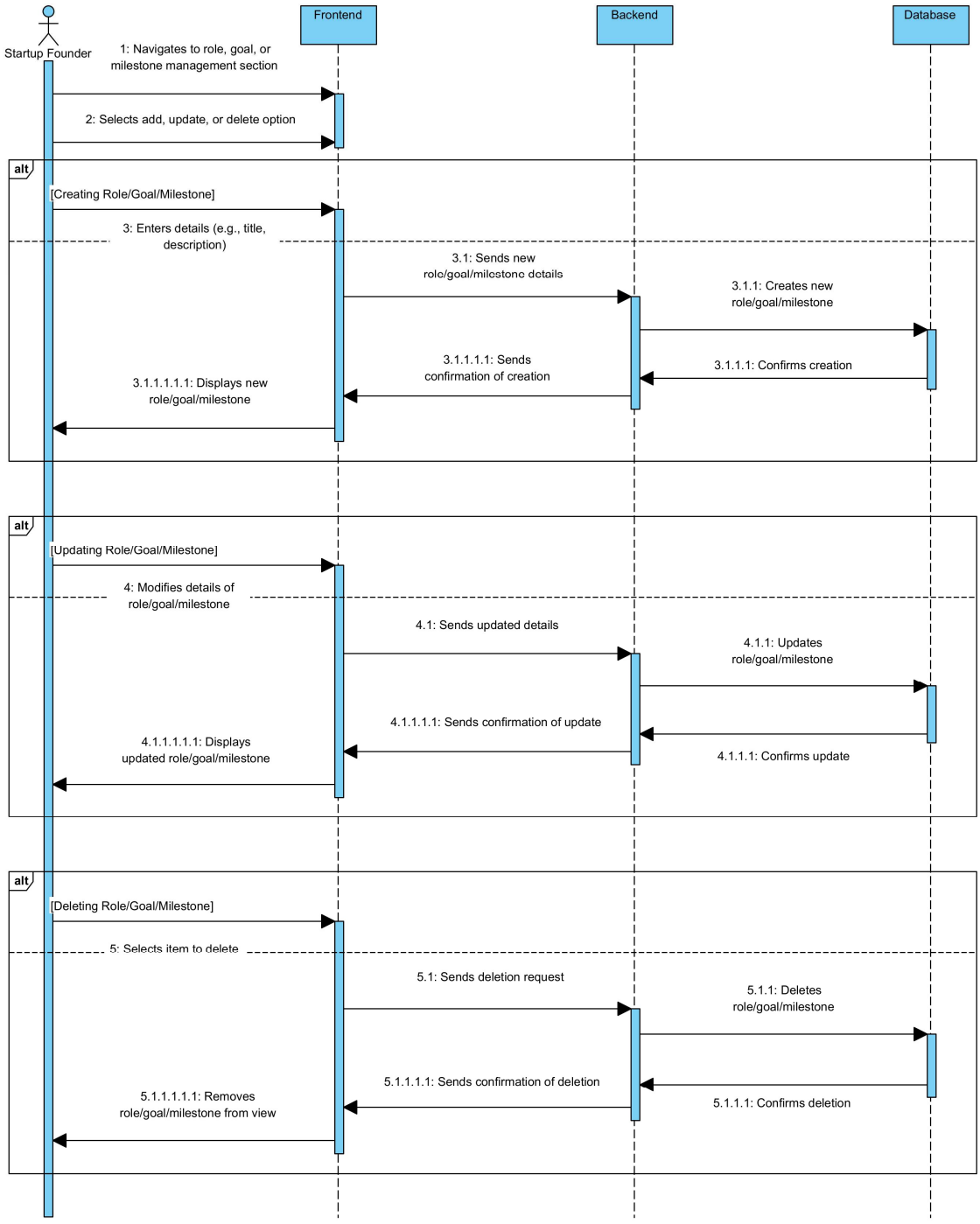
I.8 Sequence Diagram for Managing Startup Teams



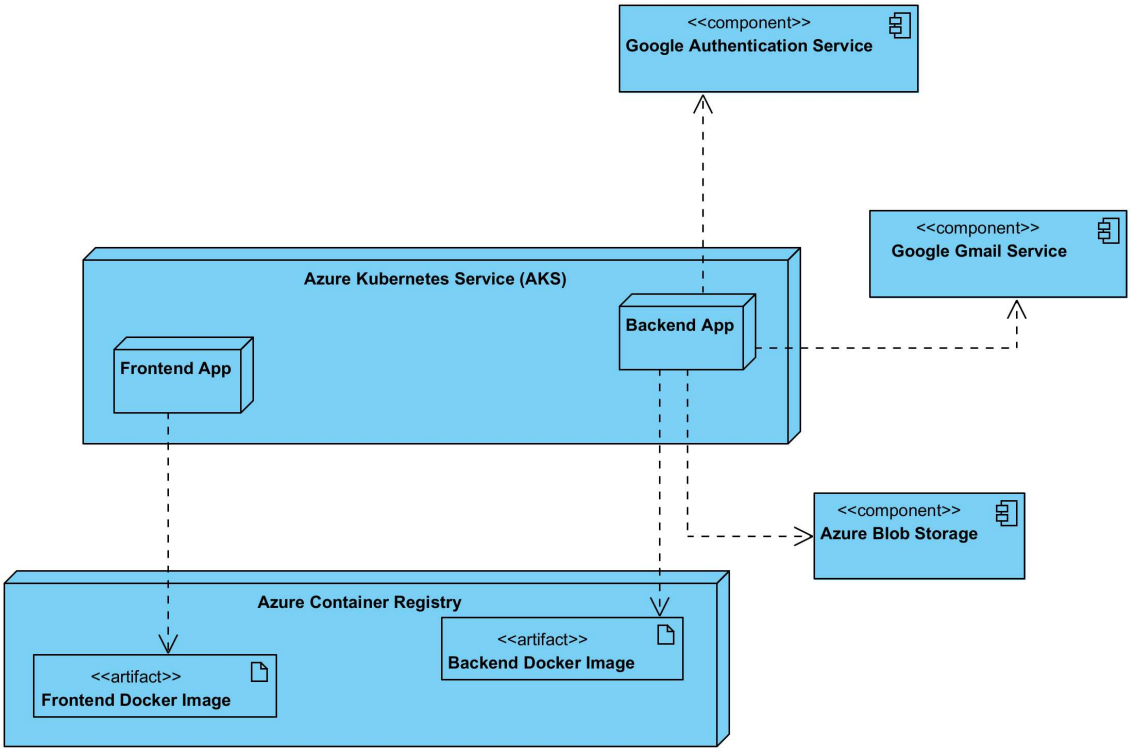
I.9 Sequence Diagram for Managing Team Members and Viewing Assigned Teams



I.10 Sequence Diagram for Managing Team Roles, Goals, and Milestones

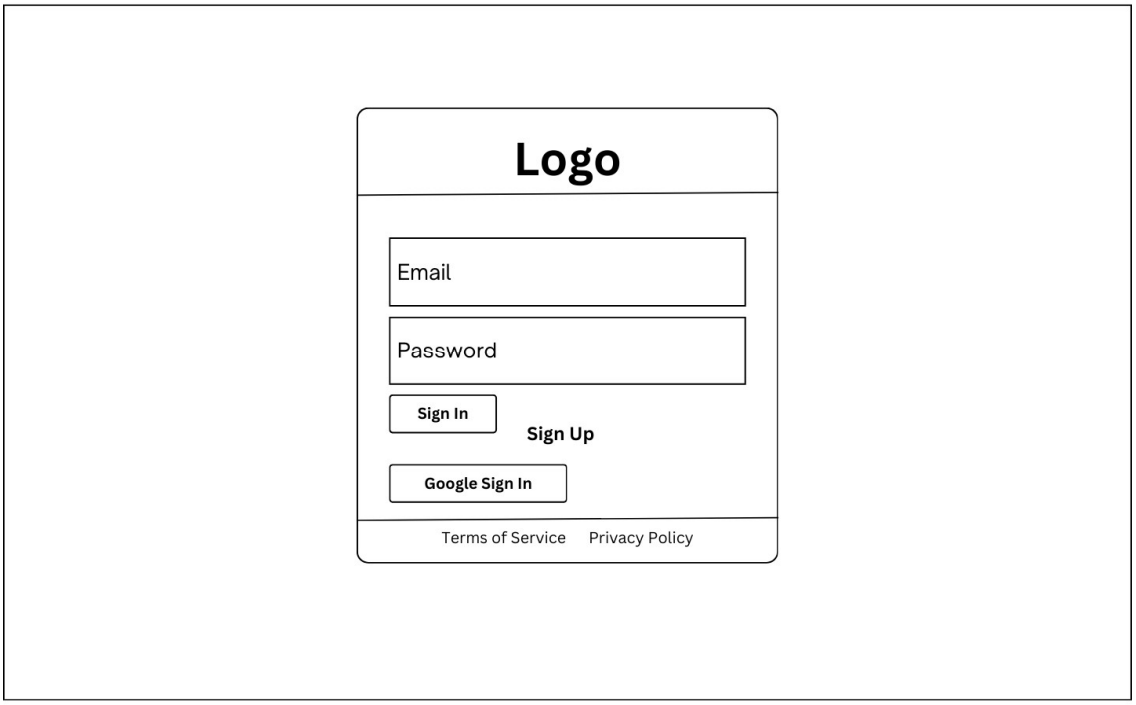


Appendix J: Deployment Diagram

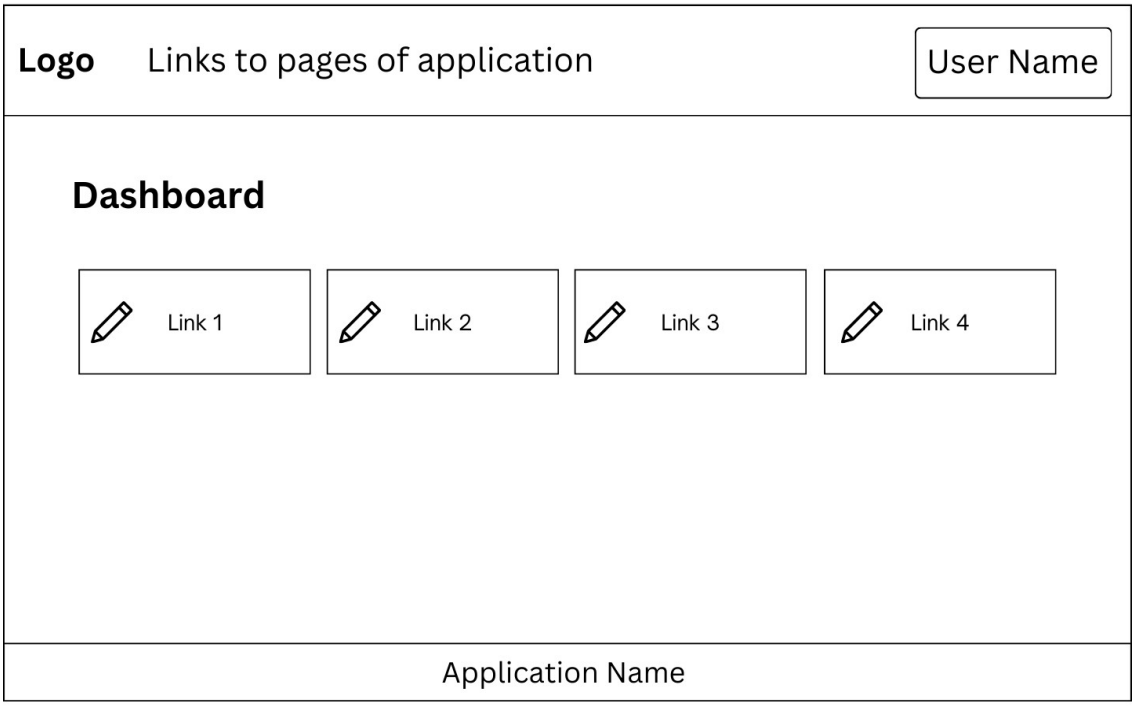


Appendix K: Wireframes

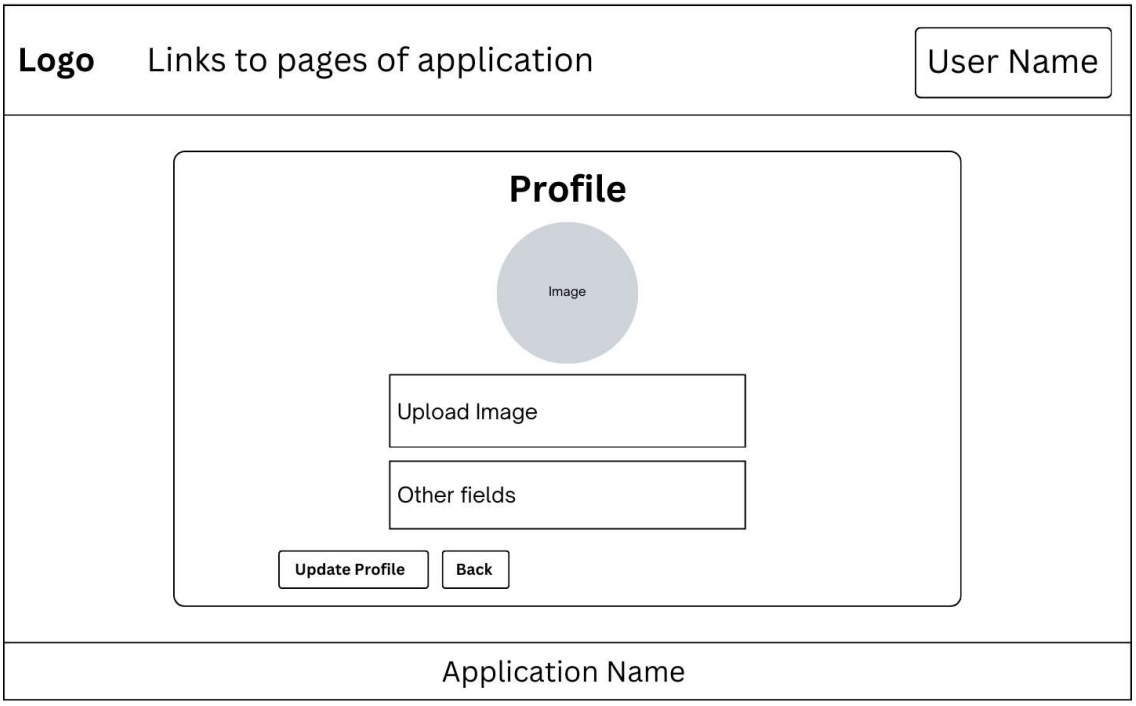
K.1 SignIn Page



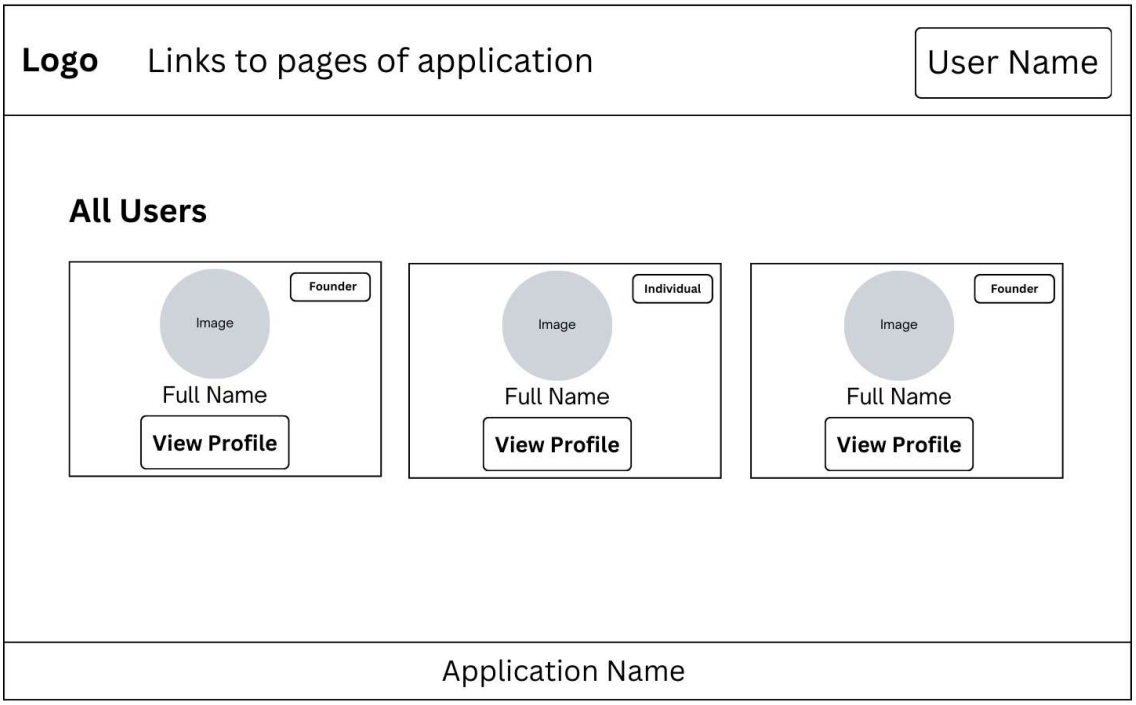
K.2 Panel Overview Page



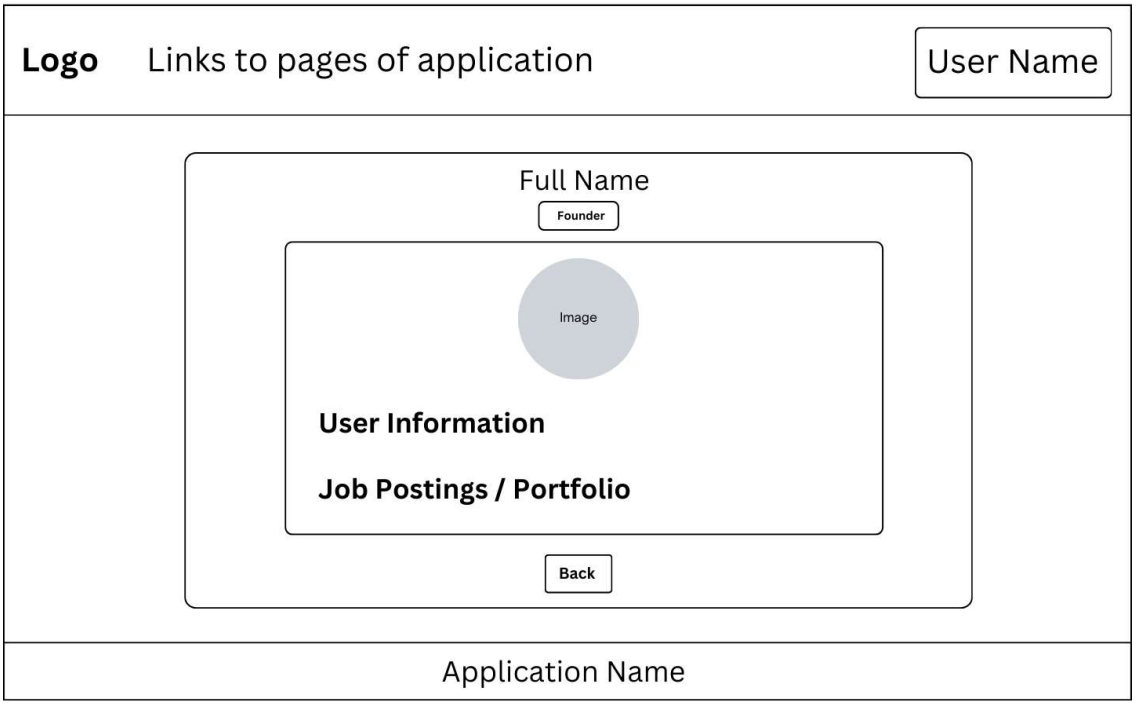
K.3 Profile Update Form



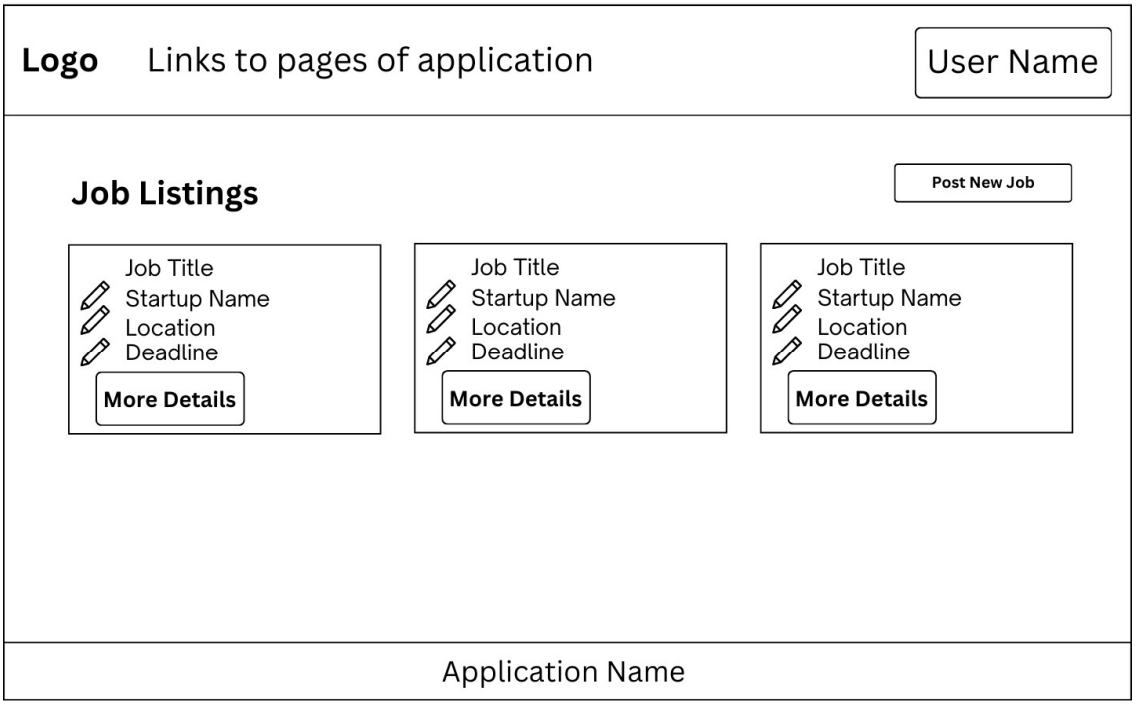
K.4 All Users Page



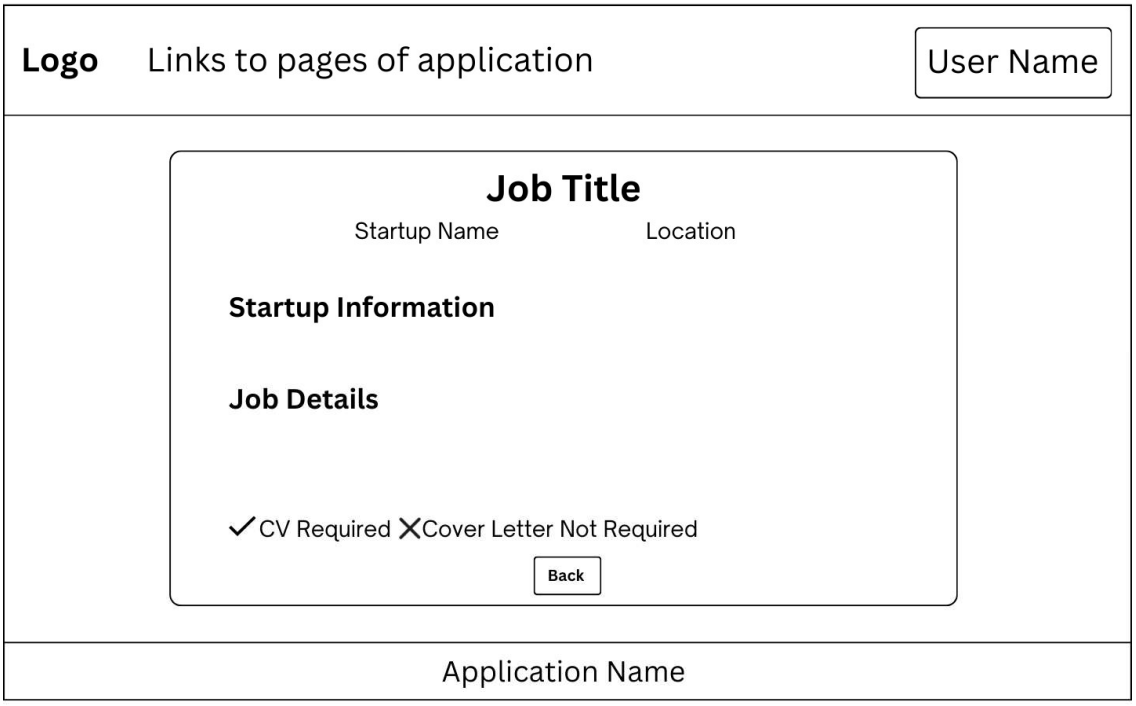
K.5 User Public Page



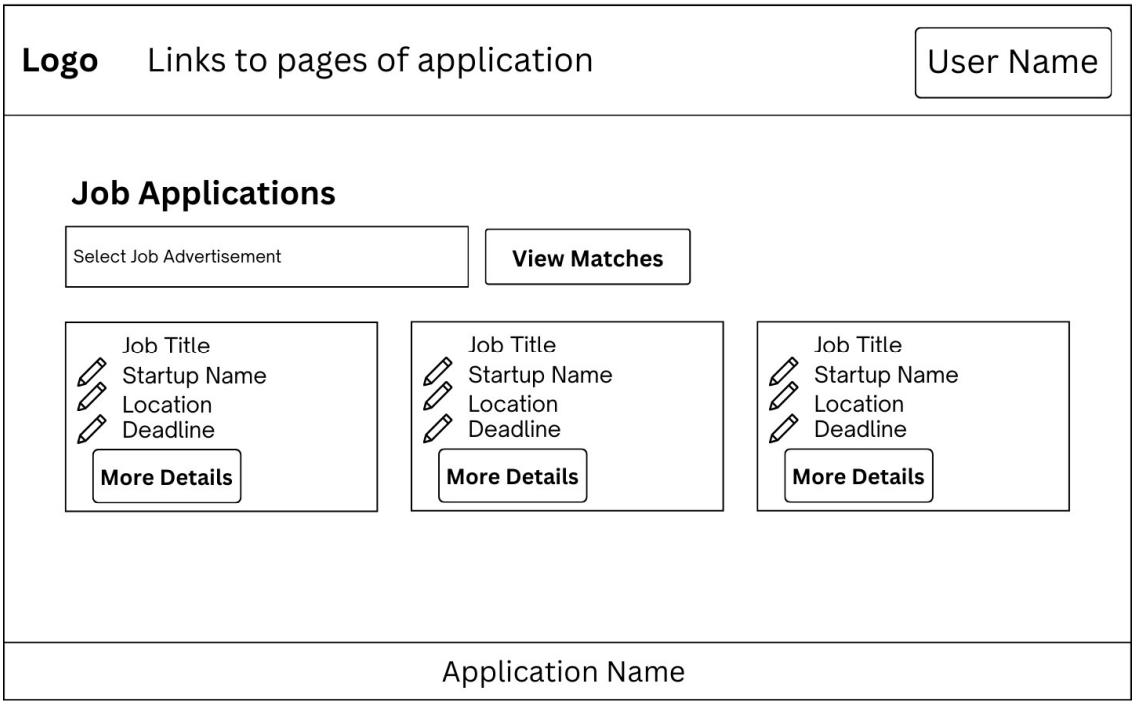
K.6 Job Listings Page



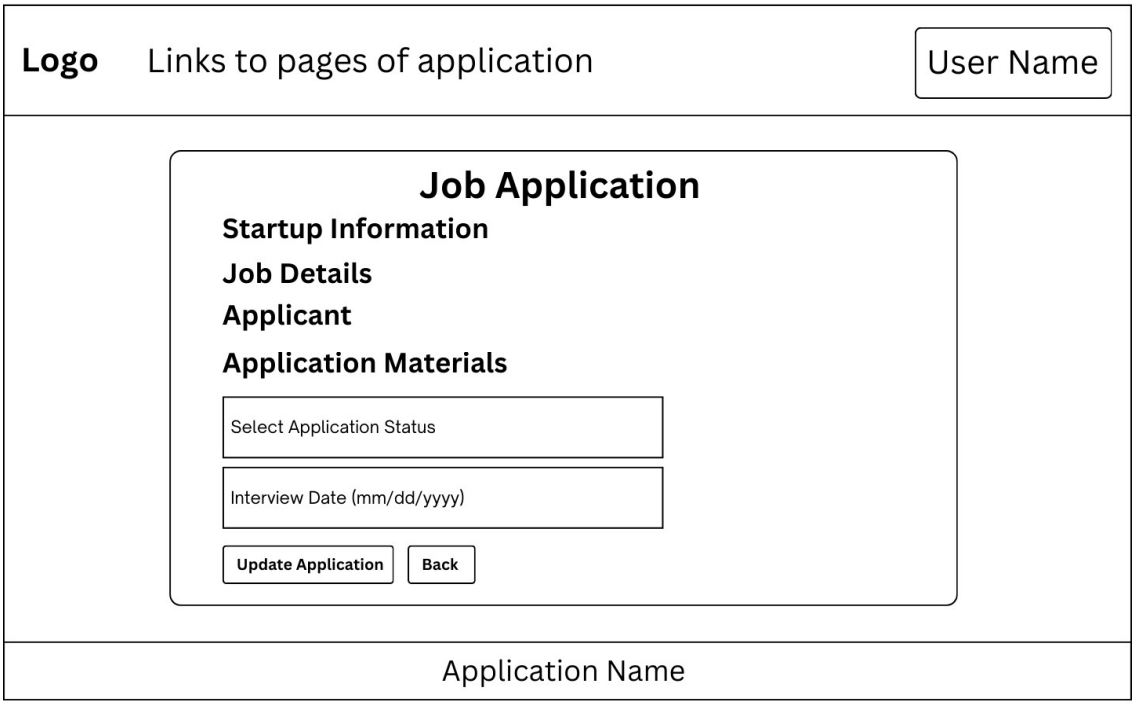
K.7 Job Advertisement Page



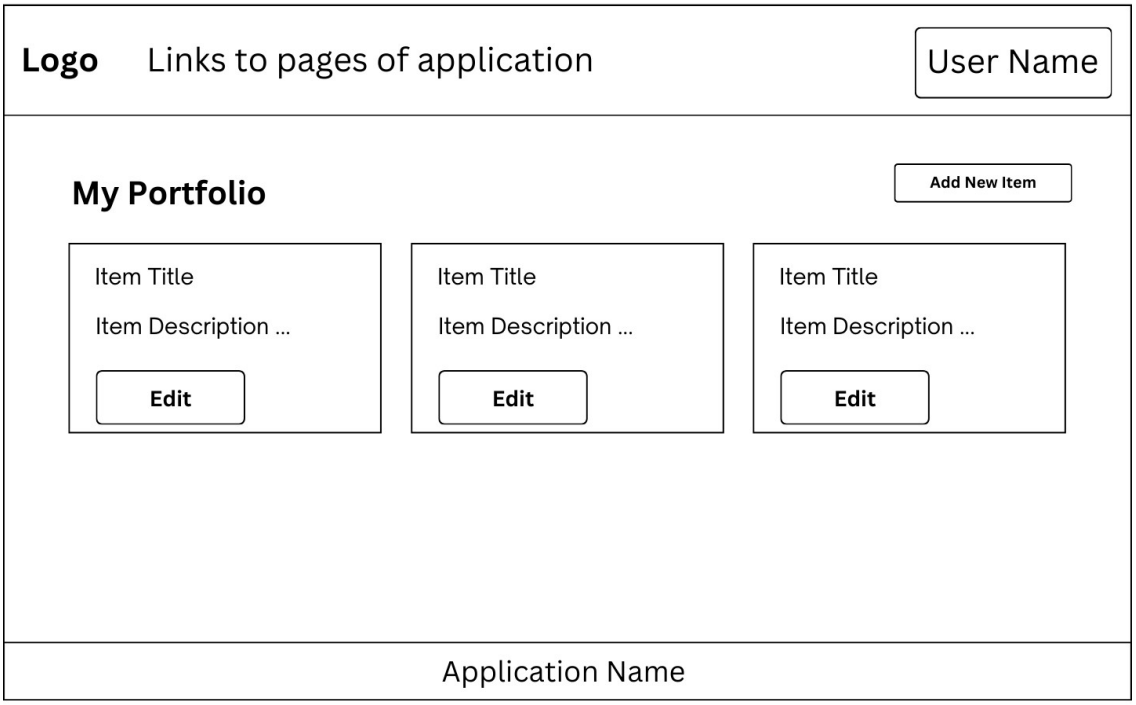
K.8 Job Applications Page



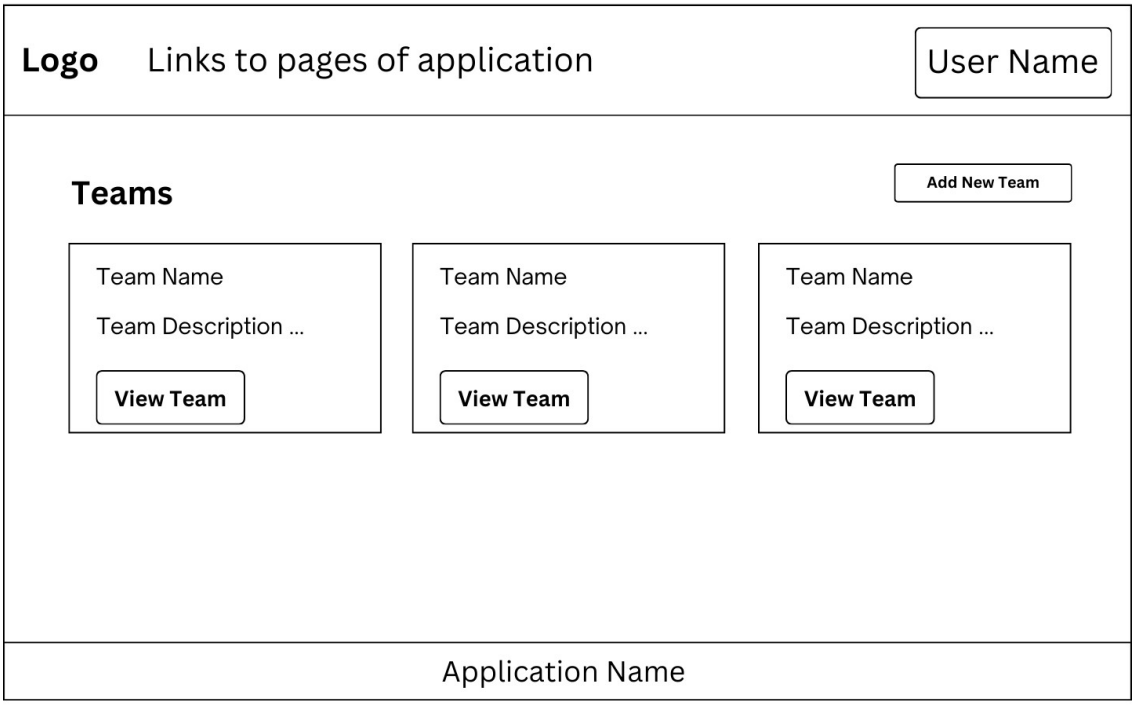
K.9 Job Application Page



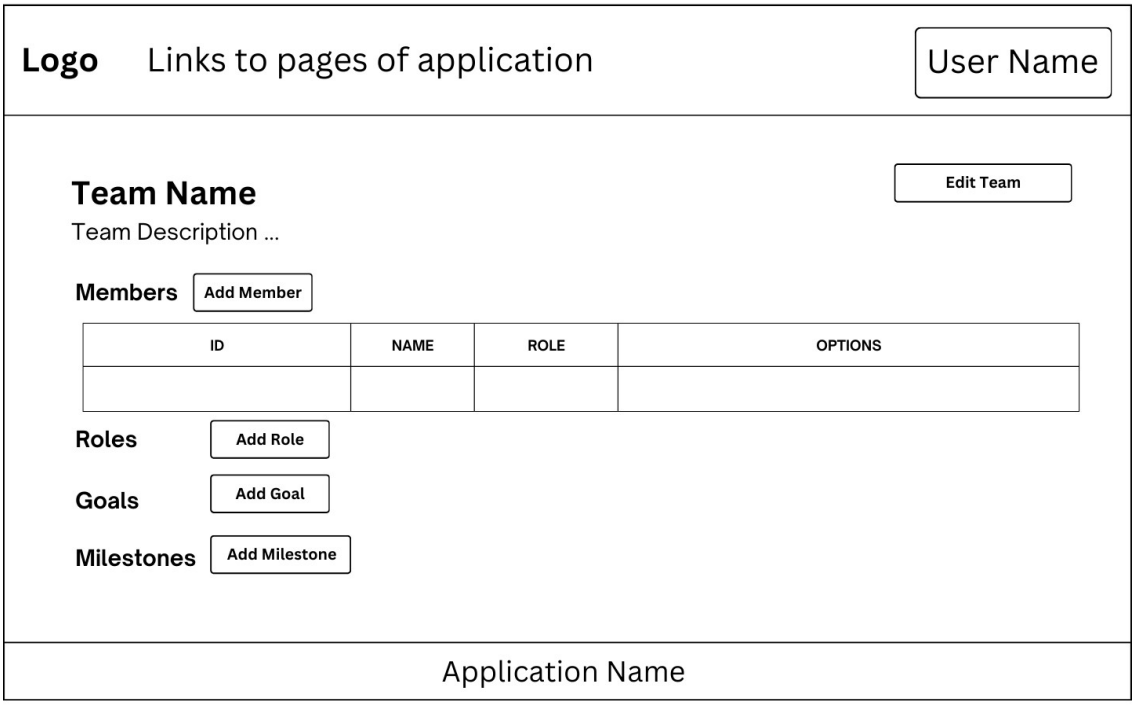
K.10 Portfolio Management Page



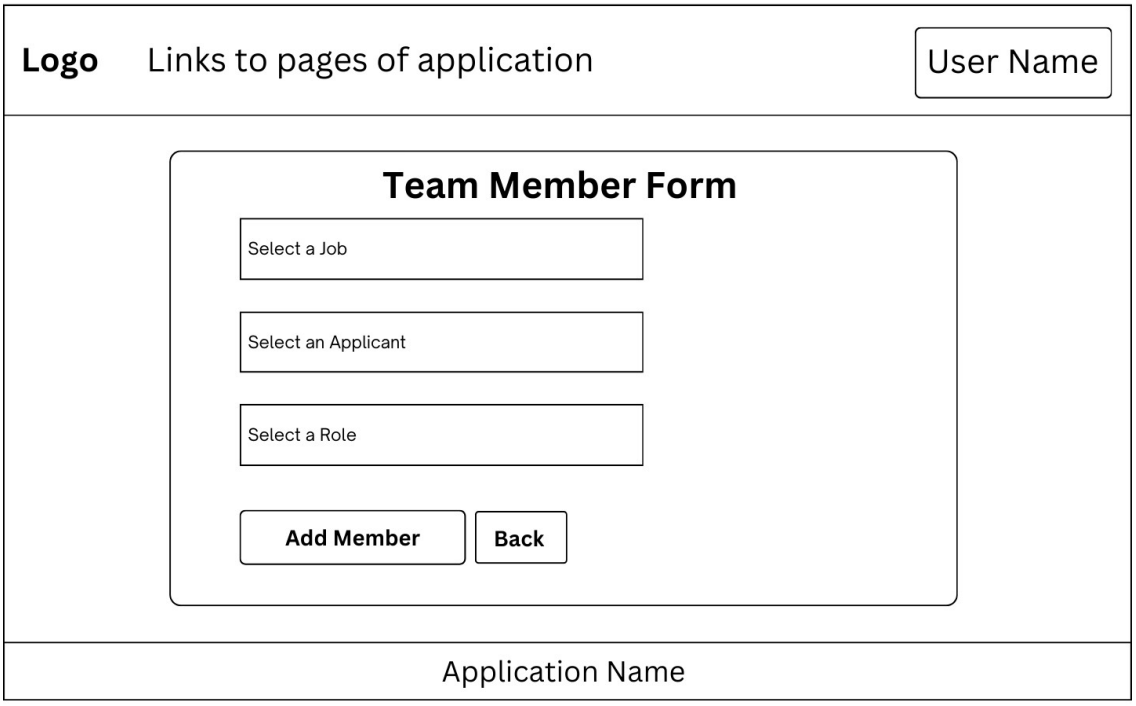
K.11 Teams Management Page



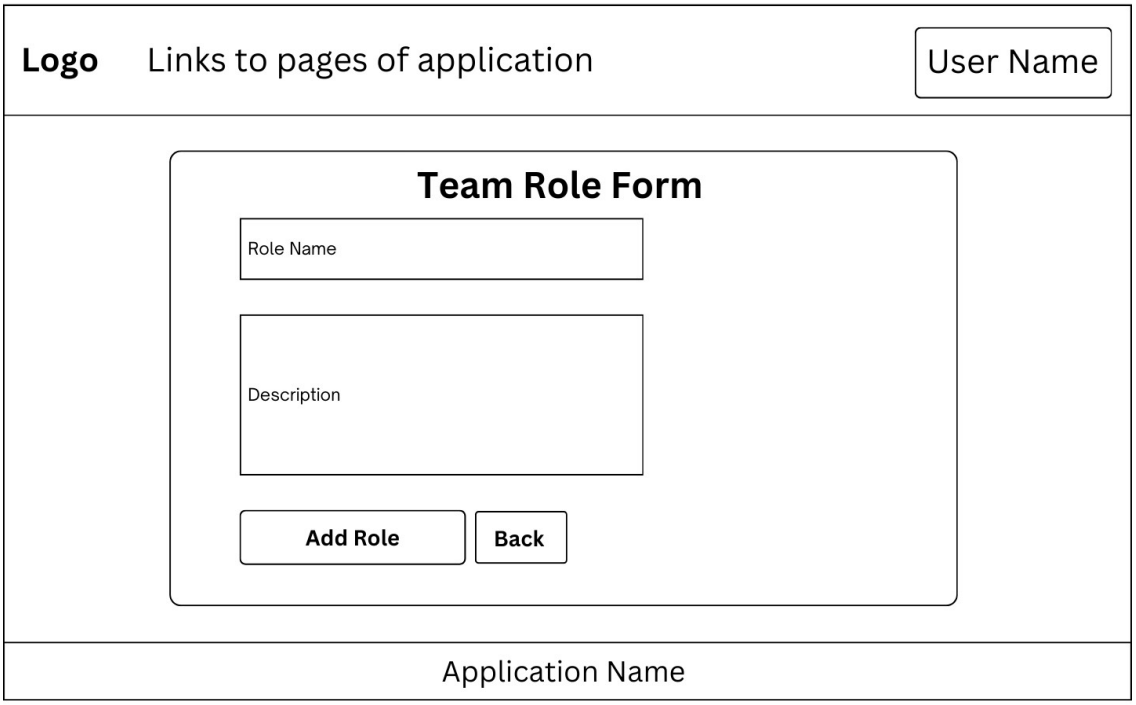
K.12 Team Overview Page



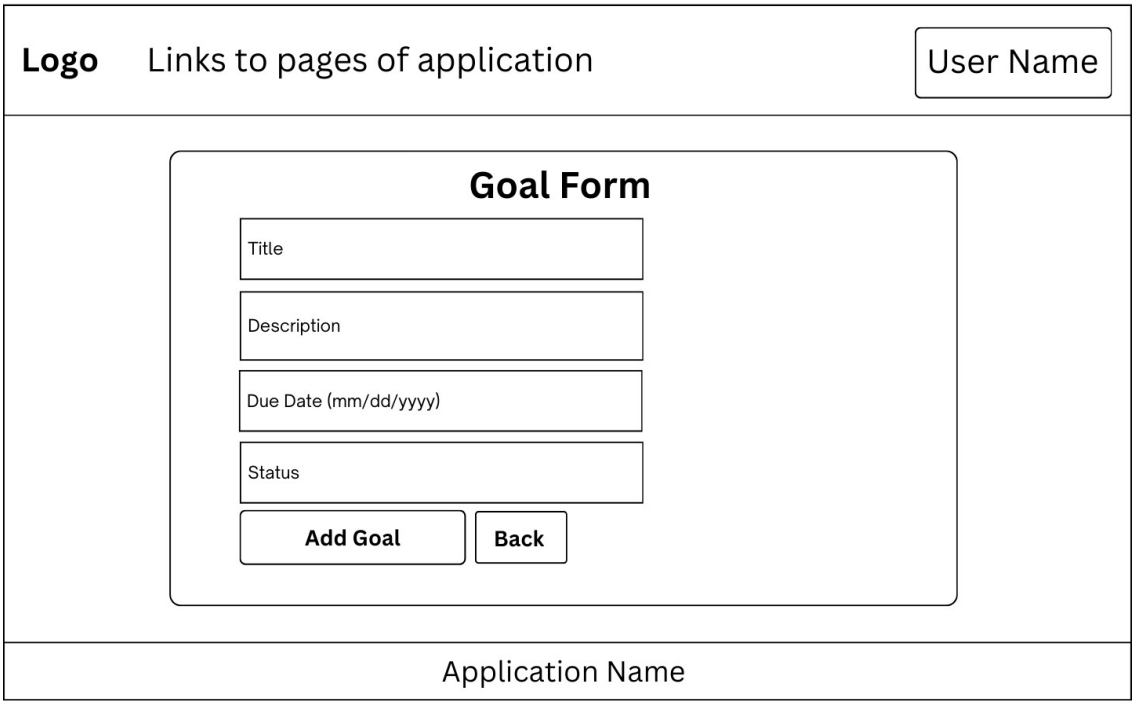
K.13 Team Member Addition Form



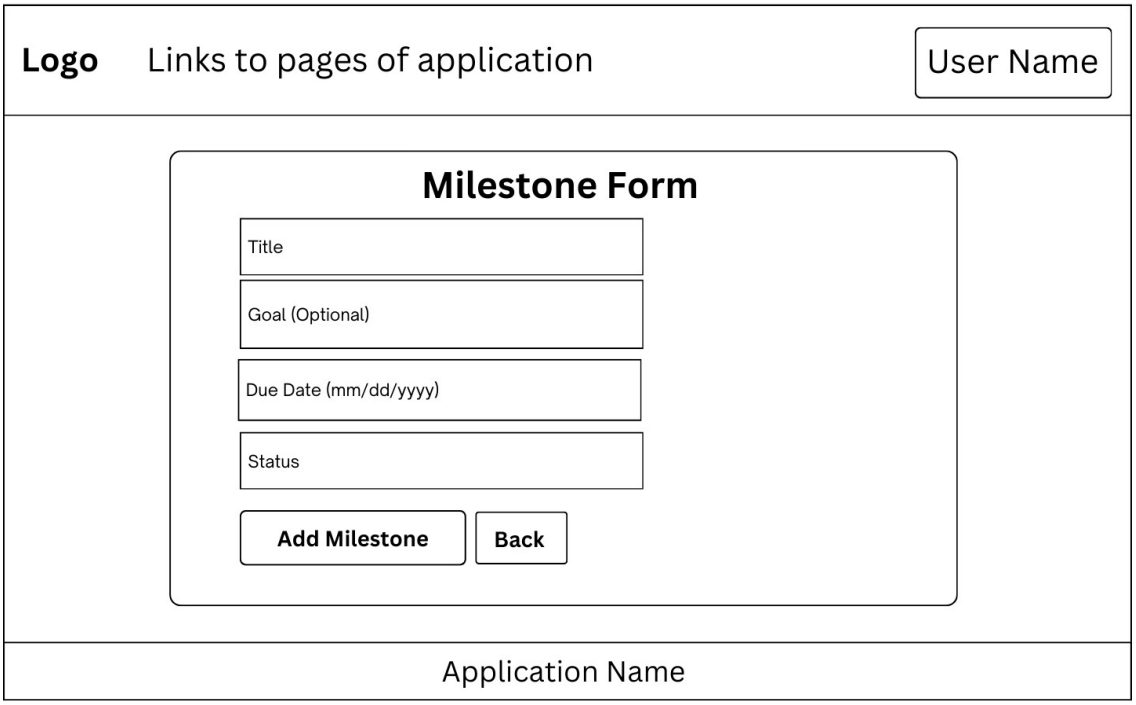
K.14 Team Role Creation Form



K.15 Goal Creation Form



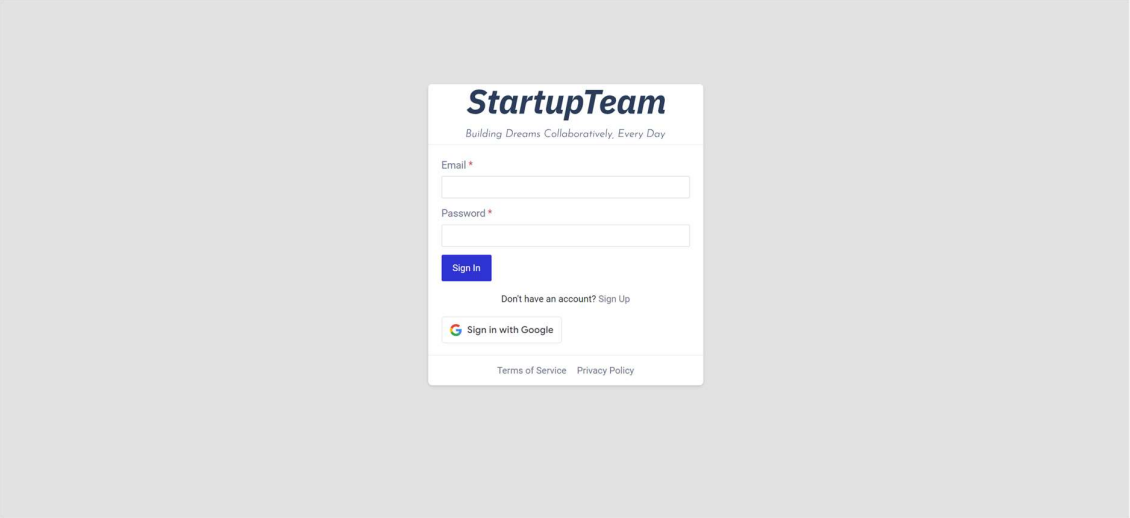
K.16 Milestone Creation Form



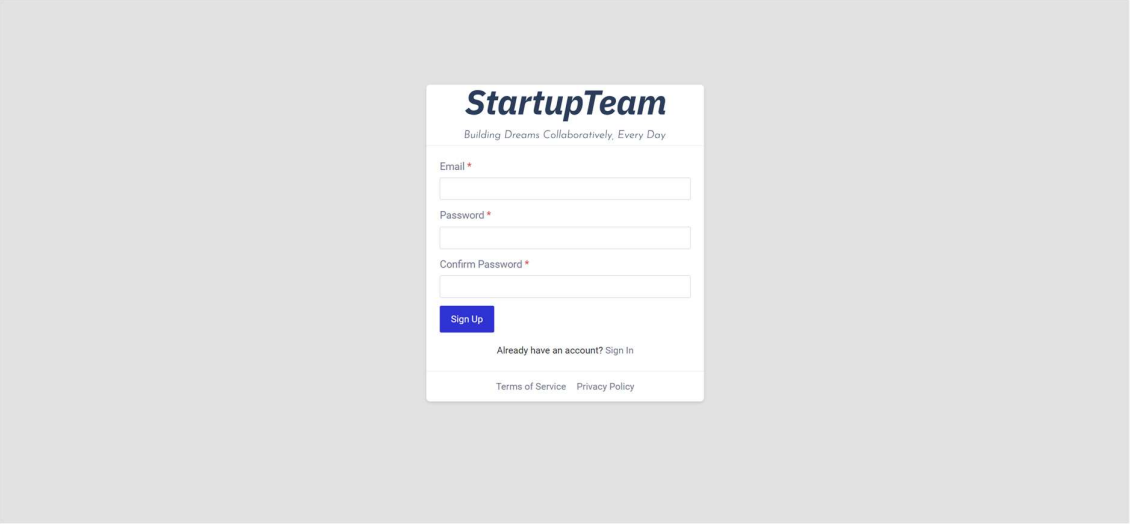
Appendix L: Platform Screenshots

L.1 SignIn and SignUp Pages

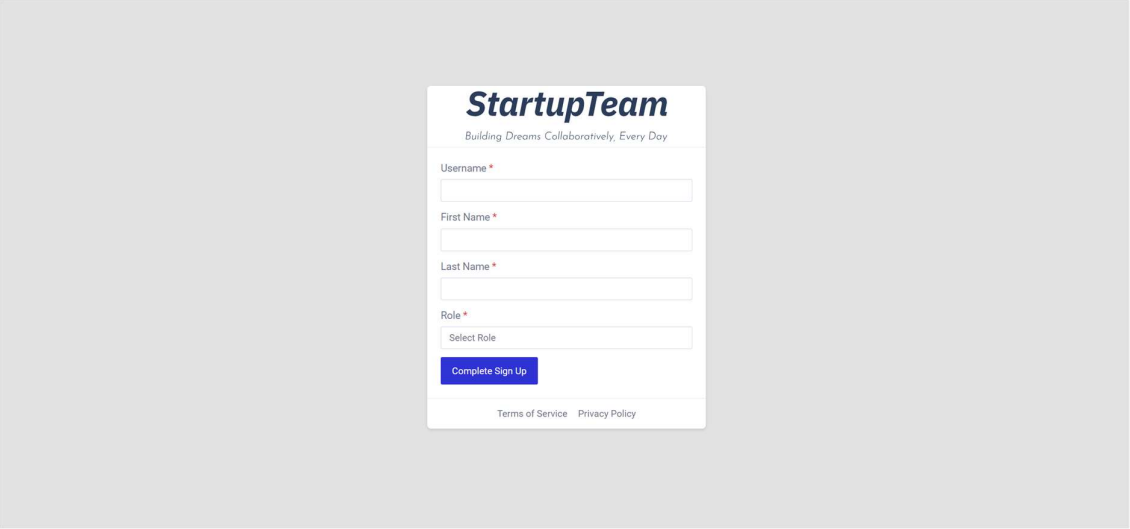
The figure below displays the SignIn page, where users can enter their email and password to log in. The page includes an option for Sign in with Google to provide users with an alternative login method.



The figure below shows the SignUp page, allowing new users to register with their email, password, and confirm password fields.

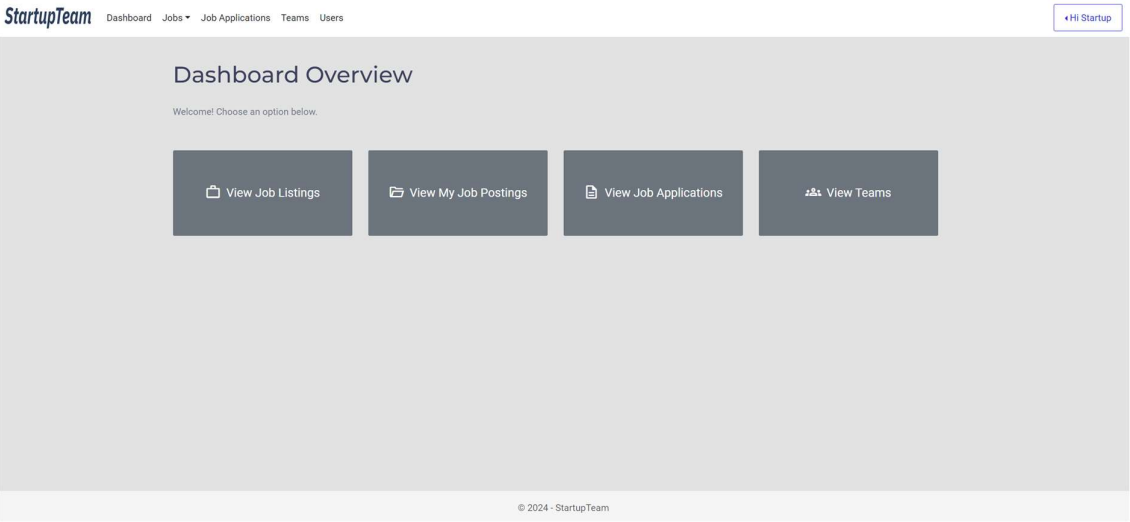


The figure below shows Step Two of the SignUp process, where users enter their username, first name, and last name, and select their role as either a startup founder or a skilled individual.



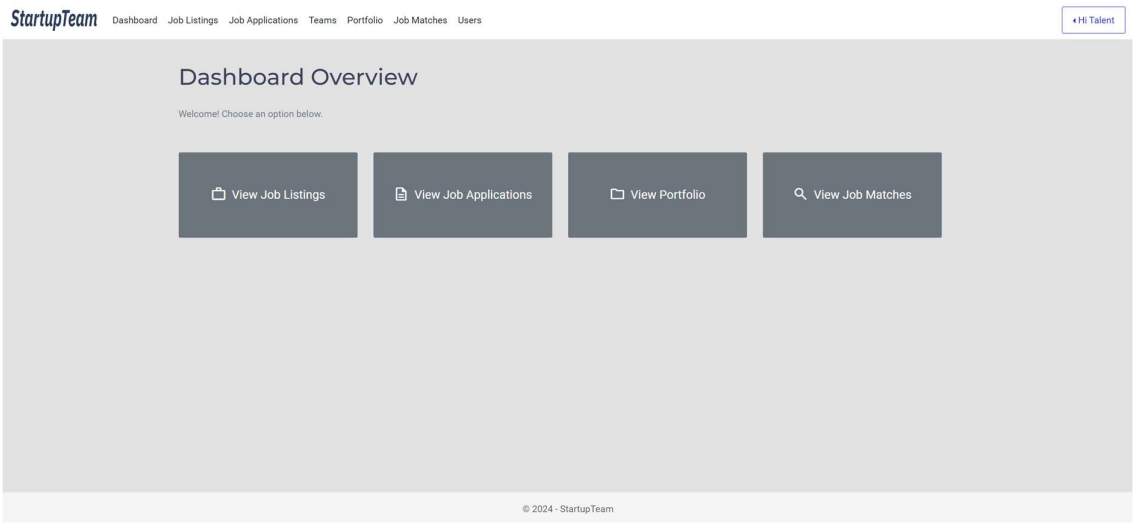
L.2 Startup Founder Panel

The figure below shows the Startup Founder Panel, where startup founders can manage job postings, view teams, and oversee job applications. They can also view job listings, and user profiles, see applicant matches, and update their profile.



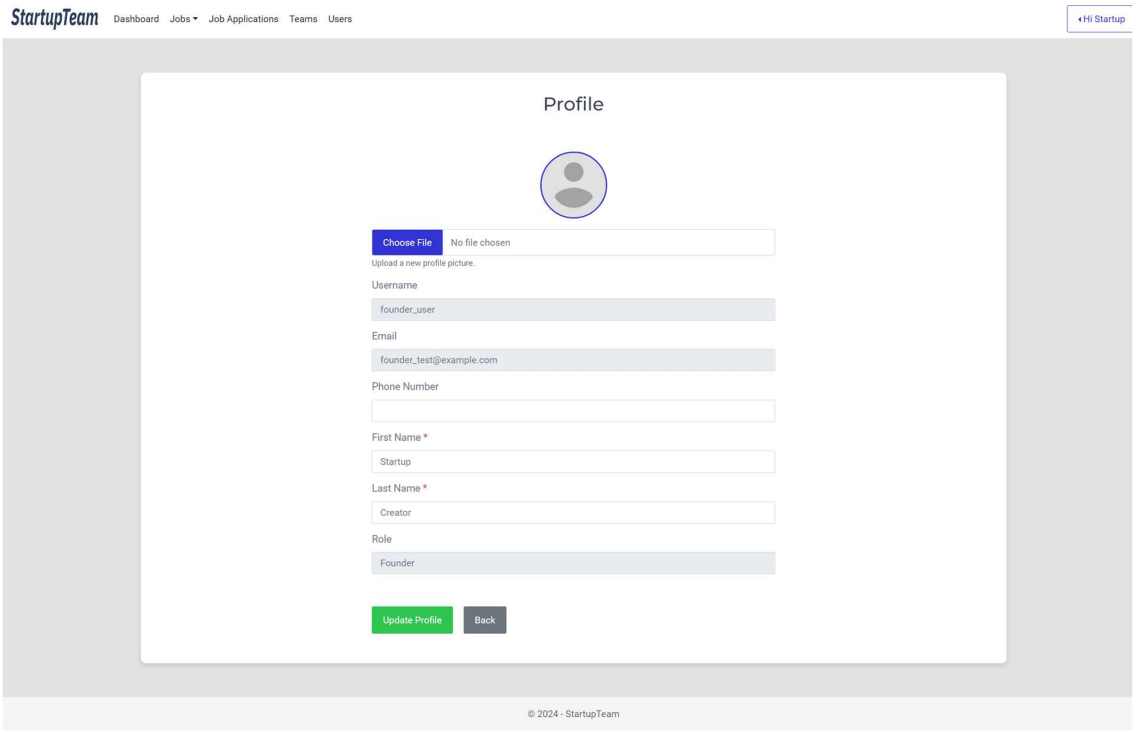
L.3 Skilled Individual Panel

The figure below shows the Skilled Individual Panel, where skilled individuals can manage their portfolios, view teams, apply for jobs, and update their job applications. They can also view job listings, and user profiles, see job matches, and update their profile.



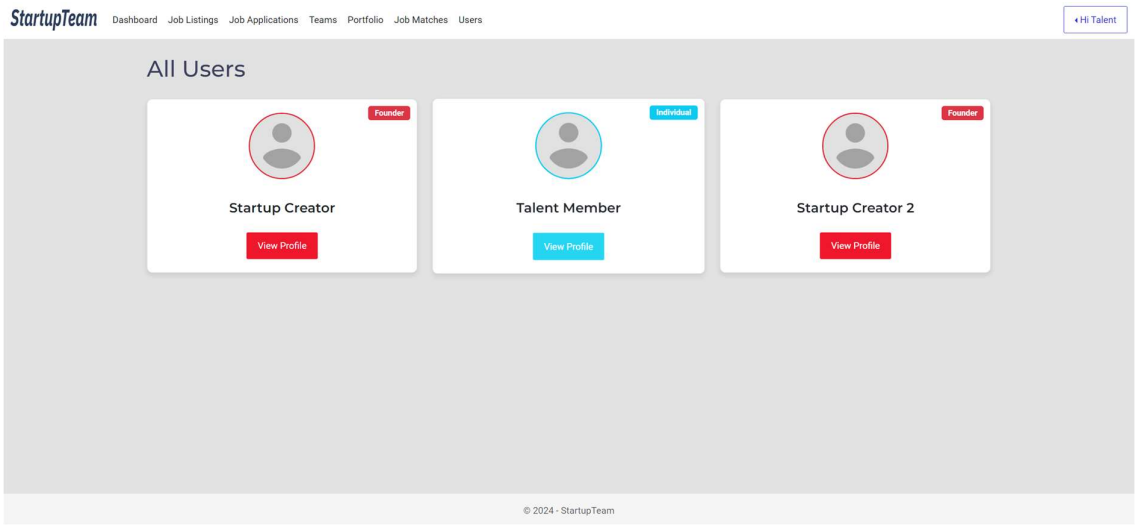
L.4 Profile Update Form

The figure below shows the Profile Update Form, where users can update their phone number, first name, last name, and profile picture.



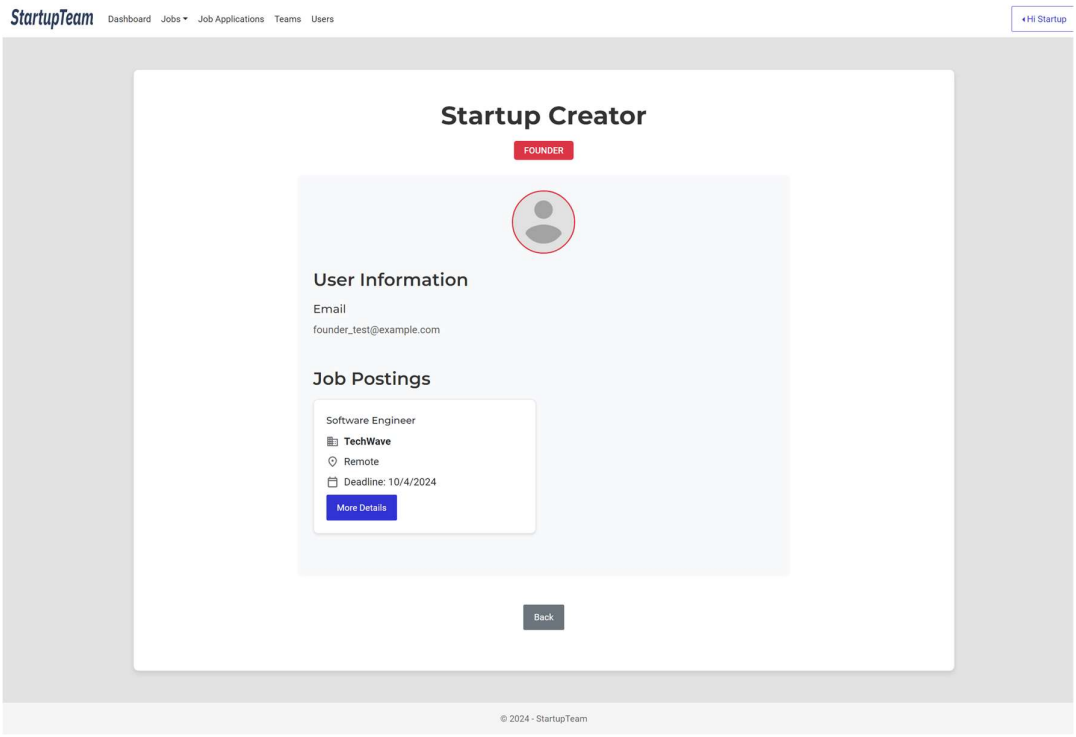
L.5 All Users Page

The figure below shows the public profiles of all startup founders and skilled individuals registered on the platform.



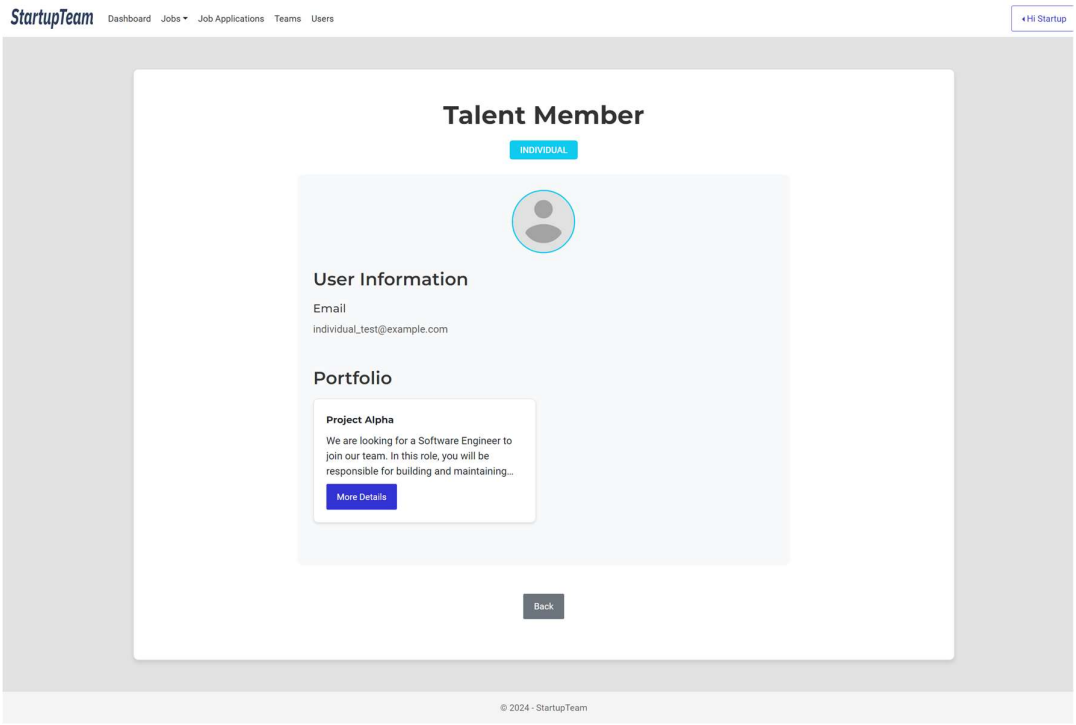
L.6 Startup Founder Public Page

The figure below shows the public profile of a startup founder, including their information and the job advertisements they have posted.



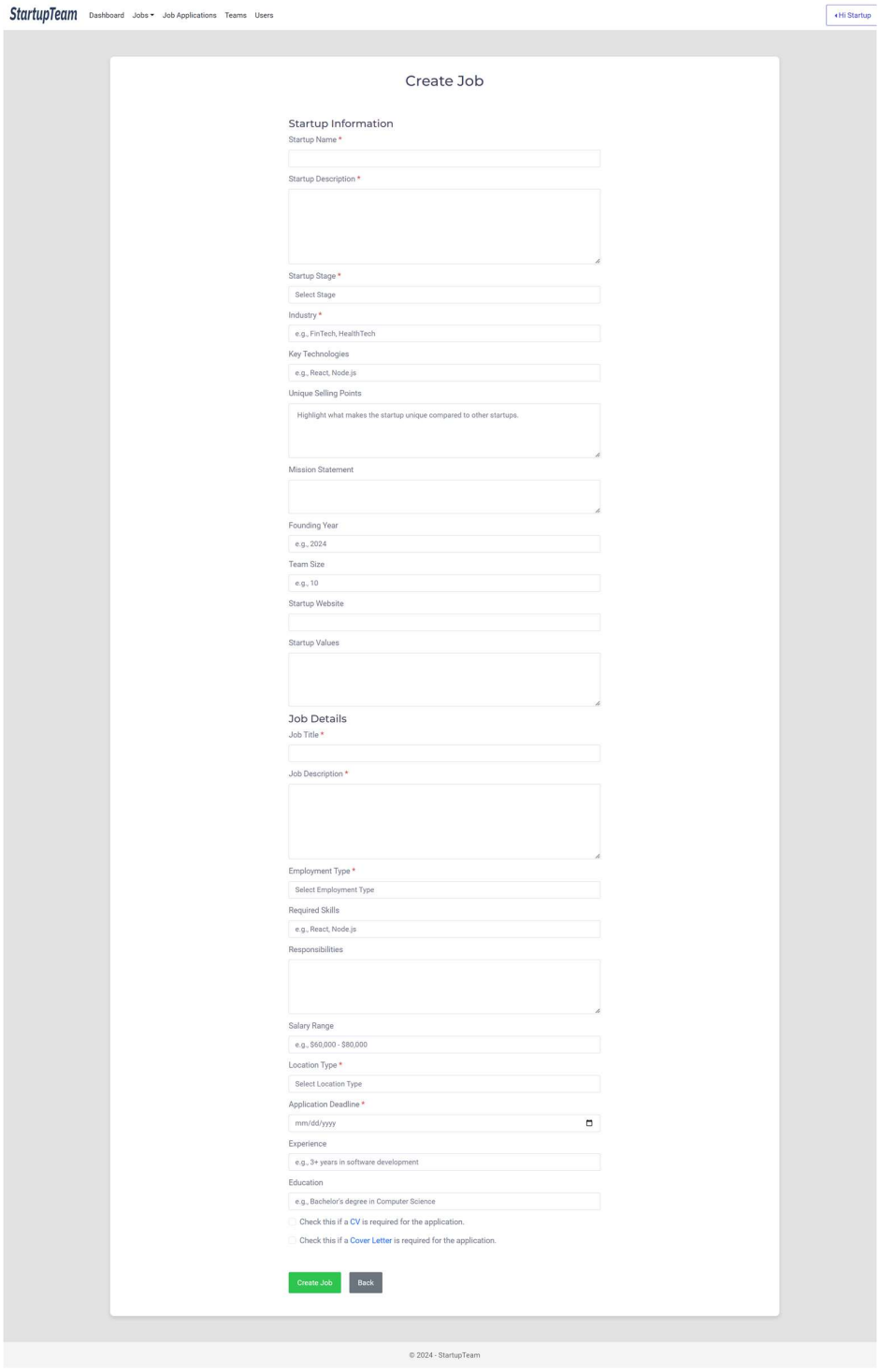
L.7 Skilled Individual Public Page

The figure below shows the public profile of a skilled individual, including their information and their portfolio.



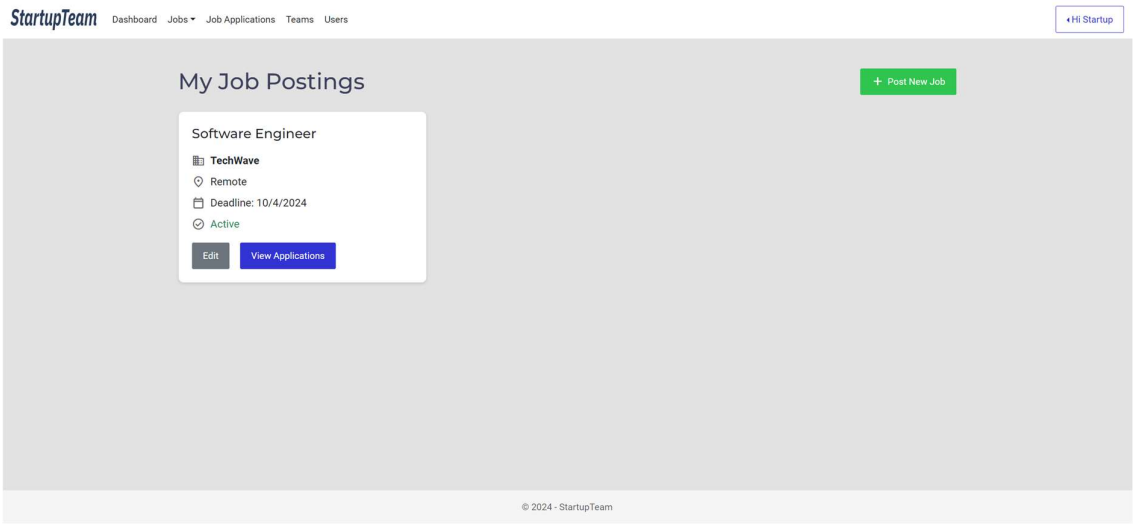
L.8 Job Advertisement Creation Form

The figure below shows the Job Advertisement Creation Form, where startup founders can post jobs by adding startup information such as the startup name, description, stage, industry, and job details like the title, description, and employment type.



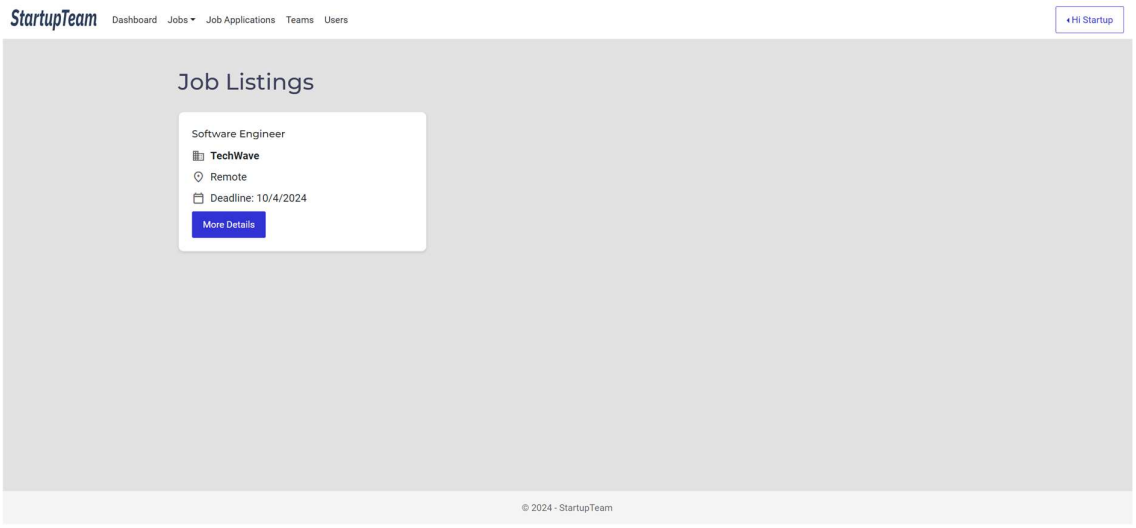
L.9 Job Advertisements Management Page

The figure below shows the Job Advertisement Management Page, where startup founders can create new job advertisements, update their job advertisements, and view job applications for each job posting.



L.10 Job Listings Page

The figure below shows the Job Listings Page, where users can view job listings and skilled individuals can apply for job advertisements.



L.11 Job Advertisement Page

The figure below shows the Job Advertisement Page, including startup information and job details. Skilled individuals can apply for the startup job by uploading their CVs and cover letters as per the job requirements.

Software Engineer

TechWave FullTime

Startup Information

Startup Description

TechWave is a rapidly growing startup that focuses on building innovative software solutions for small and medium-sized businesses. We provide digital transformation tools to help businesses automate their operations and scale efficiently.

Startup Stage

SeriesA

Industry

FinTech

Key Technologies

React, Node.js, AWS, Docker

Unique Selling Points

We are a diverse, inclusive team driven by a passion for technology and innovation. Our flexible work environment, combined with challenging projects, offers an unmatched growth opportunity.

Mission Statement

Empowering small businesses to thrive in the digital era with cutting-edge technology and scalable solutions.

Founding Year

2022

Team Size

25

Startup Values

Innovation, Collaboration, Diversity, Integrity, and Customer Success.

Job Details

Job Description

We are looking for a Software Engineer to join our team. In this role, you will be responsible for building and maintaining key features of our core platform. You will collaborate with cross-functional teams to deliver high-quality, scalable software.

Required Skills

Proficiency in React and Node.js Experience with AWS Familiarity with Docker and containerization Solid understanding of database technologies (SQL, NoSQL)

Responsibilities

Design, develop, and maintain new features on our platform Work with cross-functional teams to define and implement new features Troubleshoot and resolve software issues Participate in code reviews to ensure high-quality codebase Ensure the platform's scalability and security

Salary Range

$70,000 - $90,000

Location Type

Remote

Application Deadline

2024-10-04T00:00:00

Experience

3+ years in software development

Education

Bachelor's degree in Computer Science or related field

Application Materials

CV Required Cover Letter Required

Upload Documents

CV *

Choose File No file chosen

Cover Letter *

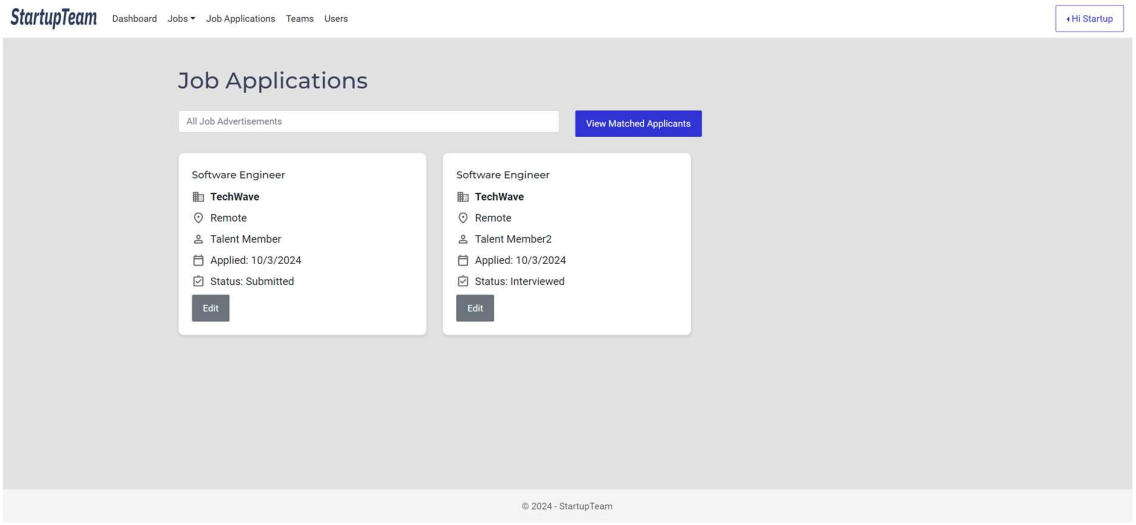
Choose File No file chosen

Submit Application

Back

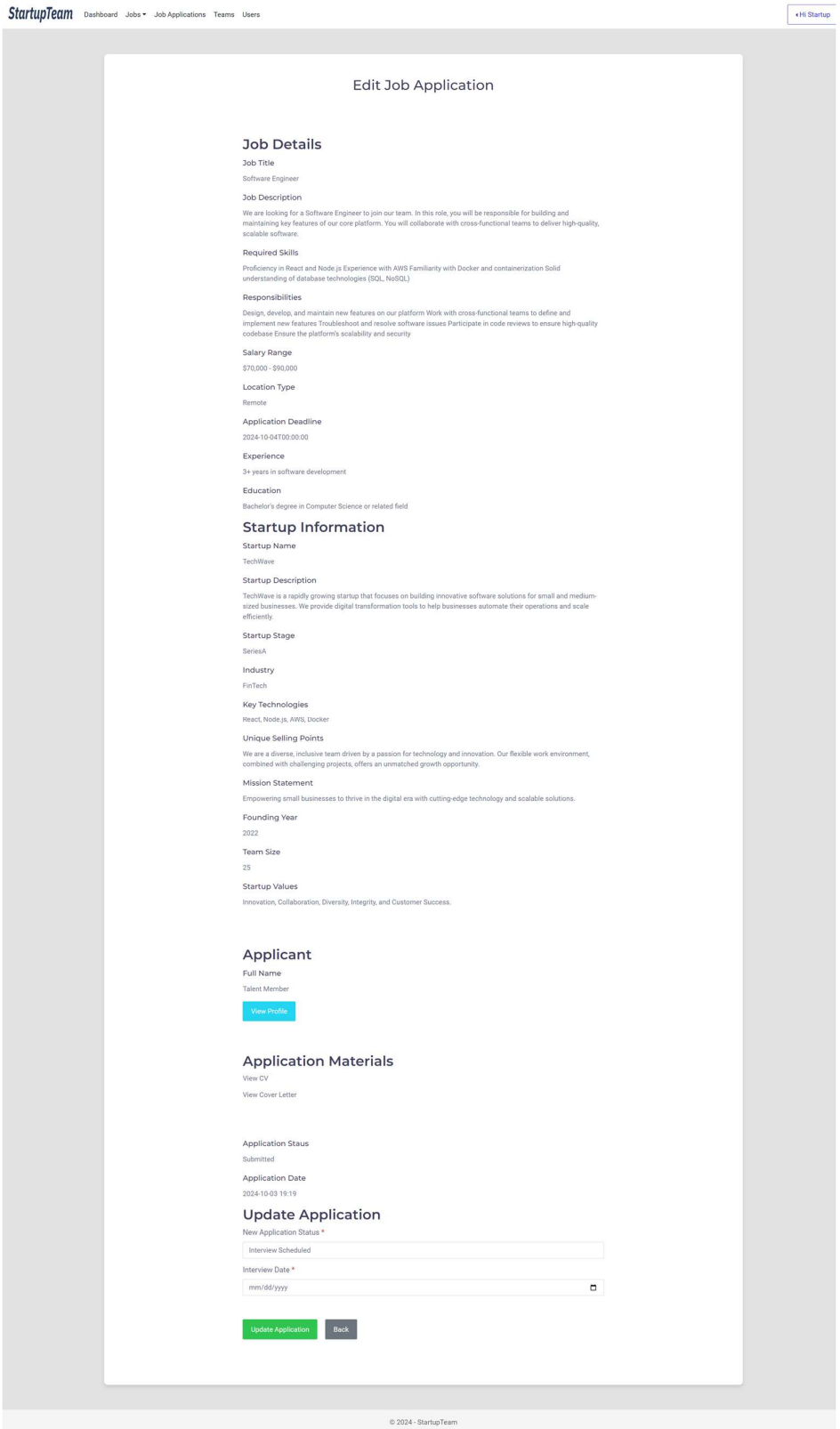
L.12 Job Applications Management Page

The figure below shows the Job Applications Management Page, where startup founders can view applications for their jobs, filter job applications for each job posting, and apply the matching algorithm to see the best matching candidates from the applicants for each job posting. Skilled individuals also have a similar page where they can view all their job applications and update them.



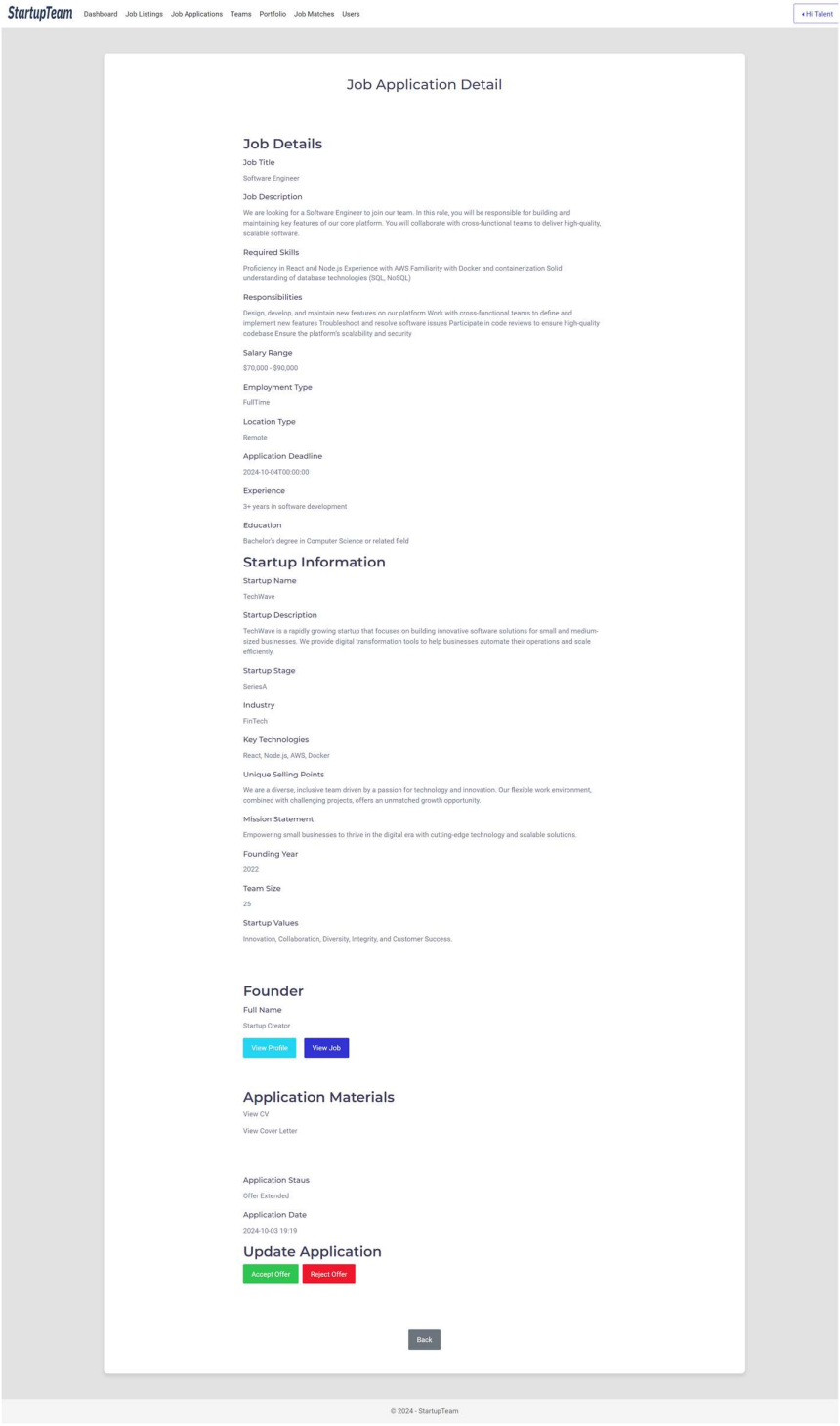
L.13 Startup Founder Job Application Update Page

The figure below shows the Job Application Update Page, where startup founders can review an application, view the job applicant's profile, see their uploaded CV and cover letter, and update the application status or set an interview date.


StartupTeam
Dashboard
Jobs
Job Applications
Teams
Users
Hi Startup
Edit Job Application
Job Details
Job Title
Software Engineer
Job Description
We are looking for a Software Engineer to join our team. In this role, you will be responsible for building and maintaining key features of our core platform. You will collaborate with cross-functional teams to deliver high-quality, scalable software.
Required Skills
Proficiency in React and Node.js Experience with AWS Familiarity with Docker and containerization Solid understanding of database technologies (SQL, NoSQL)
Responsibilities
Design, develop, and maintain new features on our platform Work with cross-functional teams to define and implement new features Troubleshoot and resolve software issues Participate in code reviews to ensure high-quality codebase Ensure the platform's scalability and security
Salary Range
$70,000 - $90,000
Location Type
Remote
Application Deadline
2024-10-04T00:00:00
Experience
3+ years in software development
Education
Bachelor's degree in Computer Science or related field
Startup Information
Startup Name
TechWave
Startup Description
TechWave is a rapidly growing startup that focuses on building innovative software solutions for small and medium-sized businesses. We provide digital transformation tools to help businesses automate their operations and scale efficiently.
Startup Stage
SeriesA
Industry
FinTech
Key Technologies
React, Node.js, AWS, Docker
Unique Selling Points
We are a diverse, inclusive team driven by a passion for technology and innovation. Our flexible work environment, combined with challenging projects, offers an unmatched growth opportunity.
Mission Statement
Empowering small businesses to thrive in the digital era with cutting-edge technology and scalable solutions.
Founding Year
2022
Team Size
25
Startup Values
Innovation, Collaboration, Diversity, Integrity, and Customer Success.
Applicant
Full Name
Talent Member
View Profile
Application Materials
View CV
View Cover Letter
Application Staus
Submitted
Application Date
2024-10-03 19:19
Update Application
New Application Status *
Interview Scheduled
Interview Date *
mm/dd/yyyy
Update Application
Back
© 2024 - StartupTeam

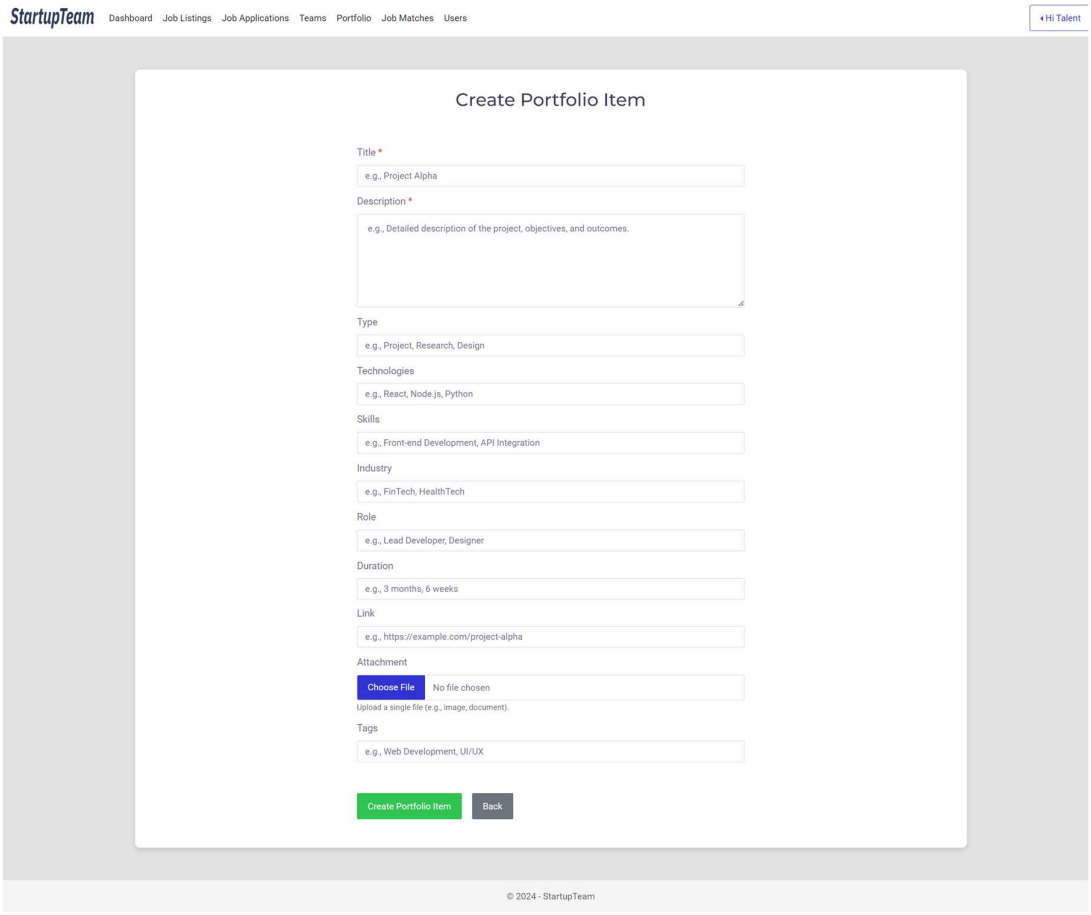
L.14 Skilled Individual Job Application Update Page

The figure below shows the Job Application Details Page, where skilled individuals can review their applications, view the founder's profile and related job advertisement, and see their application status. They can also accept or reject the job offer if they have received one.



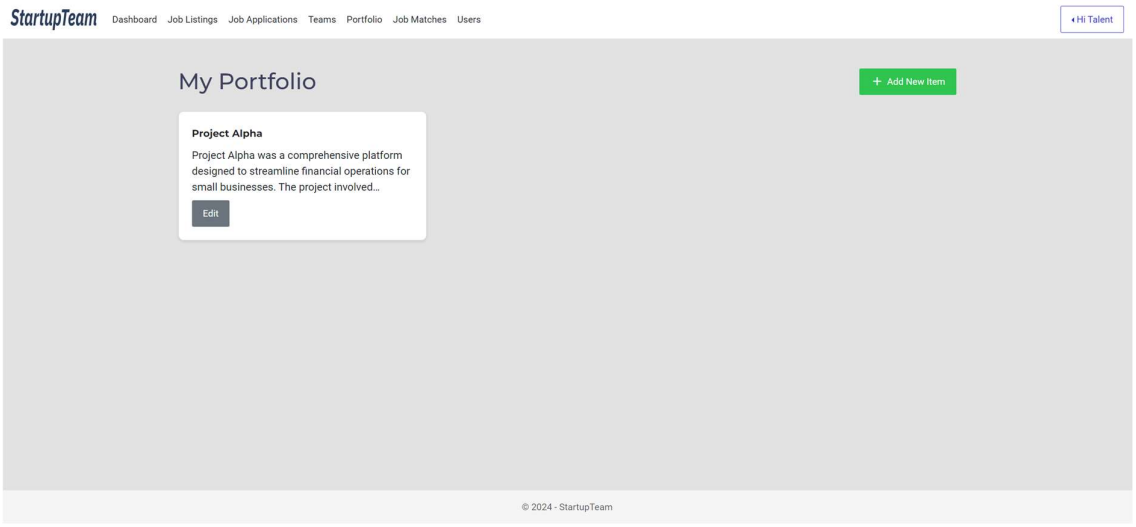
L.15 Portfolio Item Creation Form

The figure below shows the Portfolio Item Creation Form, where skilled individuals can add an item to their portfolio by providing the portfolio item title, description, type, technologies, and other relevant details.



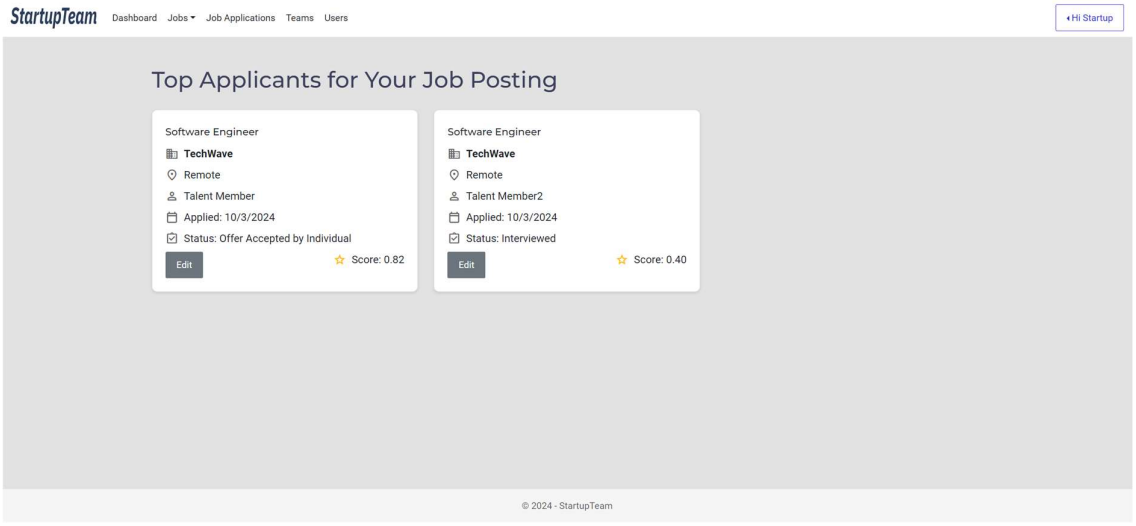
L.16 Portfolio Management Page

The figure below shows the Portfolio Management Page, where skilled individuals can view their portfolio items, add new items to their portfolio, and edit existing items.



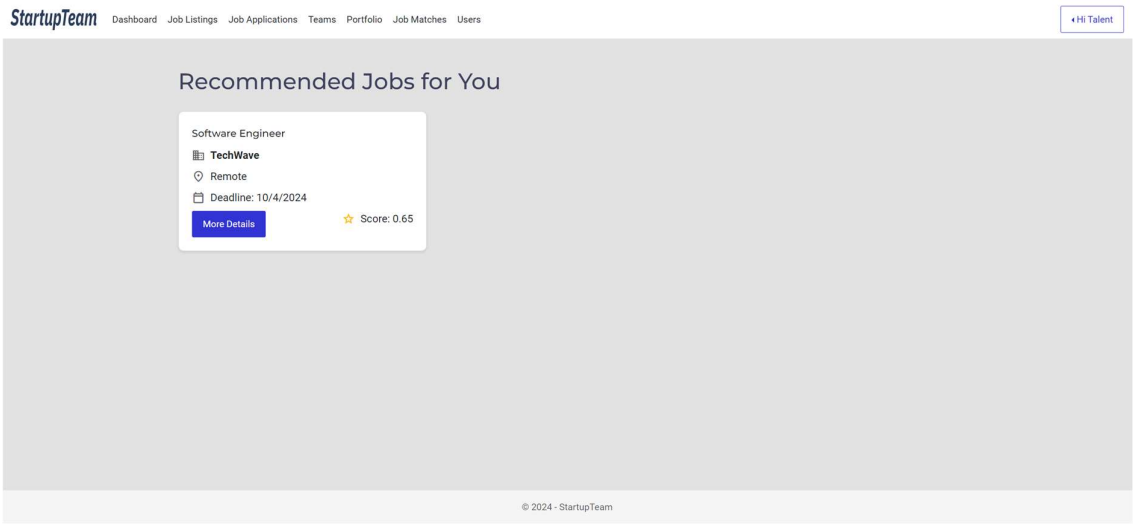
L.17 Job Applicants Matching Page

The figure below shows the Job Applicants Matching Page for startup founders, where they can view the top matches from skilled individuals who applied for their startup jobs.



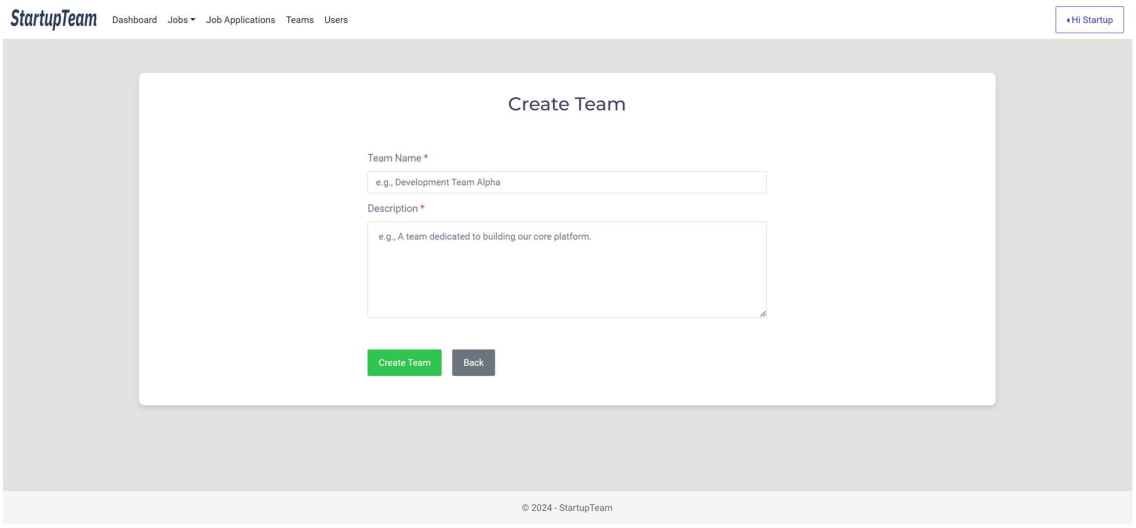
L.18 Job Advertisements Matching Page

The figure below shows the Job Advertisements Matching Page for skilled individuals, where they can view the top matches from startup job advertisements that align with their portfolio.



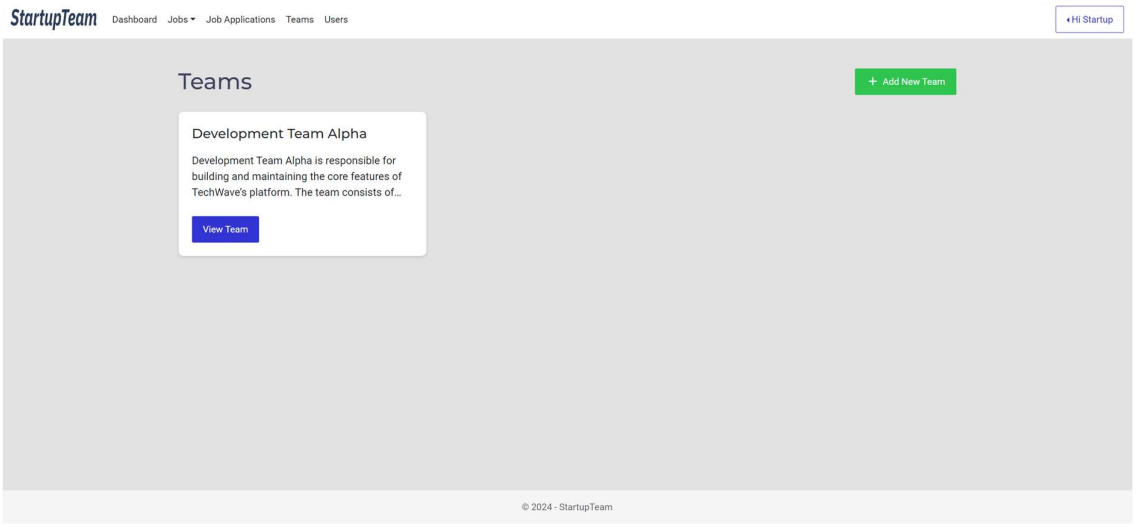
L.19 Team Creation Form

The figure below shows the Team Creation Form, where startup founders can create their startup team by entering the team name and description.



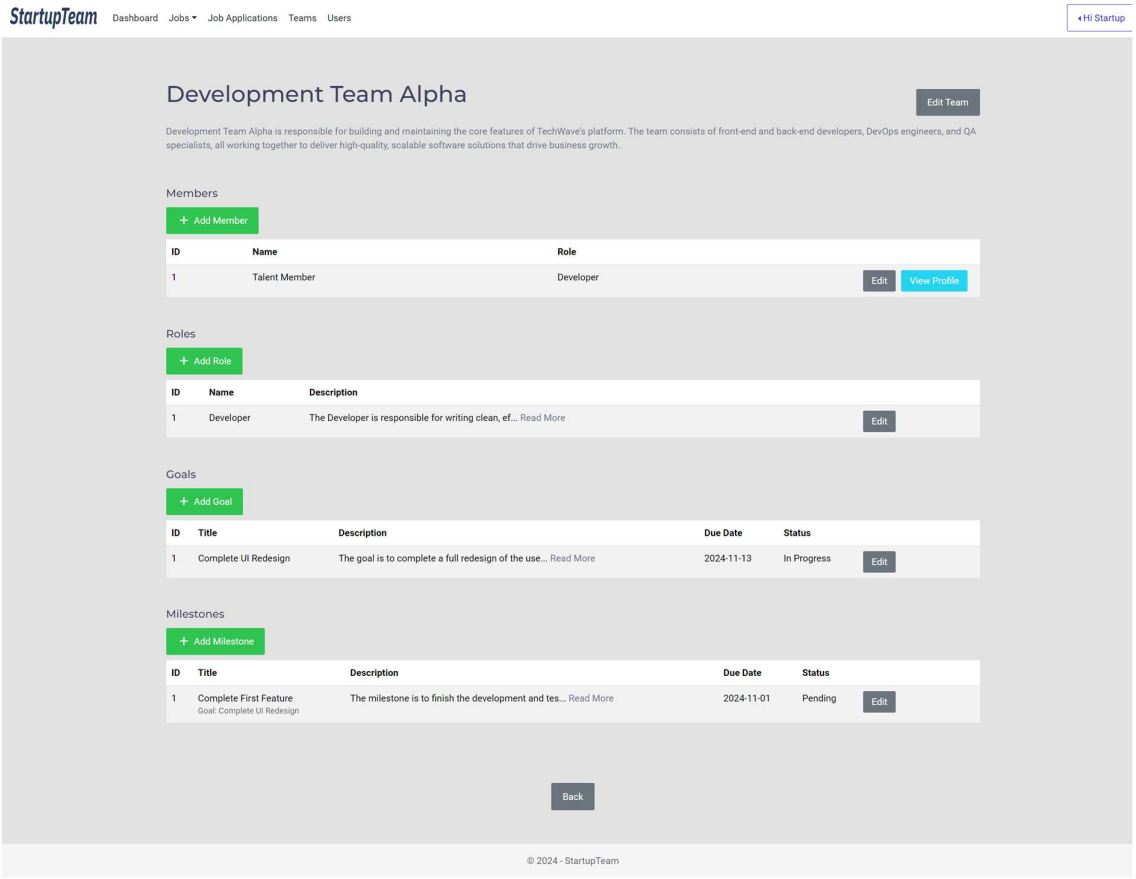
L.20 Teams Management Page

The figure below shows the Teams Management Page, where startup founders can create and view their startup teams, while skilled individuals can only view their assigned teams.



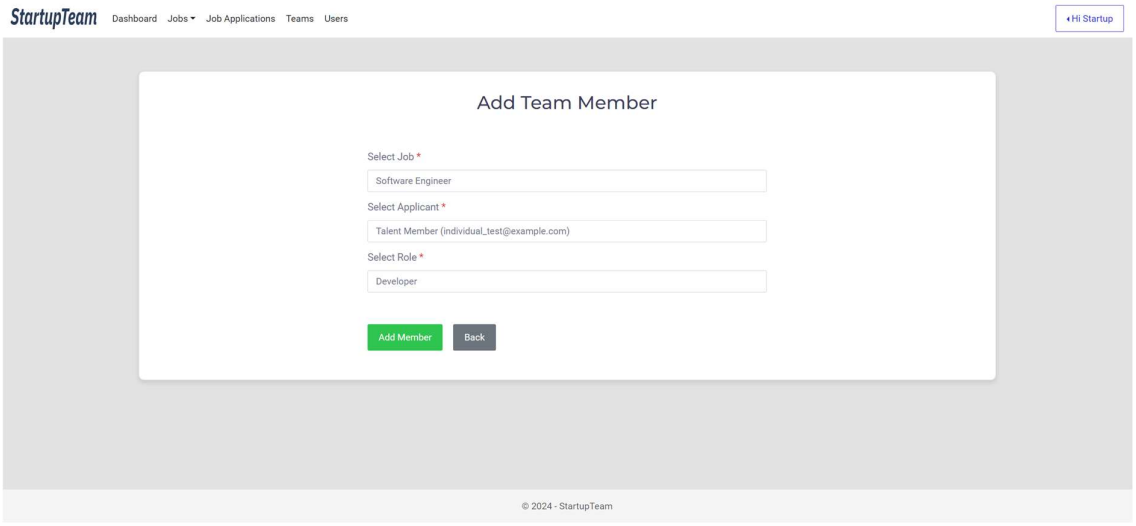
L.21 Team Overview Page

The figure below shows the Team Overview Page, displaying team members, roles, goals, and milestones within the team. Startup founders have access to edit these items, while skilled individuals can only view the details.



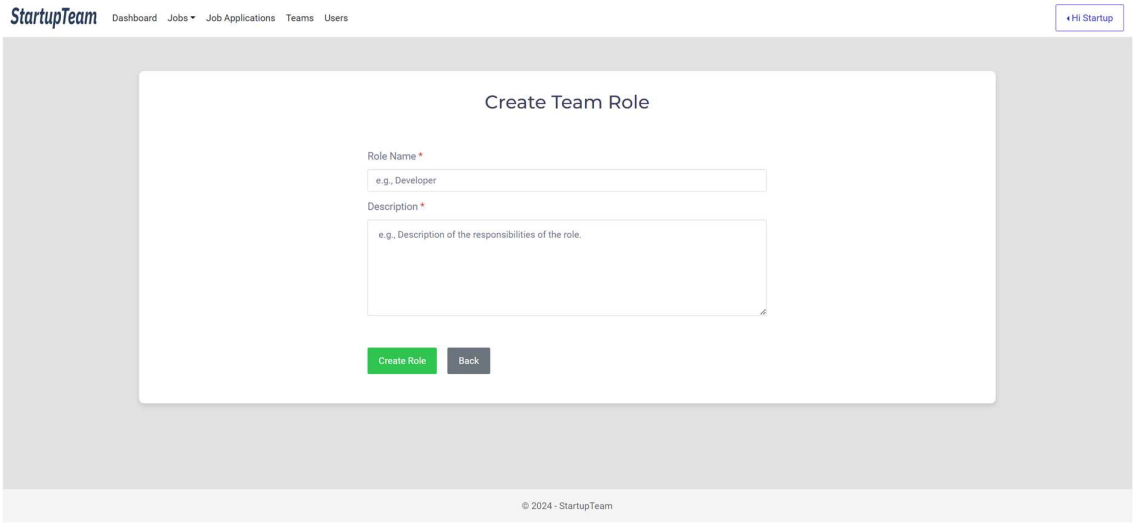
L.22 Team Member Addition Form

The figure below shows the Add Team Member Form, where startup founders can add a new member by selecting a job advertisement, choosing an accepted candidate from the skilled individuals who applied for the job, assigning them a role, and adding them to their startup team.



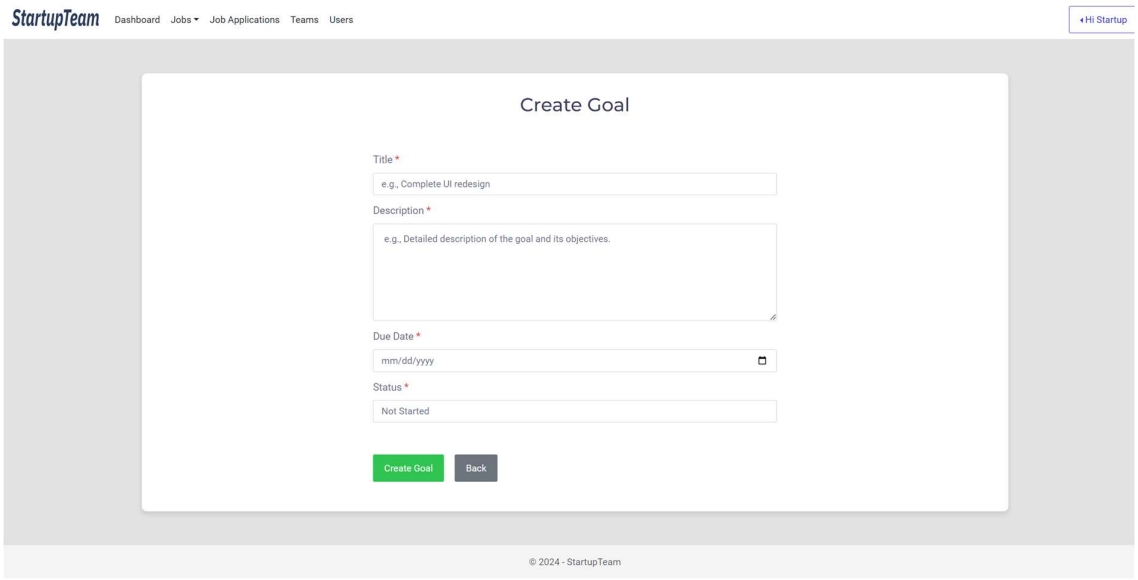
L.23 Team Role Creation Form

The figure below shows the Create Team Role Form, where startup founders can create a new team role by defining the role name and description.



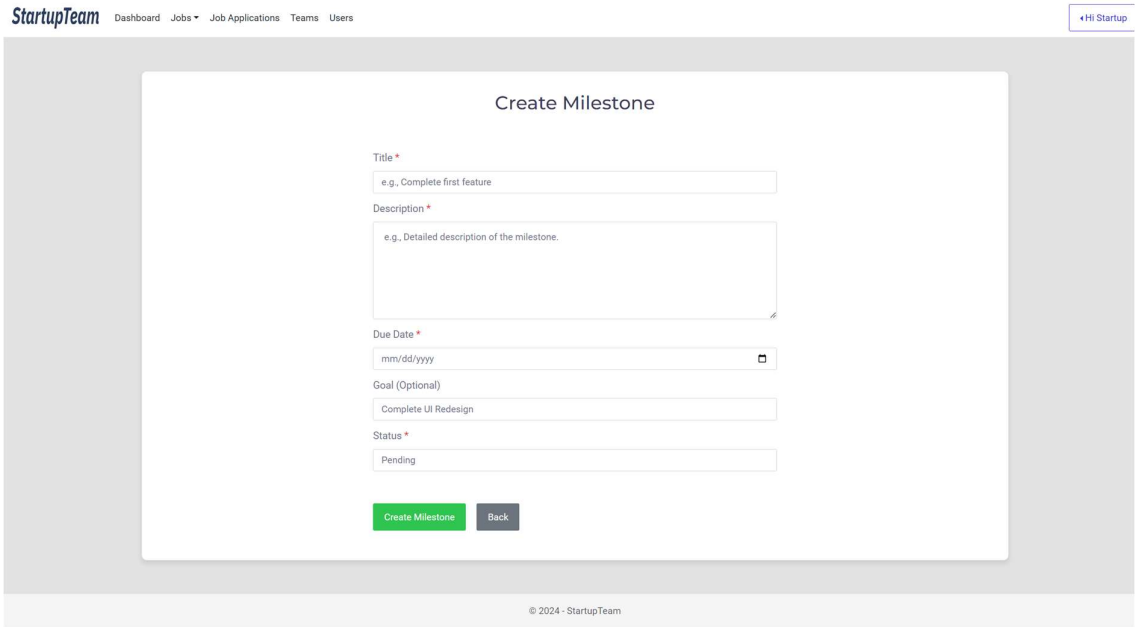
L.24 Goal Creation Form

The figure below shows the Create Goal Form, where startup founders can create a new goal by defining the title, description, and due date, and setting the status.



L.25 Milestone Creation Form

The figure below shows the Create Milestone Form, where startup founders can create a new milestone by defining the title, description, due date, and status. Optionally, they can add a goal to the milestone, which helps break down a goal into milestones if needed.



Appendix M: Information Sheet and Informed Consent Sheet

M.1 Information Sheet

INFORMATION SHEET

REC Reference Number: [Insert Reference Number]
Date: 17 September 2024
Version Number: 1.0

Title of study:
Usability Testing of a Technology-Based Startup Team Organization Platform

Researcher:
Ali Momenzadeh Kholenjani
ali.momenzadeh-kholenjani@city.ac.uk

We would like to invite you to take part in a research study. Before you decide whether or not to participate, it is important that you understand why the research is being done and what it will involve for you. Please take time to read the following information carefully, and feel free to ask any questions if anything is unclear. You will be given a copy of this information sheet to keep.

What is the purpose of the study?
This study aims to evaluate the usability of a platform designed to help organize and manage technology-based startup teams. The platform offers features for job advertisement posting, portfolio management, candidate job matching, and team management. By conducting this usability test, we hope to gather feedback on how effectively users can interact with the platform and identify areas for improvement. This study is part of City University's MSc Software Engineering with Cloud Computing final project.

Why have I been invited to take part?
You have been invited to take part in this study because you fit the profile of a potential user of the platform, such as a startup founder or a skilled individual seeking job opportunities within a startup environment. We are aiming to recruit 5 participants for this study. Participation is open to individuals with experience in startups or relevant skills in technology-based industries. If you are an employee, please be assured that choosing to take part or not will have no impact on your employment, promotion prospects, or any work-related assessments.

Do I have to take part?
Participation in this project is entirely voluntary. You are free to decide whether or not you wish to take part. If you choose to participate, you will be asked to sign a consent form. Even if you agree to take part, you are still free to withdraw at any time without providing a reason and without any penalty. Please note that once you have completed the usability test and your data has been anonymized, it will no longer be possible to withdraw your information. Any data collected up until the point of withdrawal will be retained and used in the study in its anonymized form.

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

What will happen if I take part?

If you decide to participate, you will take part in a 90–120 minute online usability test. During this session, you will be asked to complete a series of tasks designed to simulate real-world interactions with the platform. These tasks may include actions such as posting a job, applying for a job, or managing team members, roles, and goals within a team. This will be a one-time session only. There is no need for follow-up meetings after the usability test. The usability test will take place online, and you will be able to participate from your own location using your computer. I will provide you with a link to the online meeting platform (e.g. Google Meet).

Session Structure:

- **Introduction (5-10 minutes):** I will provide an overview of the session and explain the tasks you will be completing.
- **Usability Tasks (60-90 minutes):** You will be asked to interact with the platform and complete a series of tasks. I will observe your interactions to understand how intuitive and user-friendly the platform is. You will be required to share your screen and provide verbal feedback throughout the process.
- **Post-Test Feedback (10-15 minutes):** At the end of the session, you will be asked to complete a brief online survey to rate your experience and provide additional feedback on the platform's usability. The survey will include both rating-scale questions and open-ended questions to capture your overall impressions.

Data Collection:

- **Screen and Voice Recording:** During the usability test, your screen interactions and verbal feedback will be recorded. You will be informed in advance of the recording, and consent for this will be sought prior to the session. These recordings will be securely stored on OneDrive, adhering to City, University of London's data protection policies. Local copies of the recordings will be deleted after analysis is complete. However, the recordings will be kept on OneDrive for review if necessary, and access will only be provided to relevant academic personnel for evaluation purposes.
- **Personal Data:** To participate, you will be provided with a unique username and password to interact with the platform, ensuring that no personally identifiable information (PII) such as your real name or email is collected during the session. All data entered during the test, such as job postings, profile information, or applications, will be anonymized and used solely for the purpose of assessing the platform's usability. This data will not be shared externally or used beyond this research. Once the analysis is completed, this data will be deleted.
- **Survey Responses:** At the end of the session, you will complete an online survey. Your responses will be anonymized, and no identifiable information will be linked to your answers. Raw survey data will be stored securely on OneDrive and shared only with the academic personnel. As part of the project submission, a link to this anonymized raw data will be provided for academic review. The raw survey data will remain stored in accordance with City, University of London's data retention policy.

What are the possible disadvantages and risks of taking part?

We do not foresee any risks in taking part in this study. However, if you feel uncomfortable at any point, you are free to withdraw without any penalty.

CITY
UNIVERSITY OF LONDON
EST 1894

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

What are the possible benefits of taking part?
While there are no direct personal benefits related to the research itself, your feedback will help improve the platform, which could benefit future users in the startup community. Additionally, your participation will contribute to valuable research on enhancing team organization tools for technology-based startups.

Expenses and Payments
Participants will be compensated for their time at the London Living Wage rate of £13.50 per hour. For this study, you will receive £27 for the two-hour usability test. Payments will be made via bank transfer after the session. Please note that this payment is compensation for your time and will not affect your employment status or any benefits you may be receiving.

How is the project being funded?
This project is part of my MSc degree and is self-funded. There are no external sponsors or conflicts of interest.

What should I do if I want to take part?
If you would like to participate, please contact me at ali.momenzadeh-kholenjani@city.ac.uk to arrange a time for the usability testing session. You will then be provided with a consent form to complete before the session begins.

Data privacy statement
City, University of London is the sponsor and the data controller of this study based in the United Kingdom. This means that we are responsible for looking after your information and using it properly. The legal basis under which your data will be processed is City's public task.

Your rights to access, change, or move your information are limited, as we need to manage your information in a specific way in order for the research to be reliable and accurate. To safeguard your rights, we will use the minimum personally identifiable information possible (for further information please see https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/lawful-basis-for-processing/public-task/).
City will use your name and contact details to contact you about the research study as necessary. If you wish to receive the results of the study, your contact details will also be kept for this purpose. The only people at City who will have access to your identifiable information will be members of the research team, and, if appropriate, individuals with responsibility for monitoring and auditing at City, including research projects. There may be occasions when regulatory authorities may access research data in accordance with their statutory powers. City will keep identifiable information about you from this study for 10 years after the study has finished.

You can find out more about how City handles personal data by visiting https://www.city.ac.uk/about/governance/policies/data-protection-policy. You can also read City's general privacy notice by visiting https://www.city.ac.uk/about/governance/policies/general-privacy-notice . If you are concerned about how we have processed your personal data, you can contact the Information Commissioner's Office (ICO) directly https://ico.org.uk/.

Will my taking part in the study be kept confidential?
Yes, your participation will be kept completely confidential. All data collected during the usability test will be anonymized, and no personally identifiable information will be included in the research report.

CITY
UNIVERSITY OF LONDON
EST 1894

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

Only the research team (my supervisor and I) will have access to the data, along with any academic personnel involved in evaluating the project. Any recordings, transcripts, or other data will be securely stored on OneDrive, adhering to City, University of London's data protection policies. Local copies of the recordings will be stored securely on a password-protected computer and deleted after the analysis and project submission on 2nd October 2024. The recordings will remain on OneDrive in case they are required for academic review, in accordance with City, University of London's data protection policies.

What will happen to the results?

The results of this study will be used as part of my MSc dissertation at City, University of London in an anonymized format. While it will not be possible to obtain a copy of the dissertation, you can contact me by email at ali.momenzadeh-kholenjani@city.ac.uk if you would like to receive a summary of the findings related to your participation.

Who has reviewed the study?

This study has been approved by City, University of London Computer Science Research Ethics Committee (CSREC).

What if there is a problem?

If you have any problems, concerns, or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can do this through City's complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary of Senate Research Ethics Committee and inform them that the name of the project is Technology-based Startup Team Organization Platform.

You can also write to the Secretary at:

Annah Whyton
Research & Enterprise Office
City, University of London
Northampton Square
London, EC1V 0HB
Email: senaterec@city.ac.uk

Insurance

City University London holds insurance policies that apply to this study, subject to the terms and conditions of the policy. If you feel you have been harmed or injured by taking part in this study you may be eligible to claim compensation. This does not affect your legal rights to seek compensation. If you are harmed due to someone's negligence, then you may have grounds for legal action.

Further information and contact details

For more information about this study, or if you have any queries or concerns, contact Dr Christos Kloukinas (c.kloukinas@city.ac.uk) as the project supervisor.

Thank you for taking the time to read this information sheet.

CITY
UNIVERSITY OF LONDON
EST 1894

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

M.2 Informed Consent Sheet

INFORMED CONSENT

REC Reference Number: [Insert Reference Number]
Date: 17 September 2024
Version Number: 1.0

Title of study:
Usability Testing of a Technology-Based Startup Team Organization Platform

Researcher:
Ali Momenzadeh Kholenjani
ali.momenzadeh-kholenjani@city.ac.uk

		Please tick the boxes to indicate your agreement
1	I confirm that I have read and understood the participant information dated 17 September 2024, Version 1.0, for the above study. I have had the opportunity to consider the information and ask questions that have been answered satisfactorily.	
2.	I understand that my participation is voluntary and that I am free to withdraw without giving a reason and without being penalised or disadvantaged.	
3.	I agree to my screen interactions and voice being recorded during the usability test.	
4.	I understand that raw data, including screen recordings and survey responses, will be stored securely on OneDrive in accordance with City, University of London's data protection policies, and may be retained for academic review.	
5.	I understand that my personal data will not be included in any reports, and all identifiable data will be anonymized before analysis.	
6.	I understand that the data I input into the platform, including job postings, profile information, or applications, will be anonymized and used solely for assessing the platform's usability. No personally identifiable information (PII) like my real name or email will be collected, and all data will be deleted after analysis.	
7.	I understand that I can withdraw from the study at any time without penalty. However, once the usability test is complete and my data has been anonymized, it will no longer be possible to withdraw my information.	

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

8.	I agree to City recording and processing this information about me. I understand that this information will be used only for the purpose(s) explained in the participant information and my consent is conditional on City complying with its duties and obligations under the General Data Protection Regulation (GDPR).	
9.	I would like to be informed of the results of this study once it has been completed and I understand that my contact details will be retained for this purpose.	
10	I agree to take part in the above study.	

_____	_____	_____
Name of Participant	Signature	Date
_____	_____	_____
Name of Researcher	Signature	Date

When completed, 1 copy for participant; 1 copy for researcher file.

City, University of London
Northampton Square
London EC1V 0HB
Tel: +44 (0)207 040 5060 www.city.ac.uk

Appendix N: Usability Test Overview

Usability Test Overview

Welcome! Thank you for participating in this usability test. Your feedback is invaluable in helping improve the platform. Below is an overview of what to expect during the session.

Purpose of the Test

This usability test aims to evaluate the effectiveness and user-friendliness of the platform from both the startup founder and skilled individual perspectives. You will switch roles during the session to simulate real-world interactions.

Participant Roles

- **Startup Founder**: You will post job advertisements, review candidate applications, and manage your team.
- **Skilled Individual**: You will apply for jobs, manage your portfolio, and review job matches.

General Guidelines

- **Data Generation**: You are encouraged to use ChatGPT to generate input data (e.g., job details and portfolio items) during the test if needed. This will help ensure consistent test data and streamline the process.
- **Browser Requirement**: Please use a different browser to sign in as the startup founder and the skilled individual during the test (e.g., if you post the job in Chrome, use Firefox or another browser to apply for the job).

Test Structure

The usability test will consist of the following tasks:

Task 1: Posting a Job and Adding Talented Individuals to a Team

Task 2: Portfolio Showcase and Its Impact on Recruitment Decisions

Task 3: Matching Suitable Candidates with Job Postings

Task 4: Defining Roles, Goals, and Milestones

Final Step

After completing all tasks, you will fill out a survey to provide feedback on your overall experience with the platform, including ease of use, satisfaction with features, and any suggestions for improvement.

Thank you for your participation.

Now you can start the test!

Task 1: Posting a Job and Adding Talented Individuals to a Team

Task Scenario: You are a startup founder looking to hire a web developer for your project. Your goal is to create a job post and, after reviewing candidate applications, add a suitable candidate to your team.

Detailed Task Steps:

1. Create a Job Post:
    - Action: Navigate to the job posting section of the platform.
    - Action: Fill in the job title (e.g., Web Developer), job description (e.g., Must be proficient in React and Node.js), required skills, etc.
    - Action: Submit the job posting.
2. Apply for the Job (Simultaneous Action):
   Action: After posting the job, open a new browser window using a browser different from the current one (e.g., if you posted the job in Chrome, use Firefox or another browser) and sign in as an individual looking for a job.
    - Action: Search for the posted job (e.g., Web Developer) and apply using your profile.
    - Action: The researcher will also apply for the job, creating another candidate for the review process.
3. Review Candidate Applications:
    - Action: After both applications (the participant's and the researcher's) have been submitted, navigate back to the founder's job posting account and review the incoming applications.
    - Action: Evaluate the profiles of applicants. Choose one suitable candidate based on the skills and qualifications displayed in their portfolio.
4. Add a Candidate to the Team:
    - Action: After selecting a suitable candidate, add them to your team.
    - Action: Assign the new team member a role (e.g., Frontend Developer).

Task 2: Portfolio Showcase and Its Impact on Recruitment Decisions

Task Scenario (for Skilled Individuals): You are a web developer looking for a new startup job. You aim to create or update your portfolio on the platform to showcase your skills and experience.

Detailed Task Steps:

1. Update Portfolio:
    - Action: Navigate to the portfolio section.
    - Action: Add a new item to your portfolio, describing your role and the technologies used (e.g., Built a website using React and Node.js). Include links to GitHub or project URLs where applicable.

Task Scenario (for Startup Founders): As a startup founder, review several applicants' portfolios for a web developer position and make a hiring decision based on the portfolio.

Detailed Task Steps:

1. Review Portfolios:
    - Action: Review the portfolio of each candidate who has applied for your job posting.
    - Action: Compare portfolios based on technical skills, past projects, and their relevance to the job.

Task 3: Matching Suitable Candidates with Job Postings

Task Scenario (for Skilled Individuals): You are a skilled individual looking for a startup job that matches your skills. Use the platform's matching feature to find relevant job postings.

Detailed Task Steps:

1. Review Job Matches:
    - Action: Navigate to the job matching section of the platform.
    - Action: Review the list of job postings recommended by the platform's matching feature.
2. Update Portfolio and Review Job Matches Again:
    - Action: Navigate to the portfolio section and update your portfolio with new skills, projects, or other relevant details.
    - Action: Return to the job matching section and review the list of recommended jobs again, checking for any changes in the job matches after updating your portfolio.

Task Scenario (for Startup Founders): You are a startup founder looking to find a skilled individual for your web development project. Use the platform's matching algorithm to find suitable candidates.

Detailed Task Steps:

1. Review Recommended Candidates:

- Action: Use the matching feature to view a list of recommended candidates based on your job posting requirements and review their applications and portfolios.

Task 4: Defining Roles, Goals, and Milestones

Task Scenario (for Startup Founders): You've just added a new team member. Now, define their role within the team and set overall goals and milestones for the project.

Detailed Task Steps:

1. Assign Role:
    - Action: Navigate to the team management section.
    - Action: Assign a role to the new team member (e.g., Frontend Developer).
2. Set Goals and Milestones:
    - Action: Define a goal for the team (e.g., Develop the frontend layout by the end of the first sprint).
    - Action: Set specific milestones to track the team's progress towards that goal (e.g., Submit wireframes by Day 10, Complete initial layout by Day 20).

Task Scenario (for Skilled Individuals): As a new team member, view your assigned role and team goals and milestones.

Detailed Task Steps:

1. View your team and Assigned Role:
   Action: Log in as a team member and view your assigned role and the team goals and milestones set by the startup founder.

Survey

Now that you have completed all tasks, please fill out the survey using the link that is provided to you. This survey will gather your feedback on your overall experience with the platform, including ease of use, satisfaction with features, and any additional comments or suggestions for improvement. Thank you.

Appendix O: Usability Test Survey

Usability Test Survey

Thank you for participating in the usability test for the Technology-Based Startup Team Organization Platform. Your feedback is crucial for improving the platform. Please take a few minutes to complete this survey, which will help us better understand your experience.

The survey includes both rating-scale questions and open-ended questions to capture your overall impressions.

Section 1: General Usability

1. How easy was it to navigate the platform?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

2. How intuitive did you find the job posting and application process?
   Very Confusing (1) | 2 | 3 | 4 | 5 | Very Intuitive (5)

3. How intuitive did you find the team management process (e.g., creating teams, adding members, assigning roles)?
   Very Confusing (1) | 2 | 3 | 4 | 5 | Very Intuitive (5)

4. Did you encounter any technical issues (e.g., errors, bugs) during the usability test?
   Yes | No

   If yes, please describe the issue:

5. How satisfied were you with the platform's response time (loading speed, etc.)?
   Very Unsatisfied (1) | 2 | 3 | 4 | 5 | Very Satisfied (5)

Section 2: Specific Features

Task 1: Posting a Job and Adding Talented Individuals to a Team

1. How easy was it to post a job as a startup founder?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

2. Were you able to find and apply for the posted job as an individual without difficulties?
   Yes | No

   If no, please explain:

3. How clear and effective was the candidate review process for selecting team members?
   Not Clear (1) | 2 | 3 | 4 | 5 | Very Clear (5)

4. What challenges, if any, did you face while posting a job or reviewing candidates?
5. What challenges, if any, did you face while applying for the job as an individual?

Task 2: Portfolio Showcase and Its Impact on Recruitment Decisions

1. How easy was it to update or add items to your portfolio?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

2. As a startup founder, how useful was the portfolio feature in making hiring decisions?
   Not Useful (1) | 2 | 3 | 4 | 5 | Very Useful (5)

3. What additional portfolio features would be useful for hiring decisions?

Task 3: Matching Suitable Candidates with Job Postings

1. How accurate were the job matches based on your skills and portfolio?
   Not Accurate (1) | 2 | 3 | 4 | 5 | Very Accurate (5)

2. Did updating your portfolio improve the accuracy of job matches?
   Yes | No

   If yes, please explain how:

3. As a startup founder, how accurate were the candidate recommendations for your job posting?
   Not Accurate (1) | 2 | 3 | 4 | 5 | Very Accurate (5)

4. What aspects of the matching algorithm did you find most useful, and what would you improve?

Task 4: Defining Roles, Goals, and Milestones

1. How easy was it to create a new team on the platform?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

2. How easy was it to add new members to your team?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

3. How easy was it to assign roles to team members?
   Very Difficult (1) | 2 | 3 | 4 | 5 | Very Easy (5)

4. How clearly did the platform present the assigned roles to team members?
   Not Clear (1) | 2 | 3 | 4 | 5 | Very Clear (5)

5. How easy was it to set team-wide goals and milestones?
   Not Easy (1) | 2 | 3 | 4 | 5 | Very Easy (5)

6. How clearly did the platform present the goals and milestones to team members?
   Not Clear (1) | 2 | 3 | 4 | 5 | Very Clear (5)

7. How useful were the team-wide goals and milestones for managing overall team progress?
   Not Useful (1) | 2 | 3 | 4 | 5 | Very Useful (5)

8. What challenges, if any, did you face while adding new members, assigning roles, or setting goals and milestones for the team?
9. What aspects of team management did you find most useful, and what improvements would you suggest?

Section 3: Overall Experience

1. How satisfied are you with the platform as a whole?
   Very Unsatisfied (1) | 2 | 3 | 4 | 5 | Very Satisfied (5)

2. Would you recommend this platform to others in the startup community?
   Definitely Not (1) | 2 | 3 | 4 | 5 | Definitely (5)

3. What did you like the most about the platform?
4. What would you improve in the platform?

Section 4: Additional Feedback

Please provide any additional comments or suggestions for improving the platform.

Thank you again for your participation! Your insights are invaluable in shaping the future of this platform.

Appendix P: Detailed Usability Testing Observations

P.1 Task 1: Job Posting and Application Process

Participant	Theme	Observations
Participant 1	Navigation and Discoverability	• Opened “Job Listings” instead of “Manage Jobs”; expected to manage jobs from job listings. • Confused “Job Applications” with “Job Listings”. • Experienced confusion when searching for job advertisement creation; eventually found “Manage Jobs”. • Did not understand the “View My Job Postings” button on the panel; did not find it intuitive. • Initially did not know how to edit job postings; found the “Manage Jobs” section to edit later. • When viewing job advertisements from the job listings section, expected to see “Job Details” before “Startup Information”.
	Form Design, Data Entry, and User Preferences	• Entered team size as a range (e.g., 1-10) instead of a single number. • Wanted multiple-choice options for “Key Technologies” in the job advertisement form. • Suggested specifying the number of days in office for the “Hybrid” job type. • Desired to fill out startup information only once, not for each job advertisement; expected startup details to be in the profile. • Experienced confusion regarding two different “Key Technologies” sections—one for the startup and one for the job posting.
	User Interface Issues	• Did not understand the “View My Job Postings” button on the panel. • Expected the “Job Details” section to appear before “Startup Information” when viewing a job advertisement.
	Workflow and Team Management	• Attempted to add team members without creating roles first; could not add members because no roles were defined. • Realized the necessity of defining roles before adding members. • Suggested that roles should precede members in the team view and the member addition process.

		• Expected job applications to be displayed in a table format with detailed applicant information, including a "View Application" option beside the edit button. • Found the team section themselves and created the team. • Initially thought they could add members while creating the team; expected to link or add applicants/members on the team creation page. • Accepted the job application as a founder and logged in as an individual; understood the flow of accepting job applications. • Found the "View Profile" option independently from the members table.
	Additional Features and Enhancements	• Inquired about adding hiring person details to job postings; suggested showing founder or hiring manager details on the job advertisement page. • Mentioned that company or startup profile should be available in the profile section.
	Error Handling and Feedback	• Encountered an error when uploading a file in an unexpected format; tried uploading a .doc file, which was also unsupported.
	Positive Feedback	• Appreciated the message indicating that the job had already been applied for. • Found the job application detail page on the skilled individual panel satisfactory, with more details including founder information (which was missing from the job advertisement).
Participant 2	Navigation and Discoverability	• Opened "Manage Jobs" correctly without confusion. • Found "Job Listings" and "Job Applications" without confusion from the skilled individual panel but confused them in the founder panel. • Faced confusion with unsupported CV formats during upload when applying for the job as an individual.
	Form Design, Data Entry, and User Preferences	• Expected dropdown for the "Industry" field of the job advertisement form. • Preferred bullet points for the "Required Skills" field in job advertisement form. • Appreciated the date picker for the "Application Deadline" in the job advertisement form.
	User Interface Issues	• Mistook non-clickable blue text ("CV", "Cover Letter") for clickable links; thought they were links, but they were not.

	Workflow and Team Management	• Tried adding team members without accepting applications or creating roles first. • Initially confused as to why the applicant was unavailable to add as a team member. • Understood the need to accept job applicants and define roles before adding team members after initial confusion.
	Additional Features and Enhancements	• Suggested clearly labelling acceptable file formats for uploads to avoid confusion.
	Error Handling and Feedback	• Faced confusion with unsupported CV formats during upload.
	Positive Feedback	• Valued the minimum details required for job applications; found it beneficial and user-friendly. • Appreciated the message indicating that the job had already been applied for.
Participant 3	Navigation and Discoverability	• Opened "Job Listings" instead of "Manage Jobs". • Confused "Job Applications" with the "Job Listings" section. • Expected to see a "Teams" button on the dashboard for easier access.
	Form Design, Data Entry, and User Preferences	• Corrected an invalid founding year entry after a prompt; the form provided appropriate feedback.
	User Interface Issues	• No specific UI issues were noted; navigation became smoother after initial familiarization.
	Workflow and Team Management	• Initially confused about being unable to add a new member to the team. • Understood that job applications needed to be accepted before adding a member. • After accepting job applications, tried to add members to the team but faced confusion due to not having predefined roles. • Understood the need to create roles and added the new team member afterward.
	Additional Features and Enhancements	• N/A
	Error Handling and Feedback	• No specific errors were noted; the system provided appropriate prompts when necessary.
	Positive Feedback	• Navigated job management and created job postings with minimal issues.

		• Found the job application view comprehensive when accessed from the job listings section. • Successfully viewed the candidate's profile from the job application.
Participant 4	Navigation and Discoverability	• Opened "Manage Jobs" correctly initially. • Confused "Manage Jobs" with "Job Listings" later on. • Mistook "Job Applications" for "Job Listings" initially but corrected the mistake. • Expected to create the startup separately; ideally during account setup.
	Form Design, Data Entry, and User Preferences	• Believed the platform should enforce the expected sequence for startup founders.
	User Interface Issues	• No specific UI issues were noted; navigation improved after initial familiarization.
	Workflow and Team Management	• Attempted to add members without creating roles first; faced confusion when the applicant was not available to add. • Understood that accepting the job applicant and defining roles were necessary before adding members. • Understood the need to accept offers as individuals and proceeded accordingly. • Expected a forced sequence for team creation steps (e.g., create team, then roles, then job posts based on roles, then hire to roles); suggested that these steps should be required and guided by the system to prevent confusion.
	Additional Features and Enhancements	• Suggested that the platform should enforce the correct process flow for team creation and management to avoid user confusion.
	Error Handling and Feedback	• No specific errors were noted; the system provided appropriate prompts when necessary.
	Positive Feedback	• N/A
Participant 5	Navigation and Discoverability	• Confused between "Job Listings" and "Manage Jobs". • Struggled to find the "Log Out" button.
	Form Design, Data Entry, and User Preferences	• No specific issues were noted; form completion was smooth.

	User Interface Issues	• No specific UI issues were noted; navigation improved after initial familiarization.
	Workflow and Team Management	• Faced challenges adding members; did not realize the need to accept job applicants' offers and create roles first. • Understood that some steps were missed but was not clear which ones. • Encountered difficulties when trying to add team members after already accepted their offer, without predefined roles.
	Additional Features and Enhancements	• Suggested that displaying error messages or prompts when users miss required steps in the team management process would be helpful.
	Error Handling and Feedback	• No specific errors were noted; the system provided appropriate prompts when necessary.
	Positive Feedback	• Appreciated the minimum details required for job applications, making the platform user-friendly. • Found accepting the offer as an individual straightforward.

P.2 Task 2: Portfolio Creation and Application Review

Participant	Theme	Observations
Participant 1	Portfolio Management and Integration	• Faced initial confusion about the meaning of a portfolio item. • Uncertain about adding a portfolio after applying for a job. • Found viewing the portfolio post-application confusing; thought it should be before applying. • Suggested prompts during the application process to guide individuals if they want to upload a portfolio before the job application. • Suggested having a checkbox on the job application form for founders to make the portfolio mandatory if needed.
	User Categorization and Expectations	• Expected to see only company team members, not all platform users, especially from the skilled individual panel. • Expected to see the founder's startup/company information on their profile. • Mentioned that as a founder, it would make sense to view all people on the platform, but not as an individual. • Suggested categorizing users into company or team members and all users or applicants.

	Application Review and Candidate Profile Integration	• Found and accessed the candidate's profile independently from the job application. • Noted that reviewing the portfolio while reviewing applications is sensible; it helps look through the history of the applicant.
Participant 2	Portfolio Management and Integration	• Initially confused the portfolio section with the profile section; found the correct section eventually. • Added a portfolio item without issues.
	User Categorization and Expectations	• Found the "Users" section without difficulties. • Expected to see only people within the company, not all users in the "Users" section. • Suggested categorizing users to differentiate between company members and all platform users.
	Application Review and Candidate Profile Integration	• Understood that the portfolio could be viewed from the applicant's profile during the application review process. • Noted that reviewing the portfolio during application evaluation is sensible and helps in decision-making.
Participant 3	Portfolio Management and Integration	• Created a portfolio item without issues. • Viewed the portfolio from the "team", "Users", and "Job Applications" sections.
	User Categorization and Expectations	• Initially thought "Users" would show all the people in the company or startup team (members only). • Expected "Users" to display company team members, not all platform users.
	Application Review and Candidate Profile Integration	• Successfully viewed portfolio from the team and job application sections. • Understood the need to view portfolios during application evaluation.
Participant 4	Portfolio Management and Integration	• Found the portfolio section quickly. • Completed portfolio creation without any issues.
	User Categorization and Expectations	• Expected "Users" to display company-related individuals (those who interacted with the company via previous job applications and current members). • Not interested in seeing users who have no interaction with their company or job postings.
	Application Review and Candidate Profile Integration	• Understood the need to view portfolios during application evaluation.

		• Viewed the portfolio from the application section without issues.
Participant 5	Portfolio Management and Integration	• Completed portfolio creation without issues. • Found the overall portfolio creation process user-friendly.
	User Categorization and Expectations	• No specific issues were noted regarding user categorization.
	Application Review and Candidate Profile Integration	• Faced challenges locating where to view the portfolio from the application section. • Suggested enhancing the visibility of the portfolio link within job applications to streamline the review process. • Understood the need to view portfolios during application evaluation.

P.3 Task 3: Interacting with the Matching Algorithm

Participant	Theme	Observations
Participant 1	Understanding Matching	• Found the "Job Matches" section in the individual panel. • Recognized that job matches are based on the skilled individual's portfolio. • Changed their portfolio and reviewed the changes; satisfied with the results. • Noticed the matching score independently. • Updated their portfolio again and observed the impact on matching results; understood the importance of portfolio content on match quality.
	Matching Criteria Preferences	• Desired the ability to choose between considering CV, portfolio, or both for matching. • Appreciated the inclusion of CV details in the matching process. • Wanted to specify which fields to consider for matching (e.g., skills, years of experience). • Suggested having preferences to choose which criteria (portfolio, CV, etc.) to consider and to set their priority. • Suggested that the system could select all options by default if preferences are not set.

	User Interface Issues	• Expected to see the "Apply" button on matching results along with "More Details". • Found the job advertisement selection mechanism in the matching feature confusing as a founder; the select box design was unclear. • Expected to see matches immediately after opening the job applications section by default. • Experienced confusion when trying to access matches as a founder; initially navigated to the job advertisements section.
	Matching Accuracy and Relevance	• Suggested transparent indications of how matching scores are calculated and what factors influence them.
	Suggestions for Improvement	• N/A
Participant 2	Understanding Matching	• Found "Job Matches" easily and understood the relationship to their portfolio. • Recognized that updating their portfolio affects job matches. • Expected to match the job posting since the keywords were already in the portfolio.
	Matching Criteria Preferences	• Preferred to see matches immediately without additional steps. • Desired options to select matching criteria (CV, portfolio, or both). • Suggested having preferences for matching criteria to avoid skewed results.
	User Interface Issues	• Understood that applicant matches could be viewed from the job applications section. • Clicked the matching button without choosing the job advertisement; found the select box confusing initially but understood it shortly.
	Matching Accuracy and Relevance	• Confused about why adding similar technologies to the portfolio did not improve matches as expected. • Worried that defaulting algorithm to use CV for matches might skew the results. • Believed considering the CV and portfolio simultaneously is a good decision. • Suggested that if the CV and portfolio have overlaps, the matching score should be stronger.

		• Believed putting more emphasis on skills and key technologies in algorithm preferences would be a good improvement.
	Suggestions for Improvement	• Suggested allowing users to select matching criteria (CV, portfolio, or both) and to select fields granularly. • Recommended ensuring default matching criteria do not skew results, as the CV might be lengthier than the portfolio and could skew the results.
Participant 3	Understanding Matching	• Found "Job Matches" as an individual easily. • Understood the relationship between job matches and their portfolio. • Updated the portfolio with an irrelevant item to test the matching algorithm and noticed a drop in matching scores.
	Matching Criteria Preferences	• Wanted options to specify which criteria (CV, portfolio) to prioritize in matching rather than using both by default. • Suggested that individuals should be able to upload CVs separately, irrespective of their job application, to see matches.
	User Interface Issues	• Found the job matches section from the job applications in the founder panel confusing; expected matches to be more accessible. • Confused about how the select box works (thought it was a text box) in the job applications matching section; understood it shortly. • Suggested it would be better to see applicant matches immediately upon accessing job applications by default, without the need for a select box.
	Matching Accuracy and Relevance	• Did not understand that job applications also consider CV results.
	Suggestions for Improvement	• Suggested it would be better to see applicant matches immediately upon accessing job applications by default, without the need for a select box. • Recommended providing options to select matching preferences—whether to include CV, portfolio, or both.
Participant 4	Understanding Matching	• Understood the consideration of portfolio in matching results. • Recognized the need to update the portfolio for better job matches.

	Matching Criteria Preferences	• Preferred default matching based on either CV or portfolio (not both); if either match, it should appear in the results. • Expected user-controlled preference settings for matching criteria. • Mentioned a desire to see candidates who might have relevant CVs but portfolios that are irrelevant to the job posting yet valuable as experience. • Believed that having the option to prioritize matching criteria would be beneficial.
	User Interface Issues	• Navigated to "Job Matches" from the individual panel without issues. • Found it confusing to navigate applicant matches from job applications in the founder panel. • Experienced confusion with the selection mechanism for job advertisements in the matching feature; shortly after, understood the select box.
	Matching Accuracy and Relevance	• Believed that if either the CV or portfolio matches, the applicant should appear in the results to view candidates who might have relevant CVs but also have portfolios that, while irrelevant to the job posting, are valuable as experience.
	Suggestions for Improvement	• Suggested allowing users to prioritize matching criteria (CV, portfolio, or both) to improve relevance. • Recommended ensuring default matching settings align with user preferences. • Advised making the matching feature more intuitive and user-friendly.
Participant 5	Understanding Matching	• Recognized the need to update the portfolio for better job matches. • Updated the portfolio and viewed the results.
	Matching Criteria Preferences	• Believed CV might be more reliable than the portfolio for matching. • Desired options to prioritize CV over portfolio in the matching algorithm.
	User Interface Issues	• Found job matches easily from the individual panel. • Confused about the select box in the application matching section.

		• Found matches from job applicants as the founder after some confusion.
	Matching Accuracy and Relevance	• Initially did not understand that job applicant matches consider CV results when viewing job applicants from the founder panel. • Expected clearer indications of the factors influencing matching results and desired the matching system to be transparent.
	Suggestions for Improvement	• Suggested providing clearer indications of factors influencing matching results. • Advised enhancing the user interface to make matching features more intuitive.

P.4 Task 4: Roles, Goals, and Milestones Setup

Participant	Theme	Observations
Participant 1	Goal and Milestone Management	• Goals and milestones were created without any issues. • The process of breaking down goals into milestones was understood. • Suggested that tasks should further break down goals into assignable units for team members. • Mentioned that tasks have statuses or progress indicators and assist in achieving goals, whereas goals should not have statuses. • Expected that goals are meant for management and tasks are intended for team members. • No issues with the goal and milestone hierarchy; suggested adding a breakdown of goals or milestones into tasks. • Indicated that incorporating a task management feature to track progress would be beneficial.
	Roles Assignment	• Expected a restriction of only one role per member. • Suggested that roles should be defined before adding members (in the team setup process).
	User Interface and Navigation	• The process was found to be straightforward; no navigation issues were noted. • The “View Profile” option from the members’ table was found to be useful.

	Suggestions for Improvement	• Suggested integrating task management features to facilitate progress tracking and ensure goals and milestones are met. • Mentioned that status tracking for tasks (not goals) would improve project management within the platform.
Participant 2	Goal and Milestone Management	• Goals and milestones were created without any issues. • The concept of breaking down goals into milestones was understood. • Expected that accountability could be assigned to both goals and milestones, with individuals responsible for each. • Mentioned that tasks could be assigned to team members under milestones. • No issues with the goal and milestone hierarchy.
	Roles Assignment	• No specific issues were noted; roles were assigned appropriately.
	User Interface and Navigation	• The process was found to be straightforward; no navigation issues were noted.
	Suggestions for Improvement	• Desired the ability to assign goals to managers and milestones to team members for better role distribution. • Suggested integrating accountability measures to ensure team members are responsible for assigned milestones. • Emphasized the importance of incorporating reporting and commenting features within the goals and milestones setup, especially in hybrid team settings. • Mentioned that communication features (e.g., sending reports/ or comments on milestones and goals) are important for progress tracking.
Participant 3	Goal and Milestone Management	• Goals and milestones were created without any issues. • Wanted to be able to add milestones separately or break down goals into milestones. • After initial confusion, the connection between goals and milestones was understood independently. • No issues with the goal and milestone hierarchy.
	Roles Assignment	• Wanted to assign multiple roles to a single team member to reflect multiple responsibilities but did not understand the correct way of doing so.
	User Interface and Navigation	• The process was found to be straightforward; no navigation issues were noted.

	Suggestions for Improvement	• Suggested integrating a feature to facilitate progress tracking and ensure goals and milestones are met. • Emphasized that task management would be a good improvement for tracking progress.
Participant 4	Goal and Milestone Management	• Goals and milestones were created without any issues. • Expected milestones to have goals inside them, suggesting an inverse hierarchy compared to the current design. • Suggested implementing a more granular structure to better manage complex projects and team responsibilities. • Proposed a detailed hierarchy encompassing milestones > goals > objectives > tasks, with progress indicators at each level to enhance tracking and accountability. • Mentioned that tasks will have statuses (completed or not completed), contributing to the percentage completion of objectives, goals, and milestones.
	Roles Assignment	• Desired the ability to assign multiple roles to a single team member but did not understand that it was currently possible.
	User Interface and Navigation	• The process was found to be straightforward; no navigation issues were noted. • Signed in as an individual and viewed their team without issues.
	Suggestions for Improvement	• Suggested that milestones should contain goals, which contain objectives, achieved through tasks. • Emphasized the need for progress indicators (percentages) at each level to enhance tracking and accountability. • Believed that this hierarchical structure would help in managing complex projects effectively.
Participant 5	Goal and Milestone Management	• Goals and milestones were created without any issues. • The process of breaking down goals into milestones was understood. • No issues with the goal and milestone hierarchy.
	Roles Assignment	• No specific issues were noted; roles were assigned appropriately.
	User Interface and Navigation	• From the skilled individual panel, there was some confusion in finding teams. • Expected the “Teams” button on the dashboard for easier navigation. • Overall, the process was found to be user-friendly and straightforward after initial familiarization.

	Suggestions for Improvement	• Emphasized the importance of tracking progress to ensure goals and milestones are met. • Recommended adding progress management features to effectively track progress on goals and milestones. • Suggested that task assignment would be a good improvement for tracking progress.

Appendix Q: Detailed Quantitative Survey Results

Q.1 General Usability

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating
How easy was it to navigate the platform?	5	5	4	4	4	4.4
How intuitive did you find the job posting and application process?	4	5	5	4	5	4.6
How intuitive did you find the team management process?	5	5	4	4	5	4.6
How satisfied were you with the platform's response time?	5	5	5	3	4	4.4

Q.2 Technical Stability

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Did you encounter any technical issues during the usability test?	No	No	No	No	No

Q.3 Job Posting, Applications, and Team Member Addition (Task 1)

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating
How easy was it to post a job	5	5	4	5	4	4.6

as a startup founder?						
How clear and effective was the candidate review process for selecting team members?	4	5	4	5	5	4.6

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Were you able to find and apply for the posted job as an individual without difficulties?	Yes	Yes	Yes	Yes	Yes

Q.4 Portfolio Management and Recruitment Impact (Task 2)

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating
How easy was it to update or add items to your portfolio?	5	5	4	5	5	4.8
As a startup founder, how useful was the portfolio feature in making hiring decisions?	5	5	4	4	4	4.4

Q.5 Matching Algorithm (Task 3)

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating

How accurate were the job matches based on your skills and portfolio?	5	5	4	4	5	4.6
As a startup founder, how accurate were the candidate recommendations for your job posting?	5	5	4	4	5	4.6

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Did updating your portfolio improve the accuracy of job matches?	Yes	Yes	Yes	Yes	Yes

Q.6 Roles, Goals, and Milestones Management (Task 4)

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating
How easy was it to create a new team on the platform?	5	5	5	5	5	5
How easy was it to add new members to your team?	5	5	4	5	5	4.8
How easy was it to assign roles to team members?	5	5	4	5	5	4.8
How clearly did the platform present the	5	5	4	5	5	4.8

assigned roles to team members?						
How easy was it to set team-wide goals and milestones?	5	5	4	4	4	4.4
How clearly did the platform present the goals and milestones to team members?	5	5	4	3	5	4.4
How useful were the team-wide goals and milestones for managing overall team progress?	5	5	4	4	4	4.4

Q.7 Overall Satisfaction and Recommendations

Survey Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Average Rating
How satisfied are you with the platform as a whole?	5	5	4	4	4	4.4
Would you recommend this platform to others in the startup community?	5	5	4	4	5	4.6

Appendix R: Detailed Qualitative Survey Results

R.1 Challenges in Job Posting and Candidate Review (Task 1)

Participant	Response
Participant 1	• While posting a job, the requirement to fill in company details manually was noted. It would be more efficient for the platform to automatically retrieve this information from the company profile.
Participant 2	• N/A
Participant 3	• When clicking on a candidate's job application, the job description opens first, necessitating scrolling down to access the candidate's profile. Directly opening the candidate's profile would be preferable.
Participant 4	• The integration of recruitment and team management could benefit from guiding founders in forming suitable teams and roles before posting a job, ensuring the validity of the role and streamlining the recruitment and shortlisting process.
Participant 5	• N/A

R.2 Job Application Challenges (Task 1)

Participant	Response
Participant 1	• No challenges were reported.
Participant 2	• No challenges were reported.
Participant 3	• No challenges were reported.
Participant 4	• No challenges were reported.
Participant 5	• No challenges were reported.

R.3 Portfolio Feature Suggestions (Task 2)

Participant	Response
Participant 1	• Making portfolio details mandatory for applicants would streamline the application review process.
Participant 2	• N/A
Participant 3	• Adding a cover letter option to the portfolio would enhance the application process.
Participant 4	• Guiding candidates to create numerous portfolio items would be useful. Diverse examples would be appreciated, even if not directly related to job postings.
Participant 5	• N/A

R.4 Matching Algorithm Reflecting Portfolio Updates (Task 3)

Participant	Response
Participant 1	• Updating the portfolio improved job matching recommendations provided by the platform.

Participant 2	• Reducing verbosity in the portfolio descriptions, which were previously considered unmatched by the algorithm, resulted in better matches with job postings. • It would be beneficial if the portfolio matching algorithm prioritized relevant matches for job postings.
Participant 3	• Adding a new portfolio item that was irrelevant to the job advertisement caused a slight drop in the match score.
Participant 4	• Including relevant technologies in the portfolio improved alignment with the job posting.
Participant 5	• Updating the portfolio allowed for better alignment of current skills, experience, and projects with relevant job openings. Recent work and technologies made job recommendations more tailored to expertise and career aspirations.

R.5 Matching Algorithm Usefulness and Improvements (Task 3)

Participant	Response
Participant 1	• The matching algorithm benefited both companies and applicants by scoring portfolio skills, aiding in the review process.
Participant 2	• The accuracy of keyword matching was impressive, enhancing clarity and significantly improving the hiring process.
Participant 3	• The automatic recognition of portfolio items is useful, but making consideration of CV information optional would be preferable.
Participant 4	• Automatically reading CVs is a valuable feature. However, it would be better to see all users with either matching CVs or portfolio items by default, with options to filter for one or the other as needed.
Participant 5	• Incorporating more personalized recommendations for the matching algorithm based on user behaviour could be beneficial.

R.6 Team Management Challenges (Task 4)

Participant	Response
Participant 1	• No challenges were encountered. The user interface facilitated easy management of the team. • Reordering the steps for adding a team member to first select the role would enhance the workflow.
Participant 2	• No challenges were reported.
Participant 3	• No challenges were reported.
Participant 4	• The hierarchy of goals and milestones should be structured as follows: company-wide milestones at the top (1-2 year plans), departmental goals underneath (3-6 month plans), team objectives below that (2-6 week plans), and tasks at the bottom (1-5 day plans).
Participant 5	• No challenges were reported.

R.7 Team Management Usefulness and Improvements (Task 4)

Participant	Response
Participant 1	• Having all team management features in one location simplifies organizational processes, including hiring, management, and career/KPI tracking.
Participant 2	• The team management features significantly enhance the platform's primary function, enabling applicants to feel part of the organization from the start. • Communicating team goals and milestones upon accepting a job offer allows new hires to start strong. • Improvements focusing on accountability and communication around goals and milestones would be beneficial.
Participant 3	• The ability to assign multiple roles to a single individual is very useful.
Participant 4	• The process of assigning roles is straightforward, but it should be intuitive. • Adding captions under the titles could clarify that "each milestone has multiple goals."
Participant 5	• Clear task delegation and regular communication can keep the team aligned and accountable. • Incorporating more collaborative tools and increasing feedback loops for continuous improvement would be advantageous.

R.8 Additional Comments and Suggestions

R.8.1 Top Platform Features Liked

Participant	Response
Participant 1	• The platform's simplicity, with everything in one place for team management, is commendable.
Participant 2	• The concept of making applicants feel part of the organization from the outset is key.
Participant 3	• The interface is simple and user-friendly.
Participant 4	• The integration of recruitment with team management is a great idea. Many companies already use team management systems, so merging these functionalities would be essential.
Participant 5	• The intuitive user interface and smooth navigation made the platform easy to use effectively.

R.8.2 Platform Improvement Suggestions

Participant	Response
Participant 1	• N/A
Participant 2	• Enhancing accountability among team members regarding goals and milestones is important. • Assigning ownership to specific goals/milestones would be particularly effective.
Participant 3	• Adding "tasks" to the team management functions would be a valuable addition.

Participant 4	• Standardizing naming conventions for milestones and goals would be helpful. • Better guidance on the order of steps for both founders and candidates would also be beneficial.
Participant 5	• More advanced filtering options for job searches and improving page loading speeds would enhance user experience. • Incorporating personalized recommendations (in the matching algorithm) based on user behavior would be beneficial. • Integrating task management and deadlines into the platform would be appreciated.

R.8.3 Additional Comments/Suggestions

Participant	Response
Participant 1	• The integration of team management and recruitment on a single platform is a good strategy for startups.
Participant 2	• The matching algorithm will assist in managing the influx of applications for job postings, helping hiring teams feel less overwhelmed and select candidates more clearly and concisely.
Participant 3	• The integration of team management and recruitment on a single platform is a good strategy for startups.
Participant 4	• Everything is great; A more comprehensive version of this platform would be highly anticipated.
Participant 5	• Overall, the platform is very user-friendly. • A mobile app for easier access and notifications would enhance engagement.