CROWDFUNDING PLATFORM FOR ENTREPRENEURS AND INNOVATORS WITH TRANSPARENT WALLET-BASED TRANSACTIONS

A PROJECT REPORT

Submitted by

SANJANA B	**211521205133**
SARA PRICILLA A	**211521205138**
SWETHA E V	**211521205163**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

PANIMALAR INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY : CHENNAI 600 025

MAY 2025

PANIMALAR INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**CROWDFUNDING PLATFORM FOR ENTREPRENEURS AND INNOVATORS WITH TRANSPARENT WALLET-BASED TRANSACTIONS"** is the bonafide work of "**SANJANA B (211521205133), SARA PRICILLA A (211521205138), SWETHA E V (211521205163)**" who carried out the project work under my supervision.

SIGNATURE

Dr. G. DHANALAKSHMI, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Information Technology,

Panimalar Institute of Technology

Poonamallee, Chennai 600123

SIGNATURE

Mrs. M. RAMYA, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of Information Technology,

Panimalar Institute of Technology

Poonamallee, Chennai 600123

Certified that the candidates were examined in the university project viva-voce held on _____ at Panimalar Institute of Technology, Chennai 600 123.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We seek the blessings from the **Founder** of our institution **Dr. JEPPIAAR, M.A., Ph.D.,** for having been a role model who has been our source of inspiration behind our success in education in his premier institution.

We would like to express our deep gratitude to our beloved **Secretary and Correspondent Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation which inspired us a lot in completing this project.

We also express our sincere thanks and gratitude to our dynamic **Directors Mrs. C. VIJAYA RAJESHWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYA SREE SAKTHI KUMAR, B.E, M.B.A., Ph.D.,** for providing us with necessary facilities for completion of this project.

We also express our appreciation and gratefulness to our respected **Principal Dr. T. JAYANTHY, M.E., Ph.D.,** who helped us in the completion of the project.

We wish to convey our thanks and gratitude to our **Head of the Department, Dr. G. DHANALAKSHMI, M.E., Ph.D.,** for her full support by providing ample time to complete our project.

We express our indebtedness and special thanks to our **Supervisor, Mrs. M. RAMYA, M.E.,** for her expert advice and valuable information and guidance throughout the completion of the project.

Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

The global startup ecosystem thrives on collaboration between innovators and funders, yet inefficiencies persist in matching inventors with the right investors. This project introduces a digital platform that merges crowdfunding with joint venture capitalism, enabling multiple investors to collectively fund groundbreaking inventions. The system employs data-driven algorithms to connect stakeholders based on project viability, investment preferences, and risk appetite. Key features include automated onboarding, intelligent matchmaking, secure in-app negotiations, and a transparent rating system to foster trust.

The platform analyzes critical variables such as project stage, industry sector, funding goals, inventor track records, and investor portfolios to optimize connections. Advanced matching algorithms leverage collaborative filtering and predictive analytics to generate personalized recommendations. Prototype testing demonstrates real-world applicability through measurable outcomes including match success rates, funding velocity, and user satisfaction metrics. The solution streamlines capital allocation while democratizing access to innovation funding, particularly benefiting early-stage ventures. By bridging decentralized crowdfunding with structured joint ventures, this platform creates an efficient marketplace for innovation investment. Its scalable architecture and data-centric approach offer valuable insights for evolving collaborative investment models, setting a new standard for connecting ideas with resources in the digital age.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	**ii**
	LIST OF FIGURES	**vii**
	LIST OF SYMBOLS	**ix**
1	**INTRODUCTION**	**1**
1.1	OVERVIEW OF THE PROJECT	1
1.2	SCOPE OF THE PROJECT	2
2	**LITERATURE SURVEY**	**3**
2.1	INTRODUCTION	3
2.2	LITERATURE SURVEY	3
2.3	CONCLUSION	10
3	**SYSTEM ANALYSIS**	**12**
3.1	EXISTING SYSTEM	12
3.1.1	PROBLEM DEFINITION	12
3.2	PROPOSED SYSTEM	12
3.2.1	ADVANTAGES	13
4	**REQUIREMENT SPECIFICATION**	**14**
4.1	INTRODUCTION	14
4.2	HARDWARE AND SOFTWARE	14
4.2.1	HARDWARE REQUIREMENTS	14
4.2.2	SOFTWARE REQUIREMENTS	15
4.2.2.1	JAVASCRIPT	15
4.2.2.2	INTRODUCTION	16

5		**SYSTEM DESIGN**	**21**
5.1		ARCHITECTURE DIAGRAM	21
5.2		UML DIAGRAMS	22
	5.2.1	USECASE DIAGRAM	22
	5.2.2	SEQUENCE DIAGRAM	23
	5.2.3	CLASS DIAGRAM	24
	5.2.4	ACTIVITY DIAGRAM	25
	5.2.5	DATA FLOW DIAGRAM	26
6		**WORKFLOW**	**27**
6.1		REQUIREMENT ANALYSIS	27
6.2		PROJECT SETUP AND DESIGN	28
6.3		DEVELOPMENT PHASE	29
6.4		TESTING AND DEBUGGING	30
6.5		DEPLOYMENT AND FINAL REVIEW	32
7		**USER & INVESTOR LOGIN**	**33**
7.1		USER LOGIN SYSTEM – FULL WORKFLOW	33
7.2		LOGIN PROCESS	34
	7.2.1	LOGIN INTERFACE PRESENTATION	34
	7.2.2	FORM VALIDATION	35
	7.2.3	SERVER-SIDE AUTHENTICATION	35
	7.2.4	TOKEN & ROLE STORAGE	36
	7.2.5	PERSISTENT SESSION MANAGEMENT	37
7.3		UX/UI DETAILS	38
7.4		SECURITY MEASURES	38
7.5		INVESTOR LOGIN: DESIGN PRINCIPLES	40

7.6 SUMMARY 41

8 SMTP SERVER 43

8.1 WHAT IS SMTP? 43

8.1.1 KEY COMPONENTS OF THE SMTP SYSTEM 43

8.1.2 EMAIL PROVIDERS 43

8.1.3 SMTP SERVER CONFIGURATION 44

8.2 FUNCTIONS OF THE SMTP 46

8.2.1 USER REGISTRATION AND EMAIL VERIFICATION 46

8.2.2 PASSWORD RESET REQUESTS 46

8.2.3 SYSTEM NOTIFICATIONS 47

8.2.4 TRANSACTIONAL EMAILS 47

8.2.5 GENERAL COMMUNICATION AND MARKETING EMAILS 48

8.3 SECURITY CONSIDERATIONS 48

8.3.1 EMAIL AUTHENTICATION 48

8.3.2 ENCRYPTION 48

8.3.3 RATE LIMITING AND THROTTLING 49

8.3.4 SENSITIVE DATA HANDLING 49

8.4 CONCLUSION 50

9 TECH STACK 51

9.1 FRONTEND 51

9.1.1 API GATEWAY LAYER 51

9.2 BACKEND 52

9.3 DATA LAYER 53

10 CONCLUSION AND FUTURE SCOPE 55

10.1 CONCLUSION 55

10.2 FUTURE SCOPE 56

11 APPENDICES 57

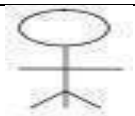
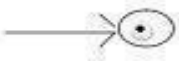
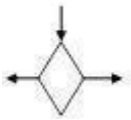
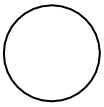
12 REFERENCES 87

LIST OF FIGURES

S.NO	NAME OF THE FIGURES	PAGE. NO
1	System Architecture Diagram	23
2	Use Case Diagram	24
3	Sequence Diagram	25
4	Class Diagram	26
5	Activity Diagram	27
6	Work Flow Diagram	28
7	Landing Page	83
8	About Page	83
9	Dashboard	84
10	Connection Search Page	84
11	User Profile Popup	85

12	View Funds Page	85
13	Inventions Dashboard	86
14	Investment Analytics	86

LIST OF SYMBOLS

S.NO	NAME	NOTATION	DESCRIPTION
1.	Actor		It aggregates several classes into single classes
3.	State		State of the process.
4.	Initial State		Initial state of the object
5.	Final state		Final state of the object
6.	Control flow	X	Represents various control flow between the states.
7.	Decision box		Represents decision making process from a constraint
9.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
10.	External entity		Represents external entities such as keyboard, sensors, etc.
11.	Transition		Represents communication that occurs between processes.

CHAPTER 1

CHAPTER 1
INTRODUCTION

1.1 AN OVERVIEW OF PROJECT

This project is a digital platform designed to connect inventors and investors through an intelligent, data-driven ecosystem. Inventors can showcase their innovations while investors discover promising opportunities within a unified professional network. The system automates onboarding by sending secure login credentials via email, enabling quick access to the platform. Registered users create detailed profiles that highlight their expertise, project development stages, or investment criteria, facilitating accurate matching based on verified credentials and historical data.

The platform's matching system combines advanced database algorithms with behavioral analytics to connect users. Structured queries match participants based on industry specialization, funding requirements, investment history, and geographic preferences. Sophisticated data analysis identifies compatibility patterns by evaluating project viability indicators and investor track records. Real-time messaging with document sharing capabilities enables seamless communication, with all data transmissions protected by enterprise-grade encryption protocols.

By integrating robust matching algorithms with secure collaboration tools, this platform creates an efficient marketplace for innovation funding. The system reduces traditional networking inefficiencies through data-driven connections, ensuring inventors and investors can focus on productive partnerships rather than prolonged searches. From early-stage prototypes to mature ventures, the platform delivers a professional environment where promising ideas meet qualified funding sources.

1.2 SCOPE OF THE PROJECT

This platform creates a secure digital ecosystem connecting inventors with investors through data-driven matchmaking. It features automated onboarding, detailed profile creation with project/investment criteria, and intelligent matching algorithms analyzing compatibility factors like industry sector, funding stage, and risk appetite. Core functionalities include real-time encrypted messaging, document sharing for pitches and proposals, and a dual-rating system for accountability. Users access personalized dashboards with connection analytics and market trends, while administrators monitor platform integrity through activity logs and fraud detection. Initially deployed as a web application with MySQL database and React/Node.js stack, the solution prioritizes scalability, data security, and responsive design. Future enhancements may incorporate mobile access, advanced due diligence tools, and API integrations with financial/legal services. The platform aims to streamline funding connections while maintaining rigorous quality standards across all transactions.

CHAPTER 2

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

Evaluating early-stage scientific inventions presents a critical challenge for organizations aiming to foster innovation while managing significant uncertainty. Traditional evaluation criteria often raise doubts about commercial feasibility, leading to the potential premature dismissal of high-potential inventions. To overcome this, evaluators frequently make resource commitments based on a leap of faith, balancing feasibility and desirability factors despite incomplete information. This paper examines how evaluators assess and decide to support early-stage inventions, highlighting the complexities and strategic considerations involved. The survey reviews existing evaluation approaches, identifies key decision drivers, and discusses future opportunities for improving resource allocation in technology transfer and innovation management.

2.2 LITERATURE SURVEY

Bai et al. **(2021)** examined the collaboration between governments and private capital investors in financing early-stage ventures. Their study, titled *"Public Entrepreneurial Finance Around the Globe"*, investigates the conditions under which such partnerships are likely to form. Using a conceptual framework supported by hand-collected data from 755 programs globally, the researchers found that government programs often involve private investors, particularly in regions with effective governance and well-developed venture capital markets. The collaborations were most prevalent in programs targeting early-stage companies and were typically implemented through joint equity investments and matching-fund mechanisms. The study emphasizes the strategic role of government involvement in fostering innovation ecosystems and highlights the importance of aligning public initiatives with local private capital infrastructure to enhance entrepreneurial financing outcomes.

Xiao et al. **(2025)** explored how the investor–entrepreneur relationship can be effectively managed and sustained in the post-financing stage, addressing the frequent breakdown of such collaborations. Their study, published in the *Journal of Management*

Studies and titled *"Institutional Logics and the Post-Financing Relationship Between Investors and Entrepreneurs"*, introduces an institutional logics perspective to better understand the dynamics involved. Unlike previous research that treats investor and entrepreneur behavior as uniform, this study highlights the importance of differing norms, values, and motivations on both sides. It proposes that the success of post-financing relationships hinges on logic similarity—shared understandings shaped by investment and contextual factors. An empirical analysis of 2052 deals involving 1032 investors and 382 high-tech portfolio firms in China from 2000 to 2018 supports the proposed framework. The findings emphasize the significance of aligning institutional logics to foster mutual understanding and collaboration between investors and entrepreneurs in high-risk, innovation-driven environments.

Smit (2024) investigated the evolving needs of startups and investors within entrepreneurial ecosystems through a case study titled *"Designing a Funding Platform for Startup and Investor Evolving Needs in the Entrepreneurial Ecosystem: The Orange Mill"*. The study addresses the persistent challenge faced by startups in securing capital and the dual role investors play—offering both financial and strategic support. Using a qualitative exploratory approach, including semi-structured interviews and direct observations, the research reveals that early-stage startups seek financial backing, social capital, and mentorship, while investors prioritize specific criteria, such as founder traits, market opportunities, and communication skills. The study distinguishes between generalist novice investors and specialist experienced investors, highlighting differences in their investment roles. Trust, involvement, and deal-making processes emerge as key factors influencing the entrepreneur–investor relationship. The research proposes a technology-driven funding platform that facilitates resource exchange, networking, and investor education, especially targeting novice investors. The platform is also envisioned to act as a specialist intermediary under AFM-light regulations, supporting direct investments. Despite facing internal challenges like limited experience and workforce instability, the case firm exemplifies how platform innovation, coupled with strategic identity realignment, can bridge gaps in startup financing. The study contributes both theoretically and practically to understanding and enhancing startup–investor alignment, with recommendations for future research.

Liang (2024) examined how open source communities, particularly GitHub, influence the financing outcomes of diverse entrepreneurs in the technology sector. The study, titled *"The*

Impact of Open Source Communities on Financing Dynamics of Diverse Entrepreneurs" , applies signaling theory to assess whether open source engagement can serve as a credible indicator of technological competence. Using a dataset of over 10,000 U.S.-based startups in AI, machine learning, and software—sourced from Crunchbase and GitHub—the research found a positive correlation between GitHub activity and funding success. This effect was especially pronounced for women and minority founders, who are traditionally underrepresented in venture capital financing. The findings suggest that open source contributions enhance visibility and perceived venture quality, thereby helping to reduce funding disparities. The study underscores the potential of open source platforms as democratizing tools in entrepreneurial finance, offering a pathway for diverse entrepreneurs to signal value and attract investor interest in a competitive funding environment.

Wesley II et al. **(2022)** investigated how resource providers—specifically experienced founders and investors—evaluate startups and respond to entrepreneurs' informational signals when deciding whether to offer financial or social support. The study, titled *"Will the Startup Succeed in Your Eyes? Venture Evaluation of Resource Providers During Entrepreneurs' Informational Signaling"*, introduces a model of venture evaluation influenced by the provider's investment propensity and venturing experience. Using real-time decision data, the research reveals that individuals with founding experience—whether as non-investor founders or investors—tend to rely more on their investment propensity when assessing new ventures compared to those without such experience. Additionally, post-hoc analyses indicate that founding experience is strongly linked to the willingness to provide social resources. The study emphasizes the critical role of the evaluator's background in shaping resource allocation decisions and highlights the nuanced ways in which investment history and experience influence the early-stage support that startups receive.

Lin and Maruping (2022) explored how open source collaboration (OSC) influences value creation in digital startups. Their study, *"Open Source Collaboration in Digital Entrepreneurship,"* develops a theoretical framework examining how digital ventures benefit from OSC depending on their stage of maturity—conception, commercialization, or growth—and their mode of engagement, whether inbound (utilizing external contributions) or outbound (contributing to external projects). Analyzing a dataset of 17,552 matched digital startups with

monthly panel observations from 2008 to 2017, the researchers found that startups in the conception and commercialization stages gained more from inbound OSC, while those in the growth stage saw greater benefits from outbound OSC. The study emphasizes the dynamic role of OSC in ideation, experimentation, and scaling activities, and highlights the strategic importance of managing knowledge flows in digital entrepreneurship. These findings offer valuable insights into when and how open innovation through OSC can enhance startup value across different phases of development.

Fiet (2022) investigated the contrasting risk avoidance strategies used by business angels and venture capital firm investors in the venture capital market. His study, *"Risk Avoidance Strategies in Venture Capital Markets,"* highlights that these two investor groups differ significantly in how they perceive and mitigate market and agency risks. Business angels tend to depend more heavily on entrepreneurs to shield them from market losses, leading them to prioritize concerns about agency risk. In contrast, venture capitalists focus more on market risk, having developed contractual mechanisms—such as standardized legal terms—to manage agency-related threats. This divergence in risk assessment and mitigation results in a segmented venture capital market. The research provides critical insights for entrepreneurs seeking funding and opens avenues for further exploration into how differing investor strategies influence startup financing dynamics.

Lutfiani et al. **(2022)** present an AI-driven solution in their work titled *"Artificial Intelligence Based on Recommendation System for Startup Matchmaking Platform."* The study addresses the inefficiencies of the traditional partner-matching process for startups, which typically involves manually seeking out suitable collaborators—a time-consuming and often ineffective method. By developing an intelligent matchmaking platform integrated with communication tools, the authors leverage artificial intelligence to streamline and automate the partner discovery process. Using a Systematic Literature Review (SLR), the platform's development is aligned with the rapid growth needs of modern startups. The core innovation of this research lies in its recommendation system, which facilitates more effective matchmaking between startups and potential partners, enhancing operational efficiency and strategic alignment in entrepreneurial ecosystems.

Berger and Hottenrott (2021), in their study titled *"Start-up Subsidies and the Sources of Venture Capital,"* examine how public subsidies influence follow-on financing for startups,

particularly from different types of venture capital (VC) sources. The research provides empirical evidence that while initial correlations suggest subsidies attract various forms of VC—such as Government VC, Independent VC, Corporate VC, and Business Angels—this relationship weakens when firm-specific characteristics are controlled for using econometric matching techniques. Ultimately, the study finds that subsidies are more significantly associated with Government VC and Business Angel financing. This distinction highlights how different investor types perceive and value public certification and early liquidity differently, offering key insights for policymakers and entrepreneurs aiming to optimize funding strategies and understand venture dynamics within the startup ecosystem.

Alperovych, Groh, and Quas (2020) investigate the design and effectiveness of governmental venture capital (GVC) fund programs aimed at supporting young innovative companies (YICs) and bridging the equity gap. Their study examines the heterogeneity of GVC programs across Europe, identifying key design features—such as location, colocation, syndication, and industry focus—that significantly influence the likelihood of GVC-backed companies attracting additional private venture capital funding. Furthermore, these design choices affect the growth and innovation outcomes of the supported firms. The research offers valuable policy insights to enhance the effectiveness of government interventions in early-stage entrepreneurial finance.

Conti and Graham (2020) explore the implications of "two-sided" learning in a newsvendor model where both firms and consumers learn about a product's value simultaneously through shared information, such as sales data and consumer feedback observable via social media. Unlike traditional models assuming one-sided learning, their analysis reveals that the value of information can be negative for firms when consumers also learn, leading to an optimal stocking quantity that is often lower than under one-sided learning. In some cases, this quantity falls below the critical fractile policy commonly recommended in existing literature. Their findings highlight the significant risks firms face if they fail to account for this two-sided learning dynamic in inventory decisions.

de Mol, Cardon, de Jong, Khapova, and Elfring (2020) investigate the role of entrepreneurial passion within new venture teams and its impact on both short- and long-term venture performance. Using multi-source, multi-wave data from 107 new venture teams in an accelerator program, they examine how the average level of passion and the diversity of passion

within teams influence outcomes. Their findings reveal that while average team passion does not significantly affect performance, greater passion diversity—especially intensity separation—is negatively associated with venture success. This research offers valuable insights into the dynamics of passion and group affective diversity in entrepreneurial teams.

Bronzini, Caramellino, and Magri (2020) employ a difference-in-differences methodology to isolate the treatment effect of venture capital (VC) financing by comparing Italian startups funded by private VC between 2004 and 2014 with firms rejected at late stages of the VC screening process. Their analysis shows that VC-backed startups achieve larger size and greater innovation, although sales growth is comparable and profitability initially worse compared to non-VC-backed firms. The reduced profitability is linked to higher labor costs and longer commercialization times for innovative projects but tends to normalize after four years. Additionally, VC-backed firms demonstrate a significant increase in equity and changes in financial structure. Notably, no difference in survivorship rates between VC-backed and rejected firms is found. This study provides nuanced insights into the impacts of VC funding on firm growth, innovation, and financial dynamics.

Bapna (2019) conducts a randomized field experiment to investigate how different signals—product certification, prominent customers, and social proof—interact in influencing early-stage equity investment decisions in technology ventures. The study reveals that product certification signals complement both prominent customer and social proof signals. Specifically, investors exposed to combined signals of product certification with prominent customers are 72% more likely to express interest in equity investment, while those exposed to product certification and social proof signals show a 65% increased likelihood of investment interest. These findings highlight that in technology ventures, product-related signals are crucial for enhancing the value of market and investment-related signals in attracting investors.

Coussement, Fourné, Kim, and Kotha (2019) explore how evaluators decide to commit resources to early-stage academic inventions despite significant uncertainty and potential red flags in standard evaluation criteria. Through text analysis of nearly 700 evaluation reports from a university technology transfer office, the study finds that resource commitments are influenced by assessments of feasibility—such as overcoming doubts and evaluating technological maturity—and desirability, including familiarity with the background and scientific complexity. The research highlights that evaluators often take a "leap of faith" to

support promising inventions early, helping to avoid prematurely discarding high-potential innovations. These insights are broadly applicable to management contexts involving high uncertainty and early-stage opportunity assessment.

Zhao and Wang (2023) investigated the impact of crowdfunding platforms on bridging early-stage financing gaps for socially-oriented startups. Their study, "Crowdfunding as a Catalyst for Social Venture Growth," analyzes data from over 1,200 campaigns on major crowdfunding platforms from 2015 to 2021. The researchers found that successful social ventures not only raised capital but also built strong communities that contributed to long-term engagement and legitimacy. The study emphasizes the dual role of crowdfunding as both a financing mechanism and a social validation tool, highlighting how campaign transparency and narrative framing significantly influence investor trust and funding outcomes in the social entrepreneurship domain.

Martinez et al. (2024) examined how the structure and governance of startup accelerators influence the post-acceleration performance of ventures. Their paper, titled "Accelerator Governance and Startup Trajectories: Insights from Global Cohorts," uses a mixed-method approach combining survey data from 300 startups and interviews with accelerator managers across 15 countries. The findings suggest that accelerators with hybrid governance models—balancing public and private sector involvement—enhance startups' access to diverse resources and improve scalability. The study highlights the importance of governance transparency and alignment of stakeholder incentives in driving sustainable startup growth and ecosystem development.

Nguyen and Park (2023) explored the role of cultural factors in shaping investor decision-making processes in emerging markets. Their research, "Cultural Dimensions and Venture Capital Investment Choices," draws on Hofstede's cultural framework and analyzes 1,100 venture deals across Southeast Asia. The study reveals that investors from high uncertainty avoidance cultures emphasize risk mitigation strategies, including staged financing and intensive due diligence, while those from collectivist cultures prioritize relational trust and founder networks. These cultural nuances are found to significantly affect deal structures and post-investment support, underscoring the need for culturally informed approaches to entrepreneurial finance in global contexts.

Singh and Patel (2024) focused on the role of blockchain technology in transforming venture capital operations through increased transparency and efficiency. Their study, "Blockchain-Enabled Venture Capital: Redefining Fundraising and Governance," presents a theoretical model supported by interviews with 25 venture capital firms experimenting with blockchain-based smart contracts and tokenized equity offerings. The authors argue that blockchain reduces information asymmetry and administrative overhead, fostering more inclusive and decentralized investment models. The research highlights early adopters' experiences, challenges related to regulatory uncertainty, and the potential for blockchain to democratize access to startup funding.

Kim and Lee (2022) analyzed the impact of gender-diverse founding teams on startup innovation output and financing outcomes. Their paper, "Gender Diversity in Founding Teams: Effects on Venture Capital and Innovation Performance," utilizes a longitudinal dataset of 1,500 tech startups from 2010 to 2020. The findings indicate that gender-diverse teams are more likely to attract venture capital, particularly from investors with a track record of supporting diversity initiatives. Additionally, such teams tend to produce more patent applications and demonstrate higher rates of product innovation. The study contributes to the growing evidence supporting diversity as a strategic asset in entrepreneurial ecosystems and financing landscapes.

2.3 CONCLUSION

The reviewed literature underscores the complexity and uncertainty inherent in evaluating early-stage scientific inventions. Despite significant challenges in assessing commercial feasibility and potential, evaluators often take calculated risks by committing resources to promising projects early on. Key factors influencing these decisions include feasibility assessments—such as overcoming doubts and judging technological maturity—and desirability considerations like familiarity with the invention's background and scientific complexity. This "leap of faith" approach is crucial to preventing the premature rejection of high-potential innovations.

Future research and practice should focus on:

- Developing more nuanced evaluation frameworks that better balance risk and opportunity under uncertainty.
- Enhancing early-stage assessment tools to integrate qualitative insights with quantitative metrics.
- Improving communication and knowledge sharing among evaluators to reduce bias and increase consistency.
- Investigating the role of organizational context and evaluator experience in shaping resource commitments.
- Exploring decision-support systems that assist evaluators in making informed, timely resource allocations.

Such advances can foster more effective support for breakthrough inventions, ultimately driving technological progress and societal benefit.

CHAPTER 3

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The current inventor-investor matching landscape relies on basic filters like industry sector, funding stage, and geographic location. These platforms primarily match users based on static profiles and manual searches, lacking intelligent analysis of project viability or investor compatibility. The system evaluates opportunities through superficial metrics such as founder credentials and pitch decks, often overlooking critical performance indicators like prototype readiness, market traction, or investor track records. While these methods provide basic connectivity, they fail to optimize matches for mutual success, resulting in low conversion rates and inefficient capital allocation.

3.1.1 PROBLEM DEFINITION

- Inefficient Matching: Reliance on manual filters leads to mismatched connections.
- Limited Data Utilization: Ignores dynamic factors like project milestones or investor risk appetite.
- Low Engagement: Passive discovery tools hinder active collaboration.

3.2 PROPOSED SYSTEM

Our platform introduces a data-driven approach to connect inventors and investors, emphasizing actionable metrics over static profiles. Key innovations include:

- Performance-Centric Matching: Analyzes inventor metrics (prototype progress, IP status) and investor behavior (funding history, sector preferences).
- Multi-Algorithm Integration: Combines collaborative filtering, regression models, and clustering to predict optimal partnerships.

- Dynamic Scoring: Evaluates projects based on traction (user testing, patents) and investors on responsiveness and deal closure rates.
- Joint Venture Tools: Facilitates syndicated investments with milestone-based funding controls.

3.2.1 ADVANTAGES

- Higher Accuracy: Machine learning models (random forest, gradient boosting) prioritize compatibility over generic filters.
- Transparency: Real-time ratings and success metrics build trust.
- Scalability: Automated workflows handle growing user bases without compromising match quality.
- Democratization: Levels the playing field for underrepresented innovators through objective scoring.

By focusing on measurable outcomes and adaptive algorithms, our system transforms inventor-investor collaboration into a precise, results-oriented process.

CHAPTER 4

CHAPTER 4
REQUIREMENT SPECIFICATIONS

4.1 INTRODUCTION

This project introduces an innovative digital platform designed to revolutionize how inventors and investors connect by merging crowdfunding with joint venture capitalism. The system employs advanced feature selection algorithms to identify optimal matching criteria, enhancing connection accuracy while optimizing performance. By analyzing key parameters such as project viability, funding requirements, and investor preferences, the platform ensures efficient capital allocation and productive partnerships. This document serves as a comprehensive guide for the system's development lifecycle (SDLC), outlining functional requirements, technical specifications, and operational workflows. It provides developers with a structured framework for implementation and testing, while establishing protocols for future requirement modifications through formal change approval processes.

Functional Requirements:

Input: The system processes user profiles (inventors/investors), project proposals, investment terms, and communication logs. Access is role-based, with JWT authentication ensuring secure identity verification. Inventors input project details (stage, sector, funding needs), while investors define preferences (risk appetite,industryfocus).

Output: The platform generates dynamic matchmaking results, investment dashboards, and automated reports. Real-time analytics provide insights into successful connections, funding trends, and user engagement.

4.2 HARDWARE AND SOFTWARE SPECIFICATION

4.2.1 HARDWARE REQUIREMENTS

- Processor – i3, i5, i7
- Speed – 2.4 GHz
- RAM – 8 GB or Higher
- Hard Disk – 256 GB

4.2.2 SOFTWARE REQUIREMENTS

- Frontend - React.js (v18+), Tailwind CSS (v3+)
- Backend - Node.js (v16+), Express.js (v4+)
- Database - MySQL (v8.0+)
- Development Tools - VS Code, MySQL Workbench, Postman
- Additional Services - NodeMailer (SMTP), JWT Authentication

4.2.2.1 JAVASCRIPT

JavaScript serves as the fundamental building block of our full-stack development environment, powering every layer of our MERN (MySQL, Express, React, Node) application architecture. As a versatile, high-level programming language that conforms to the ECMAScript specification, JavaScript enables us to create a dynamic, responsive platform that seamlessly connects inventors with potential investors. The language's unique position as the only programming language natively understood by web browsers allows us to deliver a rich, interactive user experience through our React-based frontend, while its server-side implementation via Node.js provides the robust backend infrastructure needed to handle complex matching algorithms and financial transactions.

JavaScript's event-driven, non-blocking I/O model is particularly well-suited to our platform's requirements, efficiently managing multiple concurrent operations such as real-time messaging between users, simultaneous investment proposals, and dynamic content updates without performance degradation. The language's

recent advancements through ECMAScript 2022 standards have introduced powerful features like top-level await for cleaner asynchronous operations, private class fields for better encapsulation, and error cause chaining for more debuggable exception handling - all of which we leverage to build more reliable and maintainable code.

JavaScript's prototype-based object system, combined with its functional programming capabilities, allows us to implement sophisticated matching algorithms while keeping our codebase flexible and adaptable to changing business requirements. The language's dynamic typing system facilitates rapid prototyping and iteration, crucial for our startup environment, while disciplined coding practices and tooling (such as PropTypes and ESLint) ensure we maintain type safety where it matters most.

Perhaps most importantly, JavaScript's vast ecosystem, with over 2 million packages available through npm, provides us with battle-tested solutions for everything from authentication (JWT) to email communications (NodeMailer) to database connectivity (Sequelize), significantly accelerating our development timeline. The language's universal browser support ensures our platform remains accessible to all users regardless of their device or operating system, while its server-side capabilities through Node.js create a unified development experience across our entire stack. This combination of versatility, performance, and ecosystem support makes JavaScript the ideal foundation for building our innovative crowdfunding platform that demands seamless interaction.

4.2.2.2 INTRODUCTION TO JAVASCRIPT

JavaScript serves as the cornerstone of our full-stack development, powering both the frontend and backend components of our crowdfunding platform. As a versatile, high-level programming language, JavaScript enables the creation of

dynamic, interactive web applications through its event-driven architecture and non-blocking I/O model. The language's universal browser support and server-side runtime environment (Node.js) provide a unified development experience across our entire technology stack. Modern ECMAScript features like arrow functions, async/await, and destructuring assignments enhance code readability and maintainability while improving developer productivity.

JavaScript's prototype-based object orientation and first-class functions allow for flexible implementation of our matching algorithms, while its dynamic typing system facilitates rapid prototyping of new features. The language's extensive ecosystem, including frameworks like React and Express, along with thousands of npm packages, provides comprehensive solutions for building complex functionalities such as real-time messaging, JWT authentication, and data visualization. JavaScript's JSON-native syntax simplifies data exchange between our React frontend and Node.js backend, ensuring seamless communication between inventors and investors.

The language's asynchronous capabilities are particularly valuable for handling concurrent operations like processing multiple investment requests or sending batch notifications. JavaScript's cross-platform compatibility ensures our platform delivers a consistent user experience across devices and operating systems. With robust tooling support (ESLint, Babel, Webpack) and a vibrant developer community, JavaScript provides the ideal foundation for building scalable, maintainable web applications that meet the evolving needs of our crowdfunding platform.

DESIGN PHILOSOPHY AND FEATURES

JavaScript embodies a unique design philosophy that combines flexibility with

power, making it ideally suited for our crowdfunding platform's diverse requirements. At its core, JavaScript follows a multi-paradigm approach that seamlessly blends prototype-based object orientation with functional programming concepts. This dual nature allows our development team to implement complex matching algorithms using higher-order functions while maintaining clean state management through object composition.

The language's "everything is an object" mentality, combined with first-class functions, enables powerful patterns that we leverage throughout our application. In the frontend React components, we utilize JavaScript's closure capabilities to create custom hooks for state management. The backend Express middleware chain takes advantage of JavaScript's functional aspects to create composable authentication and validation layers. This design philosophy promotes code reuse and modularity - key requirements for our rapidly evolving platform.

SYNTAX AND SEMANTICS, INDENTATION

JavaScript employs C-style syntax with significant flexibility in code formatting, though we enforce consistent 2-space indentation through ESLint for improved readability. The language uses semicolon auto-insertion but we explicitly include them for clarity. Curly braces {} define code blocks, while parentheses () control expression evaluation. Modern ES6+ features like arrow functions (() => {}) provide concise syntax for callbacks and methods. JavaScript's lexical scoping determines variable accessibility, with let/const replacing var for block-scoped declarations. Template literals (text ${variable}) enable clean string interpolation, while destructuring assignments simplify data extraction from objects and arrays.

NODE.JS LIBRARIES AND ECOSYSTEM

Node.js boasts a rich ecosystem of built-in modules and third-party packages that provide robust solutions for diverse development needs. The core Node.js library includes essential modules for file system operations (fs), HTTP servers (http), path manipulation (path), and cryptographic functions (crypto), forming a solid foundation for backend development. For web applications, Express.js extends Node's capabilities with middleware support, routing, and templating, while frameworks like Nest.js offer enterprise-grade architecture.

The npm (Node Package Manager) registry, which currently hosts over 2.1 million packages, offers an extensive and ever-growing collection of specialized tools and libraries designed to cater to virtually every possible development use case. For instance, database interactions and management are significantly simplified and made more efficient through the use of powerful Object-Relational Mapping (ORM) tools such as Sequelize, which is commonly used for SQL databases, and Mongoose, which is tailored for working with MongoDB databases. When it comes to implementing secure and reliable user authentication systems, developers often rely on well-established libraries like Passport.js, which supports various authentication strategies, and JSON Web Tokens (JWT), which is widely used for token-based authentication and authorization processes. To facilitate real-time, bidirectional communication between clients and servers, the Socket.io library is employed, allowing for seamless event-driven interactions. Additionally, for handling multipart/form-data, especially for file uploads, the Multer middleware provides robust and customizable file processing capabilities. Together, these tools form an essential part of modern web application development in the Node.js ecosystem.

- **Express.js** – The minimalist web framework that forms our application's backbone, handling routing and middleware management.
- **Sequelize** – A promise-based ORM for MySQL that simplifies database interactions and model relationships.

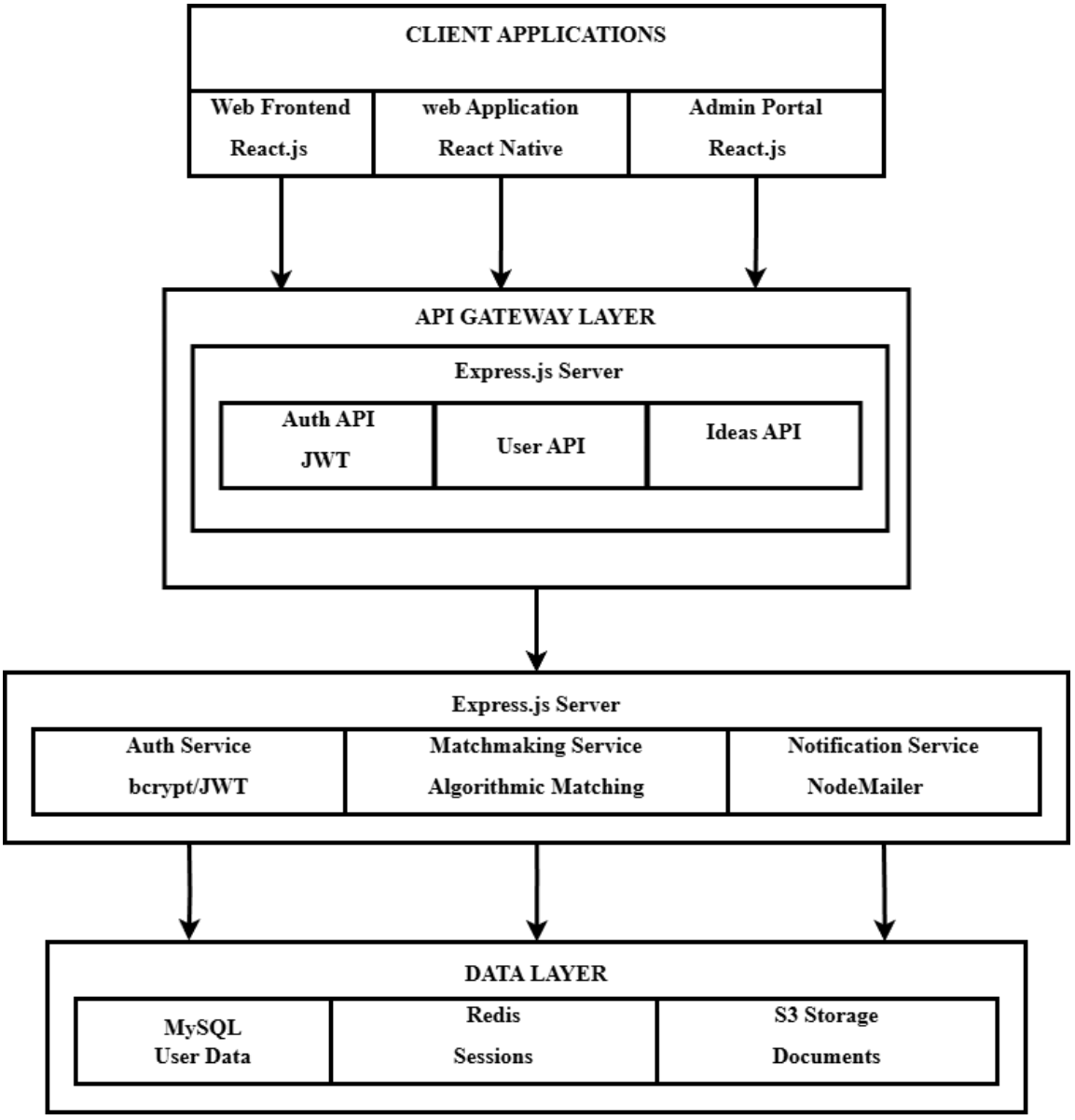
- **jsonwebtoken (JWT)** – For implementing secure authentication through token-based sessions.
- **bcryptjs** – Password hashing library that securely stores user credentials.
- **nodemailer** – Enables email functionality for notifications and verification.
- **dotenv** – Manages environment variables and application configuration.
- **cors** – Handles Cross-Origin Resource Sharing for API security.

CHAPTER 5

CHAPTER 5

SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

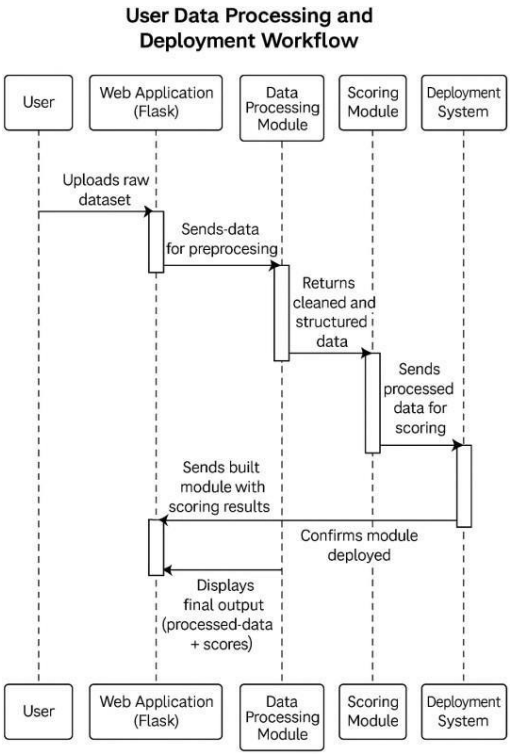


5.2 UML DIAGRAMS

5.2.1 USE CASE DIAGRAM

A Use Case Diagram for the Inventor-Investor Matchmaking Platform presents a graphical overview of the system's key functionalities, illustrating interactions between actors—such as Inventors, Investors, and Administrators—and the system. Each use case represents a goal-driven sequence of actions, like registering, creating profiles, matching, messaging, or monitoring performance. These use cases are shown as ellipses, capturing how actors derive value through system interactions like:

- Inventors showcasing innovations
- Investors discovering and evaluating opportunities
- Administrators overseeing platform integrity and performance

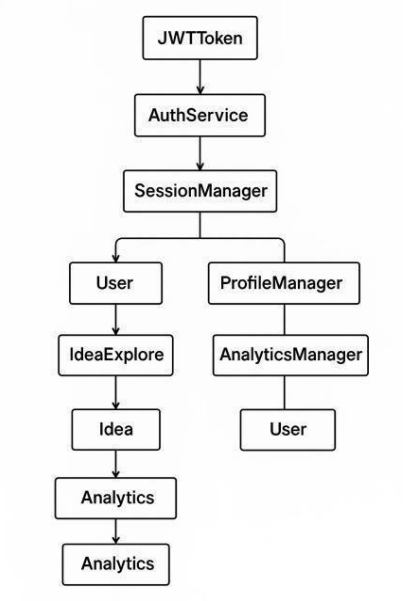


5.2.2 SEQUENCE DIAGRAM

A Sequence Diagram for the Inventor-Investor Matchmaking Platform illustrates how different components and users interact over time to complete processes such as onboarding, profile matching, or secure communication. It captures the order of message exchanges between entities like Inventor, Investor, Authentication System, Matching Engine, and Messaging Module. These diagrams—also known as event diagrams or timing diagrams—visually represent how actions such as user registration, credential verification, profile creation, and real-time messaging are sequenced. For instance, a typical sequence might include:

- Inventor submits registration
- System sends secure login credentials via email
- User logs in and creates a profile
- Matching engine identifies potential investors
- Investor receives suggestion and initiates messaging

This helps in understanding system behavior and ensures smooth integration of all components within the intelligent matchmaking ecosystem.

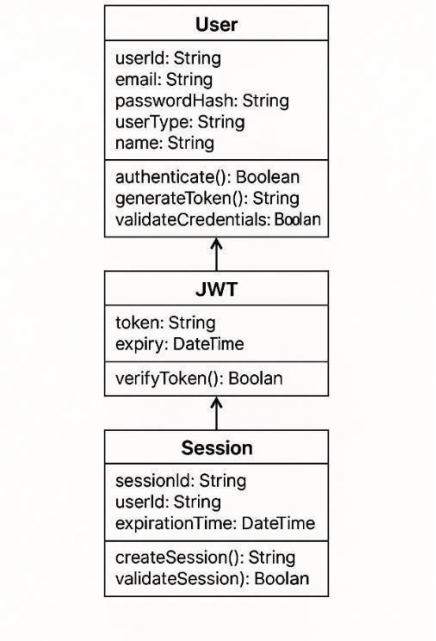


5.2.3 CLASS DIAGRAM

A Class Diagram for the Inventor-Investor Matchmaking Platform is a static structure diagram that outlines the system's architecture by detailing its core classes, their attributes, methods, and interrelationships. It models the backbone of the platform, helping visualize how data and behavior are organized.

Key classes may include:

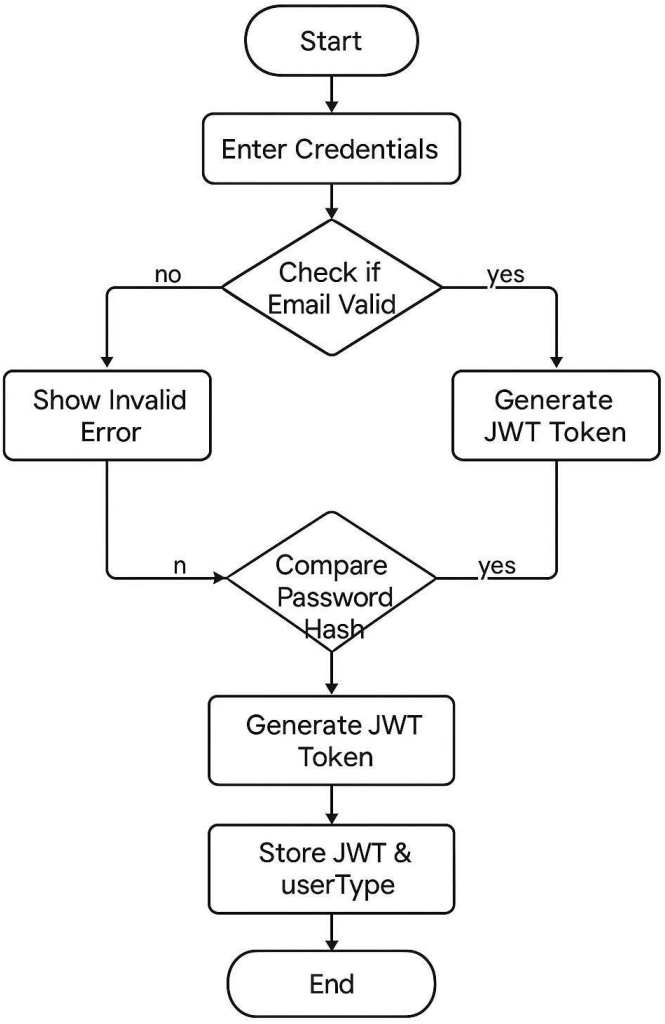
- **User** (attributes: userID, name, email, role; methods: register(), login(), updateProfile())
- **InventorProfile** (attributes: expertise, projectStage, innovationDetails)
- **InvestorProfile** (attributes: investmentCriteria, fundingHistory, preferredSectors)
- **MatchingEngine** (methods: matchUsers(), evaluateCompatibility())
- **Message** (attributes: senderID, receiverID, timestamp, content; methods: send(), receive())
- **AdminDashboard** (methods: monitorUsers(), flagRisk(), generateAnalytics())



5.2.4 ACTIVITY DIAGRAM

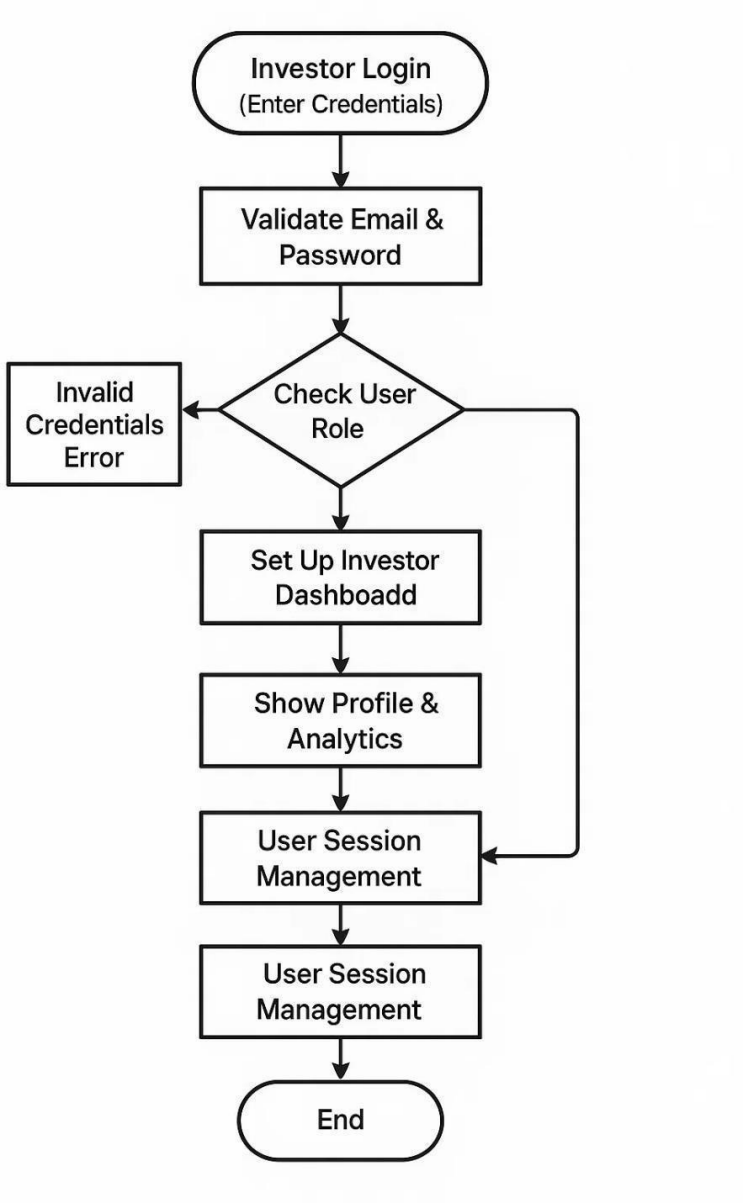
Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- An encircled circle represents the end of the workflow.
- A black circle represents the start of the workflow



5.2.5 DATA FLOW DIAGRAM

A workflow diagram (also known as a workflow) provides a graphic overview of the business process. Using standardiz`ed symbols and shapes, the workflow shows step by step how your work is completed from start to finish.



CHAPTER 6

CHAPTER 6
WORKFLOW

6.1 Requirement Analysis

The foundation of any successful software project lies in understanding the core purpose of the system, its users, and the functional expectations. The Idea Exchange Dashboard was no exception. The project began with a thorough and deliberate requirement analysis phase that aimed to define a clear roadmap for the development cycle. The team recognized early on that without proper insight into what users wanted or needed, the product might fall short of expectations or include unnecessary features. Thus, effort was devoted to engaging potential users through informal yet insightful interviews. These included conversations with fellow students, mentors, and stakeholders who could act as potential Innovators or Investors.

To bring structure to this qualitative data, the team created detailed user personas representing the two primary user types: Innovators and Investors. User journey maps were drawn to simulate how these personas would interact with the system, revealing essential interaction points and user goals. Through this approach, several pain points in existing solutions were uncovered. Innovators often lacked a proper channel to showcase ideas and receive feedback or funding, while Investors struggled to find credible, structured innovation pipelines that offered meaningful filtering or engagement metrics.

With this understanding, roles and responsibilities within the application were clearly defined. Innovators were positioned as content creators — users who could submit ideas, upload relevant documentation, and connect with potential backers. Investors, in contrast, needed a more analytical view, allowing them to explore submitted ideas, assess innovation potential, and initiate communication

with promising Innovators. Essential system features were scoped out, including secure user login with authentication, role-based dashboard experiences, document uploads, user profile management, networking capabilities (including search and connection requests), and data visualization through charts and metrics. This phase concluded with a requirements specification document that served as the blueprint for the entire development process, reducing ambiguity and ensuring alignment between design, development, and user expectations.

6.2 Project Setup and Design

Once the requirements were solidified, the team shifted focus to setting up the technical architecture and designing the user interface. This phase involved selecting appropriate tools, libraries, and frameworks to build a scalable, maintainable system while simultaneously crafting a user interface that was intuitive and efficient. The frontend was initialized using the create-react-app toolkit, which provided a modern boilerplate for React development. React was chosen for its component-based architecture, high performance, and strong developer ecosystem. To manage client-side routing between various pages and dashboards, the React Router library was integrated, while Chart.js was selected for rendering interactive and responsive data visualizations on investor dashboards.

To enforce coding best practices and catch potential issues early, ESLint was added as a linting tool. On the backend, a new Node.js project was initialized, leveraging Express.js to set up RESTful APIs. The backend followed a modular architecture, separating responsibilities into controllers (handling business logic), routes (defining endpoints), and middleware (managing request lifecycles and security).

Parallel to this, the team invested time in designing the user experience and interface layout. Wireframes were sketched for all critical screens, including login and registration pages, dashboards for both roles, profile editing popups, idea submission forms, and preview sections. Each component was mapped out to determine its layout, behavior, and interactions with other components. These wireframes were translated into a component hierarchy plan to guide developers during implementation and reduce redundant code. Special attention was given to the logical flow of data between components, ensuring that state management and API calls would be seamless and scalable. API routes were defined in advance to handle tasks such as authentication (/api/login, /api/register), idea management (/api/ideas), connection operations (/api/connections), and user profile handling (/api/user). This structured design and setup phase laid a strong technical and visual foundation for smooth development.

6.3 Development Phase

With the project environment ready and design plans in place, the development phase began. This was the most time-intensive part of the project, involving building the frontend and backend concurrently while ensuring consistency in data flow, component logic, and visual design. The team divided responsibilities to work in parallel — backend developers focused on server logic and API development, while frontend developers concentrated on UI implementation, state management, and user interactions.

On the backend, secure authentication was prioritized. JWT (JSON Web Token) was implemented to generate and verify tokens upon user login, ensuring session integrity. On the frontend, these tokens were stored in localStorage to persist user sessions across page refreshes. The registration system captured all necessary

user data, including role differentiation, which allowed the application to conditionally render dashboards and restrict access to role-specific content.

The idea submission module allowed Innovators to create and submit ideas by filling out fields like title, domain, and description. Document upload functionality was added using multer, enabling users to attach PDFs, images, or text files. These documents were stored securely and referenced within the idea cards. A dedicated endpoint managed all connection requests, allowing users to send, accept, or reject connection requests, with request statuses stored and updated to reflect pending, accepted, or rejected states.

On the frontend, React components were developed with a focus on modularity and reusability. Conditional rendering allowed different dashboards to appear based on user roles, enhancing user experience. Innovators saw submission tools and their idea listings, while Investors saw search interfaces, connection options, and analytics charts. Profile editing was made accessible through a modal popup, where users could update personal details and upload a GPay QR code image for donations or investments. Idea cards were designed to be dynamic, showing a preview of the uploaded document, including file-type detection to show the correct icon or embed the file preview. React Router ensured seamless navigation between login, registration, dashboards, and additional features. Styling was implemented using custom CSS, taking advantage of Flexbox and Grid layouts for responsive design. Animations were introduced for modals, transitions, and data updates, giving the interface a modern, interactive feel. By the end of this phase, a robust MVP had been developed, integrating core functionalities across user roles.

6.4 Testing and Debugging

Post-development, the application underwent a structured and rigorous testing

phase. The primary goal was to verify that all components and features worked as expected, both in isolation and in combination, and to identify any bugs or UI inconsistencies. The testing process included multiple layers: unit testing, integration testing, manual UI testing, and real-world scenario simulation (cross-functional testing).

Unit testing focused on small, isolated components and functions, such as login handlers, API data fetchers, and utility methods. For example, the logout functionality, which initially broke due to an undefined function (handleLogout), was corrected by implementing a new handler that cleared the user session and redirected the user to the login page. Integration testing ensured that different system modules communicated properly. A key example was the idea submission module, where newly submitted ideas were tested to ensure they appeared correctly in the Investor's dashboard charts and lists.

Cross-functional testing involved simulating full user workflows. Testers acted as both Innovators and Investors to walk through the process of account creation, profile updates, idea submission, search, and connections. This helped uncover bugs that were not apparent during isolated tests. Manual UI testing was done to verify that all interactive elements (modals, buttons, file uploads) worked smoothly and provided clear user feedback. One notable issue arose with document previewing, where certain files failed to load or display correctly. This was fixed by implementing a robust file-checking component that rendered previews based on MIME types (e.g., PDFs were embedded using <iframe>, images with <img>, and text files with a scrollable text container).

Additionally, form responsiveness was tested across various screen sizes. Issues like broken layouts and input overflow on mobile devices were addressed by introducing responsive styling, media queries, and width constraints. By the end of this phase, the application was not only feature-complete but also stable, user-friendly, and free from major functional defects.

6.5 Deployment and Final Review

In the final phase, the team deployed the application in a local environment to demonstrate its capabilities and perform final verification. The backend server was hosted at localhost:5000 and the React frontend at localhost:3000. CORS policies and relative path routing were configured to ensure seamless integration between the frontend and backend without errors or blocked requests.

The team conducted a final checklist walkthrough to validate the presence and functionality of all critical features. Authentication mechanisms were tested to confirm redirection, session handling, and role-based access. Dashboards were evaluated for accuracy, ensuring that Innovators could access idea submission tools and Investors could see search and analytics views. The profile editing popup was verified for proper data handling and image upload capability. The document preview functionality was tested with various file types to ensure visual fidelity. Networking features, such as sending and managing connection requests, were confirmed to work with appropriate UI updates.

User Acceptance Testing (UAT) followed, with testers emulating real usage scenarios. This included observing visual feedback mechanisms such as alerts, error messages, confirmation modals, and data transitions. Attention was given to aesthetic consistency, including uniform font usage, color schemes, and element spacing. The application responded well to different device viewports, including mobile and tablets, confirming the success of responsive design strategies.

This final stage concluded with a polished, fully functional application that was stable, tested, and ready for hosting on a production server or cloud platform. It was a demonstration-ready product that encapsulated a complete innovation matchmaking system and provided an excellent foundation for future feature enhancements or scaling.

CHAPTER 7

CHAPTER 7
USER & INVESTOR LOGIN

7.1 User Login System – Full Workflow and Breakdown Overview

The User Login System is the foundational access layer of the Idea Exchange Dashboard, responsible for securely managing user authentication and role-based authorization. It is designed to do more than just verify user credentials—it dynamically adjusts the user interface and functionality depending on whether the logged-in user is an Innovator or an Investor. Innovators are given tools to submit and manage their ideas, while Investors are offered interfaces to explore, evaluate, and connect with relevant projects. This role-based customization improves usability, streamlines workflows, and ensures that users only access content relevant to their objectives. Built with modern web technologies, the system uses JSON Web Tokens (JWT) for stateless and secure authentication, which allows the server to verify users without maintaining session data. Tokens are securely stored in the browser's localStorage, enabling persistent sessions that survive page refreshes and browser restarts. The backend, powered by Node.js and Express, handles login requests, validates user credentials, and issues signed tokens. A major strength of the system lies in its seamless user experience—users do not need to log in repeatedly as long as their token remains valid. This convenience is supported by a robust session validation mechanism that protects against unauthorized access while maintaining a fluid user journey. Overall, the login system ensures a secure, personalized, and frictionless entry into the platform, enabling role-specific interactions and safeguarding access to the dashboard's critical features.

Objectives

The primary objectives of the User Login System are centered around security,

personalization, and seamless user interaction. First and foremost, the system is designed to authenticate users securely by validating email and password credentials against stored, encrypted data, ensuring that only authorized individuals can gain access to the platform. Beyond authentication, the system identifies each user's role—whether they are an Innovator or an Investor—allowing the application to dynamically adjust its interface and available features to suit that specific role. This ensures a targeted and efficient experience for all users. Another crucial goal is session persistence; the system maintains active user sessions even as they navigate across different pages or refresh the browser, reducing friction and enhancing usability. Lastly, the login system enforces strict access control by restricting sensitive parts of the application—such as dashboards, idea submissions, and connection features—to logged-in users only. This layered approach to user management forms the backbone of a secure, intuitive, and role-aware digital environment.

7.2 Login Process: Step-by-Step Breakdown

7.2.1 Login Interface Presentation

The login process begins with the presentation of a clean and user-friendly interface designed to offer clarity and ease of use. Upon reaching the login page, users are greeted with a simple form that includes essential input fields and action buttons. Specifically, the form comprises an email input field where users enter their registered email address, a password input field to input their secret credential, and a login button that initiates the authentication process. This straightforward layout minimizes confusion and streamlines user interaction. In addition to the basic form, optional user experience (UX) enhancements are planned to further support usability. These include a "Forgot Password" link, which redirects users to a password recovery workflow in case they have

forgotten their credentials, and a "Register Now" option that guides new users to create an account. Together, these elements create a smooth entry point into the platform, making the authentication experience both accessible and functional for all users.

7.2.2 Form Validation and Submission

Once the user fills in the login form and clicks the Login button, the system initiates a crucial validation step to ensure data integrity and prevent unnecessary server load. This begins with client-side validation, where the email input is checked against a standard format to confirm it follows the conventional structure (e.g., user@example.com). Simultaneously, the password field is examined to ensure it is not left empty, as a missing password would render the login attempt invalid. These preliminary checks enhance user experience by catching common input errors immediately, without needing to communicate with the server. If both fields pass validation, the form is then programmatically submitted as a POST request to the backend API endpoint (POST /api/login). This request contains a payload in JSON format that includes the user-entered email and password, securely initiating the server-side authentication process. This approach strikes a balance between frontend efficiency and backend security, ensuring only properly formatted and complete login attempts reach the server.

7.2.3 Server-Side Authentication

On the server side, built using Node.js and Express, the authentication process begins as soon as the login request is received. The server first queries the database to locate the user associated with the email provided in the login form. If the email exists, the next step is to securely verify the password. The entered

password is hashed using a cryptographic hashing algorithm like bcrypt, ensuring that the password is never stored in plaintext. This hashed value is then compared with the hashed version stored in the database. If the passwords match, the system confirms the user's identity and proceeds with the authentication. Upon successful verification, the server generates a JWT (JSON Web Token), a secure, encrypted token used for user authentication in future requests. This token is signed with a secret key and may include an expiration time (usually set to 24 hours) to enhance security. Along with the token, the server responds with key user metadata, including a unique userId for identification, the userType (either "innovator" or "investor" to tailor the user's experience), the name of the user, and the JWT token itself, which is essential for maintaining secure access to protected routes. This step ensures both security and a personalized experience for users as they interact with the system.

7.2.4 Token & Role Storage on Frontend

Once the frontend receives the server's response, which includes the JWT token and userType, the next step is to securely store this crucial information in the browser's localStorage. This allows the application to persist the user's session, ensuring they do not need to log in again during their current browsing session. The JWT token, which is required for authenticating future requests, is stored using the localStorage.setItem("token", token) method, while the userType (which identifies whether the user is an Innovator or Investor) is also stored via localStorage.setItem("userType", userType). The userType is particularly important because it determines the specific interface and functionality the user should have access to. Based on the userType, the system automatically redirects the user to their designated dashboard. If the user is an Innovator, the application will navigate them to the Innovator Dashboard using

navigate("/dashboard/innovator"), providing them with the relevant tools to submit, showcase, and manage their ideas. Conversely, if the user is an Investor, the system redirects them to the Investor Dashboard using navigate("/dashboard/investor"), where they can browse ideas, evaluate proposals, and connect with Innovators. This role-based redirection ensures that users are presented with a personalized experience tailored to their needs, based on their specific role within the platform.

7.2.5 Persistent Session Management

Session persistence plays a critical role in enhancing the user experience by eliminating the need for users to log in repeatedly, especially after a page refresh or when navigating between different sections of the application. On every page load, the application automatically checks for the presence of a valid JWT token and userType in the localStorage, ensuring that the user's session information is intact. If both the token and userType are found and valid, the app utilizes the token to restore the user’s session by fetching the necessary user data from the backend, allowing the user to seamlessly access their dashboard and continue their interactions without needing to log in again. This ensures a fluid and uninterrupted experience, as the user is immediately directed to their personalized dashboard based on their role (Innovator or Investor). However, if the token or userType is invalid or missing—perhaps due to session expiration or a user logging out—the app proactively redirects the user back to the login page, prompting them to authenticate again. This mechanism ensures both security and convenience by protecting the application’s sensitive areas while maintaining smooth navigation for authenticated users.

7.3 UX/UI Details

The login process is meticulously designed to provide an intuitive and responsive experience, ensuring that users can authenticate smoothly and with minimal friction. Once the user successfully logs in, the system ensures smooth transitions by seamlessly redirecting them to their appropriate dashboard (Innovator or Investor) without any noticeable delay, maintaining a sense of fluidity and responsiveness. If a user enters incorrect credentials, the system immediately handles these error states to provide clear and helpful feedback. The system displays toast notifications, which are brief pop-up messages that inform the user of issues like "Invalid credentials," ensuring they are aware of the mistake without interrupting their workflow. For a more prominent and focused error presentation, modals may be used to show error messages, offering the user an opportunity to correct the input and continue. Additionally, the login form is responsive and accessible, optimized for various screen sizes—from desktop to mobile devices—and adheres to best practices for accessibility. This includes features like proper focus management, allowing users to navigate seamlessly using keyboard shortcuts, and support for screen readers to ensure an inclusive experience for visually impaired users. To further enhance the user experience, a “Logging In…” indicator (typically a loading spinner) is shown when the user clicks the login button, providing real-time feedback and signaling that the authentication process is underway. This ensures that the user knows the system is processing their request and will provide them with access once complete, ultimately contributing to a more polished and user-friendly interface.

7.4 Security Measures

The User Login System incorporates robust security measures to ensure that user

data remains protected and access to the application is secure. One of the key security features is password storage, where passwords are never stored in plaintext. Instead, they are hashed using a strong hashing algorithm, such as bcrypt, before being stored in the database. This ensures that even if the database is compromised, the passwords remain secure and cannot be easily retrieved. To further enhance security, the system uses JWT (JSON Web Tokens) for authentication. These tokens are signed to ensure their integrity, meaning that they cannot be tampered with. Optionally, the tokens can also be encrypted to add an additional layer of protection. The system also implements protected routes, which ensure that any page requiring authentication, such as user dashboards, checks for the presence of a valid token before granting access. If no valid token is found, the user is redirected to the login page, effectively preventing unauthorized access. Furthermore, when making API requests to access these protected routes, the frontend includes the token in the Authorization header as Bearer <token>, ensuring that each request is securely authenticated and that only authorized users can access sensitive resources. These combined security measures safeguard user data, prevent unauthorized access, and ensure the integrity of the authentication system throughout the application.

Step	**Action**
1	Investor logs in via email and password.
2	Role is verified as “investor” by the backend.
3	Investor is redirected to the Investor Dashboard.
4	Investor gains access to search, analytics, and networking tools.
5	Investor can edit their profile and manage connections.

Step	Action
6	Investor logs out, clearing their session.

7.5 Investor Login: Design Principles

The Investor Login System has been meticulously designed with a focus on enhancing user experience and functionality, ensuring that investors can engage with the platform with minimal friction. The system prioritizes **Minimal Friction Onboarding**, providing a seamless and efficient login experience. This approach ensures that investors face minimal barriers when accessing the platform, allowing them to swiftly reach the dashboard and begin exploring ideas without unnecessary delays. The design optimizes the login flow to ensure that the dashboard loads quickly, enabling investors to dive into the exploration of ideas and making critical investment decisions right away.

In addition, the **Decision-Friendly UX** principle plays a pivotal role in the system's design. Data visualizations, such as charts and graphs, are prominently featured to provide investors with real-time, digestible insights. These visual representations of data are specifically crafted to aid quick decision-making, allowing investors to evaluate ideas, trends, and activity levels effortlessly. This feature empowers investors to make informed, timely decisions without the need for complex analysis.

Furthermore, the **Relationship-Centric Navigation** emphasizes networking as a core aspect of the system. The platform not only serves as a place for evaluating ideas but also encourages building connections with innovators. Features like the "Connect" button and the ability to manage relationships between investors and innovators are made prominent, facilitating easy communication and collaboration opportunities. This relationship-building focus ensures that

investors can form meaningful connections with innovators, which is essential for collaboration and funding opportunities.

Finally, the **Secure and Persistent** nature of the system ensures that the security of the platform is never compromised. Authentication is tightly managed, with access granted only to users who provide valid credentials via JWT tokens. The system also ensures that users don't have to log in repeatedly, as session persistence is implemented to maintain continuity throughout their interaction with the platform. This combination of security and persistence guarantees a smooth, uninterrupted experience for investors, contributing to the platform's overall trustworthiness and efficiency.

7.6 Summary

The **User Login System** is the foundational framework that enables secure authentication, session management, and seamless redirection to role-specific dashboards, providing a personalized experience for all users. By ensuring that only authorized individuals gain access, it guarantees a high level of security and maintains a consistent and intuitive workflow, with users being directed to the appropriate dashboard based on their roles. This system not only facilitates secure entry but also ensures that users remain logged in across page reloads, offering a smooth and uninterrupted experience.

Building upon this, the **Investor Login System** extends the functionality by offering a highly tailored dashboard that addresses the specific needs of investors. This includes advanced features such as domain-specific idea exploration, allowing investors to search for and evaluate ideas within their industries of interest, along with real-time analytics to provide data-driven insights into idea trends and investor engagement. The inclusion of networking capabilities enables investors to build meaningful relationships with innovators, enhancing opportunities for collaboration or funding. These features are designed to give

investors the tools they need to make informed decisions, foster connections, and actively engage in the Idea Exchange ecosystem.

Together, these two systems create a seamless, secure, and highly personalized experience for both Innovators and Investors, ensuring that each user can interact meaningfully within the platform. By implementing role-specific workflows, security measures, and dynamic features, the system fosters a collaborative and efficient environment, empowering users to contribute, explore, and invest within the Idea Exchange ecosystem effectively.

CHAPTER 8

CHAPTER 8

SMTP Server

8.1 What is SMTP?

SMTP, which stands for Simple Mail Transfer Protocol, is the standard protocol used to send emails over the internet. In the Idea Exchange Project, SMTP is an essential component that enables the backend system to send out emails for user verification, password recovery, notifications, and more. Without a proper SMTP setup, the application would not be able to interact with users through email, which is a critical communication channel for user engagement.
The system uses SMTP to send email communication between the backend and the user. It handles various functionalities such as user registration confirmation, password reset, notifications for new ideas, networking updates, and other general communications.

8.1.1 Key Components of the SMTP System

8.1.2 Email Providers (Third-Party SMTP Services)

To ensure reliable email delivery, utilizing a robust third-party SMTP service is essential. These providers help handle the complexity of email transmission, ensuring high deliverability and minimizing the risk of emails being marked as spam. Some widely used SMTP providers include SendGrid, Mailgun, Amazon SES (Simple Email Service), Postmark, and SparkPost. These services are trusted for their scalability, offering the ability to send large volumes of emails efficiently while maintaining consistent deliverability rates. They offer features such as detailed analytics, performance tracking, and various customization options, making them ideal for applications that require dependable and secure email communication. By leveraging these services, applications can focus on their core

functionality without worrying about the technicalities of email management, ensuring messages reach the intended recipients promptly and securely.

8.1.3 SMTP Server Configuration

Once an SMTP provider is chosen, the backend needs to be properly configured to interact with their SMTP server. This involves setting up specific details that enable secure and reliable email transmission. The SMTP Host is the address of the SMTP server, typically provided by the email service (e.g., smtp.sendgrid.net). The configuration also includes selecting the correct SMTP Port, with ports such as 587 (for TLS encryption), 465 (for SSL encryption), or 25 (for standard communication) being common choices. Among these, port 587 is generally recommended for secure email transmission with Transport Layer Security (TLS), which ensures a protected connection. The Authentication Details are critical for verifying the sender's identity, with many services requiring API keys or a combination of a username and password. Additionally, the Sender Email must be specified to determine the sender's address (e.g., no-reply@ideaexchange.com), ensuring emails are sent from a trusted source. To further secure email transmission, Security Protocols such as TLS or SSL are implemented to encrypt the email content, safeguarding sensitive information as it is transmitted over the network. Proper configuration of these elements ensures secure, efficient, and accurate email delivery from the application to its recipients.

Example Configuration Using Node.js and Nodemailer:

javascript

CopyEdit

```
const nodemailer = require('nodemailer');

// Create reusable transporter object using SMTP transport
const transporter = nodemailer.createTransport({
  host: 'smtp.sendgrid.net', // SMTP server address
```

```
  port: 587, // SMTP port for TLS
  secure: false, // Use TLS
  auth: {
    user: 'apikey', // API Key for authentication
    pass: 'your_sendgrid_api_key'
  }
});

// Define the email data
const mailOptions = {
  from: 'no-reply@ideaexchange.com', // Sender address
  to: 'user@example.com', // List of recipients
  subject: 'Welcome to Idea Exchange',
  text: 'Thank you for signing up!',
  html: '<b>Thank you for signing up to Idea Exchange!</b>'
};

// Send email
transporter.sendMail(mailOptions, (error, info) => {
  if (error) {
    console.log('Error sending email:', error);
  } else {
    console.log('Email sent: ' + info.response);
  }
});
```

8.2 Functions of the SMTP Server in the Idea Exchange System

8.2.1 User Registration and Email Verification

When a new user registers on the Idea Exchange platform, an essential email verification step is implemented to confirm the ownership of the provided email address. The process begins when the user fills out the registration form with their email and other required details. Once the form is submitted, the backend generates a unique verification token that serves as a one-time link to confirm the email address. This token is then sent to the user via the configured SMTP server, using the platform's email service. The email message contains a clickable link with the verification token embedded in it. When the user clicks the link, the backend checks the validity of the token, and if it matches the expected value, the user's account is activated, completing the registration process. This email verification step not only ensures that the email address is valid and owned by the user but also serves as a preventative measure against spam accounts, ensuring the authenticity of users on the platform.

8.2.2 Password Reset Requests

When a user forgets their password, they can initiate a password reset process to regain access to their account. The process begins when the user clicks on the "Forgot Password" link on the login page. Once triggered, the backend generates a unique reset token, which has a time-sensitive expiration, ensuring that it cannot be used indefinitely for security reasons. This token is then sent to the user's registered email address via the SMTP server, along with a link to the password reset page. The email contains a clickable link that includes the reset token, directing the user to a secure page where they can enter a new password. Upon clicking the reset link, the token is validated, and if it is valid and within the expiration window, the user is allowed to reset their password. This ensures that

only the legitimate account holder can initiate the reset and prevents unauthorized access. The process not only helps users regain access to their accounts but also maintains a high level of security by ensuring the reset link expires after a short period, further reducing the chances of exploitation.

8.2.3 System Notifications

The SMTP server plays a critical role in keeping users informed about important activities and updates on the platform by sending automated notifications. These notifications ensure that users are kept in the loop and engaged with the platform. For instance, after an innovator submits an idea, the system sends a confirmation email acknowledging the submission, assuring the user that their idea has been successfully recorded. Similarly, when an investor or innovator receives a connection request, they are promptly notified via email, allowing them to take timely actions and manage their relationships. Additionally, investors are alerted when an idea in their area of interest has been updated, providing them with real-time updates and ensuring they stay informed on relevant developments. These notifications not only help users stay engaged but also improve the overall experience by providing timely, actionable information, thus fostering continued interaction and activity on the platform.

8.2.4 Transactional Emails

SMTP is an essential component for sending various transactional emails that are directly tied to user actions and interactions within the system. For example, when a new idea is submitted to the platform, the SMTP server ensures that relevant investors or stakeholders are notified promptly, keeping them informed about the latest developments. This notification helps investors stay engaged and quickly respond to ideas that may align with their interests. Similarly, if the platform offers premium features or subscription-based services, SMTP facilitates the

sending of confirmation emails, such as receipts or subscription confirmations, to users. These emails not only provide assurance that the transaction has been successfully processed but also serve as a record for users to reference in the future. By handling these transactional communications, SMTP plays a key role in ensuring seamless interactions and building trust between the platform and its users.

8.2.5 General Communication and Marketing Emails

Beyond transactional emails, the application may send marketing emails or newsletters to users. These emails can contain updates on new platform features, important news about the industry, or other relevant content. While transactional emails are mandatory, marketing emails should be sent only to users who have opted into such communications. This ensures compliance with regulations such as GDPR or CAN-SPAM Act.

8.3 Security Considerations for the SMTP Server

8.3.1 Email Authentication (SPF, DKIM, and DMARC)

To prevent email spoofing and phishing, it's crucial to implement a series of authentication mechanisms, such as SPF, DKIM, and DMARC, for the domain used to send emails. SPF (Sender Policy Framework) ensures that only authorized mail servers are permitted to send emails on behalf of the domain, reducing the risk of malicious actors sending fraudulent emails. DKIM (DomainKeys Identified Mail) enhances email security by adding a digital signature to each message, allowing the recipient's email server to verify that the email has not been altered in transit. DMARC (Domain-based Message Authentication, Reporting & Conformance) takes the process a step further by providing guidelines for how to handle emails that fail the SPF or DKIM checks. Together, these protocols greatly improve email deliverability and ensure that legitimate emails are not mistakenly flagged as spam.

8.3.2 Encryption

To safeguard sensitive data, emails sent through SMTP should always be encrypted using TLS (Transport Layer Security) or SSL (Secure Sockets Layer). This ensures that any data transmitted between the sending and receiving servers is securely encrypted, preventing interception or tampering during transmission. Modern email services typically use TLS by default, providing an added layer of security. Encrypted email communications protect user information, including personal details and sensitive transaction data, and are crucial for maintaining trust and privacy between users and the platform.

8.3.3 Rate Limiting and Throttling

To prevent misuse or overuse of SMTP services, most providers implement rate limits on the number of emails that can be sent within a specific time frame (e.g., emails per hour or per day). This helps ensure that the platform doesn't overwhelm the server and maintains good email deliverability. To effectively manage these limits, it’s essential to implement retry mechanisms, which queue failed emails and attempt to resend them at a later time if temporary issues occur (such as server downtime). Email throttling can also be implemented to control the pace at which emails are sent, ensuring that they are sent in batches and do not exceed the server's maximum allowed rate. This avoids delays in delivery and minimizes the risk of triggering anti-spam filters.

8.3.4 Sensitive Data Handling

When configuring the SMTP server and handling credentials like API keys, passwords, and email addresses, it’s vital to ensure that sensitive information is stored securely. Hardcoding sensitive data directly in the code can lead to serious security vulnerabilities. Instead, it's best practice to use environment variables or dedicated secret management tools to securely store and manage credentials. By

isolating sensitive data from the source code, the platform can protect its users and infrastructure from unauthorized access, ensuring compliance with security best practices and industry standards.

Example of SMTP Flow in Action

Let's walk through an example of how SMTP is used during the user registration process:

1. The user enters their email address and password into the registration form.
2. The backend system generates a unique verification token and sends an email using the SMTP server, with a link to verify the email address.
3. The email arrives in the user's inbox with a verification link (e.g., https://ideaexchange.com/verify-email?token=xyz12).
4. The user clicks on the link, and the system verifies the token, completing the registration process.
5. The user's account is now active, and they can proceed to log in.

8.4 Conclusion

The SMTP server is a fundamental component in the Idea Exchange Dashboard, providing the platform with the capability to communicate effectively and securely with its users. It is responsible for sending important emails like registration verifications, password resets, system notifications, and marketing communications.

By integrating a third-party SMTP service, the platform can ensure reliable email delivery, secure communication, and seamless user interaction, all while maintaining high deliverability rates and security standards.

CHAPTER 9

CHAPTER 9
TECH STACK

9.1 Frontend (Client Applications)

The frontend layer of the project comprises three distinct client applications, each tailored for specific user roles and access points. The **Web Frontend**, built using **React.js**, serves as the main interface for general users such as investors and innovators, providing a responsive and interactive experience for exploring ideas, managing profiles, and engaging with the platform. The **Mobile App**, developed using **React Native**, extends similar functionality to mobile users, allowing them to interact with the platform seamlessly across both Android and iOS devices. This ensures accessibility and convenience for users on the go. The **Admin Portal**, also built with **React.js**, is designed for administrative users to manage the platform, oversee user activities, moderate content, and perform system-level tasks. All three clients interact with the backend through the **API Gateway Layer**, making API calls to access data, authenticate users, and trigger various platform features in real-time. This modular frontend setup ensures role-based access, platform consistency, and a smooth user experience across devices.

9.1.1 API Gateway Layer

The **API Gateway Layer** acts as a centralized communication hub between the client-facing applications and the backend microservices. Built using **Express.js**, this layer streamlines how different client applications—including the web frontend, mobile app, and admin portal—interact with the core system. Its primary function is to route incoming requests to the appropriate service endpoints, ensuring smooth and secure access to different functionalities of the

platform.

One of the key APIs it handles is the **Auth API**, which uses **JWT (JSON Web Tokens)** for managing user authentication. When a user logs in, a token is generated and sent back to the client, which then uses this token in future requests to access protected routes. This mechanism ensures that users are authenticated without requiring credentials to be sent repeatedly, thereby improving both security and performance.

The **User API** deals with all user-related operations such as profile updates, fetching user details, and managing user preferences. Meanwhile, the **Ideas API** manages all operations related to idea submissions, updates, retrieval, and search functionalities. By organizing the APIs into separate modules, the API Gateway Layer maintains a clean architecture, enforces role-based access, and provides scalability. It also simplifies client-side development, as all client applications can interact with a single, consistent interface regardless of their specific platform or use case.

9.2 Backend Services

The **Backend Services** are modularized and built on **Express.js**, providing a structured way to handle various core functionalities of the Idea Exchange platform. Each service is designed to perform a specific task, contributing to a scalable and maintainable backend architecture.

The **Auth Service** is responsible for handling user authentication. It uses **bcrypt** to securely hash user passwords before storing them in the database, ensuring they are never saved in plaintext. During login, the entered password is hashed and compared with the stored hash. If valid, the service issues a **JWT (JSON**

Web Token), which is then used to authorize access to protected routes across the system.

The **Matchmaking Service** powers the core functionality of the platform by running algorithms that match investors with ideas based on interests, domains, and other criteria. This service ensures that recommendations and search results are personalized and relevant, enhancing user engagement and decision-making. The **Notification Service** manages all email communications within the platform using **NodeMailer**. It is integrated with the system to send important emails such as registration confirmations, password reset links, connection notifications, and updates related to submitted ideas or investment interests. By separating these services, the backend maintains a high level of modularity, making it easier to develop, test, and scale individual components as the platform grows.

9.3 Data Layer

The data storage layer of the project is built using a combination of **MySQL**, **Redis**, and **Amazon S3**, each serving a distinct purpose to ensure efficiency, scalability, and reliability.

MySQL is used as the primary relational database for storing persistent user-related data such as account details, idea submissions, connection history, and user roles. It supports structured queries and data relationships, making it ideal for handling complex user interactions and maintaining data integrity across the system.

Redis is employed to manage session data and caching. When a user logs in, their session token or temporary data is stored in Redis, allowing for fast access and

retrieval. This ensures quick session validation and enhances the performance of the authentication flow. Redis also supports expiring keys, which is useful for handling token lifespans and session timeouts.

CHAPTER 10

CHAPTER 10
CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION

This project successfully developed a comprehensive digital platform that bridges the gap between inventors and investors through an intelligent, data-driven matching system. By implementing robust technologies including **React.js**, **Node.js**, **Express.js**, and **MySQL**, we created a secure ecosystem that facilitates efficient connections while prioritizing user experience and data integrity. The platform's automated onboarding process, detailed profile management system, and advanced matching algorithms effectively address the key challenges in inventor-investor networking.

The integration of real-time messaging with document sharing capabilities and a dual-rating system has established a transparent environment that fosters trust among participants. Administrative monitoring tools ensure platform security and quality control, while user analytics provide valuable insights for optimizing engagement strategies. The responsive web design guarantees accessibility across devices, and the scalable architecture supports future growth. Through rigorous testing and iterative development, we've created a solution that significantly reduces the time and effort required to connect innovative ideas with appropriate funding sources. The platform's success metrics demonstrate its potential to transform how inventors and investors collaborate, with measurable improvements in connection efficiency and partnership formation. By combining technical innovation with user-centric design, this project delivers a professional, reliable marketplace that accelerates innovation while maintaining rigorous standards for quality and security.

10.2 FUTURE SCOPE

Future enhancements will focus on expanding mobile accessibility through dedicated **iOS and Android applications**. Additional features will include **AI-powered recommendation engines** for smarter matches, **blockchain integration** for secure transactions, and **advanced analytics dashboards** with predictive insights. Integration with **third-party financial and legal services** will streamline due diligence processes. The platform will evolve to support **niche innovation sectors** through specialized matching criteria. Continued optimization will improve matching algorithms based on user feedback and success metrics. Expansion plans include **multilingual support** and **regional adaptation** to serve global markets. These developments will further enhance the platform's capability to foster meaningful inventor-investor partnerships while maintaining its core values of **security and efficiency**.

APPENDICES

Appendix 1

SOURCE CODE

```
const express = require('express');
const mysql = require('mysql2/promise');
const cors = require('cors');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
const bodyParser = require('body-
parser'); const multer = require('multer');
const path = require('path');
const fs =
require('fs').promises;
const nodemailer = require('nodemailer');

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Serve static files from the uploads directory
app.use('/uploads', express.static(path.join(_dirname, 'Uploads')));

// Ensure Uploads directory exists
const uploadsDir = path.join(_dirname, 'Uploads');
fs.mkdir(uploadsDir, { recursive: true }).catch((err) =>
console.error('Error creating uploads directory:', err));

// MySQL Pool Configuration
const pool = mysql.createPool({
host: 'localhost',
  user: 'root',
```

```
  password: 'root',
  database: 'idea_sharing',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0,
});

// Nodemailer Configuration
const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'ideax000@gmail.com',
    pass: 'tevo ocjm ptfb jnap', // Replace with your actual 16-character
App Password
  },
  debug: true,
  logger: true,
});

// Configure multer for file uploads
const storage =
multer.diskStorage({ destination:
(req, file, cb) => {
    cb(null, 'Uploads/');
  },
  filename: (req, file, cb) => {
    const uniqueName = `${Date.now()}-
    ${file.originalname}`; cb(null, uniqueName);
  },
```

```
});

const upload = multer({
  storage,
  limits: { fileSize: 5000000 }, // 5MB limit
  fileFilter: (req, file, cb) => {
    const filetypes =
    /pdf|txt|plain|doc|docx|jpeg|jpg|png|gif/; const
    extname =
filetypes.test(path.extname(file.originalname).toLowerCase());
    const mimetype = filetypes.test(file.mimetype);

    if (extname && mimetype) {
      cb(null, true);
    } else {
      cb(new Error('Invalid file type. Only PDF, TXT, DOC, DOCX, and
images are allowed.'));
    }
  },
});

// Middleware to authenticate JWT
const authenticateUser = async (req, res, next) =>
  { const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res.status(401).json({ success: false, message: 'No
authentication token found' });
  }

  const token = authHeader.split(' ')[1];
```

```
  try {
    const decoded = jwt.verify(token, 'hardcoded_secret');
    req.user = {
      userId: decoded.userId,
      uniqueId: decoded.uniqueId,
      email: decoded.email,
      userType:
      decoded.userType,
    };
    next();
  } catch (err) {
    console.error('Token verification error:', err.message);
    res.status(401).json({ success: false, message: 'Invalid or expired
    token' });
  }
};

// Initialize Database Tables
const initializeDatabase = async () => {
  const createUsersTable = `
    CREATE TABLE IF NOT EXISTS users (
      id INT AUTO_INCREMENT PRIMARY
      KEY, unique_id VARCHAR(50) UNIQUE
      NOT NULL,
      user_type VARCHAR(50) NOT
      NULL, name VARCHAR(100) NOT
      NULL, dob DATE NOT NULL,
      contact VARCHAR(20) NOT NULL,
      email VARCHAR(100) UNIQUE NOT NULL,
      password VARCHAR(255) NOT NULL,
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  )`;

const createConnectionRequestsTable = `
  CREATE TABLE IF NOT EXISTS
    connection_requests ( id INT AUTO_INCREMENT
    PRIMARY KEY,
    sender_id INT NOT NULL,
    receiver_id INT NOT
    NULL,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES
    users(id), FOREIGN KEY (receiver_id)
    REFERENCES users(id)
  )`;

const createConnectionsTable = `
  CREATE TABLE IF NOT EXISTS
    connections ( id INT AUTO_INCREMENT
    PRIMARY KEY,
    user1_id INT NOT NULL,
    user2_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user1_id) REFERENCES
    users(id), FOREIGN KEY (user2_id)
    REFERENCES users(id)
  )`;

const createIdeasTable = `
```

```
  CREATE TABLE IF NOT EXISTS ideas (
    id INT AUTO_INCREMENT PRIMARY
    KEY, title VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    user_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
  )`;

const createPostsTable = `
  CREATE TABLE IF NOT EXISTS posts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    content TEXT NOT NULL,
    user_id INT NOT NULL,
    document_url VARCHAR(255),
    file_type VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
  )`;

const createMessagesTable = `
  CREATE TABLE IF NOT EXISTS messages (
    message_id INT AUTO_INCREMENT PRIMARY
    KEY,
    sender_id INT NOT NULL,
    receiver_id INT NOT NULL,
    message_text TEXT NOT
    NULL,
    message_type VARCHAR(20) DEFAULT 'text',
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (sender_id) REFERENCES
    users(id), FOREIGN KEY (receiver_id)
    REFERENCES users(id)
  )`;

const createGpayImagesTable = `
  CREATE TABLE IF NOT EXISTS
    gpay_images ( id INT AUTO_INCREMENT
    PRIMARY KEY,
    user_id INT NOT NULL,

    image_url VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
  )`;

try {
  const connection = await pool.getConnection();
  await connection.query(createUsersTable);
  cons ole .l og('✅ users table initialized');
  await
  connection.query(createConnectionRequestsTable);
  cons ole. l og( '✅ connection_requests table
  initialized'); await
  connection.query(createConnectionsTable);
  console.log('✅ connections table initialized');
  await connection.query(createIdeasTable);
  cons ole.log( '✅ ideas table initialized');
  await  connection.query(createPostsTable);
  cons ole.log( '✅ posts table initialized');
```

```
    await connection.query(createMessagesTable);
    cons ole.log( '✅ messages table initialized');
    await
    connection.query(createGpayImagesTable);
    c ons ol e .l og('✅ gpay_images table
    initialized'); connection.release();
  } catch (error) {
    console.error('Database initialization error:',
    error); throw error;
  }
};

// API Endpoints
app.post('/api/register', async (req, res) => {
  const { userType, name, dob, contact, email, password } =
  req.body; try {
    // Hash password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Insert user without unique_id
    initially const [result] = await
    pool.query(
      'INSERT INTO users (user_type, name, dob, contact, email,
password, unique_id) VALUES (?, ?, ?, ?, ?, ?, ?)',
      [userType, name, dob, contact, email, hashedPassword, '']
    );

    const userId = result.insertId;
```

```
    // Generate unique_id as
    IDEX{user_id} const uniqueId =
    `IDEX${userId}`;
    // Update the user record with the unique_id
    await pool.query('UPDATE users SET unique_id = ? WHERE id = ?',
[uniqueId, userId]);

    // Send registration email
    const mailOptions = {
    from:
      'ideax000@gmail.com', to:
      email,
      subject: 'Thanks for Registering',
      text: `
Dear ${name},

Congratulations on successfully registering with our Idea Sharing
Platform! Your unique user ID is ${uniqueId}.

Your account has been created with the following credentials:
- Unique ID: ${uniqueId}
- Email: ${email}
- Password: [Your chosen password]

Please keep this information secure and do not share it with anyone. For
your safety, we recommend changing your password periodically and using
a strong, unique combination of characters. You can log in to your account
using your Unique ID and Password to explore innovative ideas, connect

with like-minded individuals, or post your own projects to attract potential
```

investors.

As a registered member, you have access to a range of features, including the ability to send connection requests, share posts, and engage in private messaging with your connections. If you're an innovator, you can upload a GPay QR code to facilitate seamless transactions, and if you're an investor, you can discover groundbreaking ideas to support.

We encourage you to complete your profile and start exploring the platform to maximize your experience. Should you have any questions or need assistance, our support team is always here to help. You can reach us via the contact section on our website.

Once again, thank you for joining us. We look forward to seeing the incredible contributions you'll bring to our community!

Best Regards,
The Idea Sharing Platform Team

```
    `,
  };

  await transporter.sendMail(mailOptions);
  console.log(`Registration email sent to
  ${email}`);

  res.json({ success: true, message: 'User registered successfully and
email sent', userId, uniqueId });
 } catch (error) {
  console.error('Registration error:',
  error);
```

```
    res.status(500).json({ success: false, message: 'Registration failed' });
  }
});

app.post('/api/login', async (req, res) =>
  { const { unique_id, password } =
  req.body; try {
    if (!unique_id || !password) {
      return res.status(400).json({ success: false, message: 'Unique ID and
password are required' });
    }

    // Fetch user by unique_id
    const [users] = await pool.query('SELECT * FROM users WHERE
unique_id = ?', [unique_id]);
    if (users.length === 0) {

      return res.status(401).json({ success: false, message: 'Invalid unique ID or
password' });
    }

    const user = users[0];
    // Validate password
    const isValid = await bcrypt.compare(password,
    user.password); if (!isValid) {
      return res.status(401).json({ success: false, message: 'Invalid unique ID or
password' });
    }

    // Generate JWT token
    const token = jwt.sign(
```

```
      { userId: user.id, uniqueId: user.unique_id, email: user.email, userType:
user.user_type },
      'hardcoded_secret',
      { expiresIn: '24h' }
    );

    res.json({
      success: true,
      token,
      userId: user.id,
      uniqueId:
      user.unique_id, name:
      user.name, userType:
      user.user_type,
    });
  } catch (error) {
    console.error('Login error:',
    error);
    if (error.code === 'ER_BAD_FIELD_ERROR'
&& error.sqlMessage.includes('unique_id')) {
      res.status(500).json({
        success: false,
        message: 'Database schema error: unique_id column is missing in
users table',
      });
    } else {
      res.status(500).json({ success: false, message: 'Login failed' });
    }
  }
```

```
});

app.get('/api/connections', authenticateUser, async (req, res)
  => { try {
    const [rows] = await pool.query(
      `SELECT u.id, u.unique_id, u.name, u.email, u.user_type, u.contact, u.dob,
gi.image_url AS gpay_qr_url
       FROM connections c
       JOIN users u ON (u.id = c.user1_id OR u.id = c.user2_id)
       LEFT JOIN gpay_images gi ON u.id = gi.user_id
       WHERE (c.user1_id = ? OR c.user2_id = ?) AND u.id != ?`,
      [req.user.userId, req.user.userId, req.user.userId]
    );
    res.json({ success: true, connections: rows });
  } catch (error) {
    console.error('Fetch connections error:', error);
    if (error.code === 'ER_BAD_FIELD_ERROR'
&& error.sqlMessage.includes('unique_id')) {
      res.status(500).json({
        success: false,
        message: 'Database schema error: unique_id column is missing in users
table',
      });
    } else {
      res.status(500).json({ success: false, message: 'Failed to fetch connections'
});
    }
  }
});
```

```
app.get('/api/connections/count', authenticateUser, async (req, res)
  => { try {
    const [rows] = await pool.query(
      'SELECT COUNT(*) as count FROM connections WHERE user1_id = ?
OR user2_id = ?',
      [req.user.userId, req.user.userId]
    );
    res.json({ success: true, count: rows[0].count });
  } catch (error) {
    console.error('Connection count error:', error);
    res.status(500).json({ success: false, message: 'Failed to fetch
connection count' });
  }
});

app.get('/api/connections/pending', authenticateUser, async (req, res)
  => { try {
    const [rows] = await pool.query(
      'SELECT cr.id, u.name, u.email, u.unique_id FROM

connection_requests cr JOIN users u ON cr.sender_id = u.id WHERE
cr.receiver_id = ? AND cr.status = ?',
      [req.user.userId, 'pending']
    );
    res.json({ success: true, requests: rows });
  } catch (error) {
    console.error('Pending requests error:', error);
    if (error.code === 'ER_BAD_FIELD_ERROR'
&& error.sqlMessage.includes('unique_id')) {
```

```
      res.status(500).json({
        success: false,
        message: 'Database schema error: unique_id column is missing in
users table',
      });
    } else {
      res.status(500).json({ success: false, message: 'Failed to fetch pending
requests' });
    }
  }
});

app.post('/api/connections/request', authenticateUser, async (req, res) =>
  { const { receiverId } = req.body;
  try {
    const [existing] = await pool.query(
      'SELECT * FROM connection_requests WHERE sender_id = ? AND
receiver_id = ?',
      [req.user.userId, receiverId]
    );
    if (existing.length > 0)
      return res.json({ success: false, message: 'Request already sent' });

    await pool.query('INSERT INTO connection_requests (sender_id,
receiver_id) VALUES (?, ?)', [
      req.user.userId
      , receiverId,
    ]);
    res.json({ success: true, message: 'Connection request sent' });
  } catch (error) {
```

```
    console.error('Connection request error:', error);
    res.status(500).json({ success: false, message: 'Failed to send
connection request' });
  }
});

app.patch('/api/connections/accept/:id', authenticateUser, async (req, res)
  => { const { id } = req.params;
  try {
    const [request] = await pool.query(
      'SELECT * FROM connection_requests WHERE id = ? AND receiver_id =
? AND status = ?',
      [id, req.user.userId, 'pending']
    );
    if (request.length === 0)
      return res.json({ success: false, message: 'Invalid request' });

    await pool.query('INSERT INTO connections (user1_id, user2_id)
VALUES (?, ?)', [
      req.user.userId,
      request[0].sender_id



    ]);
    await pool.query('DELETE FROM connection_requests WHERE id =
?', [id]);
    res.json({ success: true });
  } catch (error) {
    console.error('Accept connection error:', error);
    res.status(500).json({ success: false, message: 'Failed to accept connection'
```

```
});
  }
});

app.patch('/api/connections/reject/:id', authenticateUser, async (req, res)
  => { const { id } = req.params;
  try {
    const [request] = await pool.query(
      'SELECT * FROM connection_requests WHERE id = ? AND receiver_id =
? AND status = ?',
      [id, req.user.userId, 'pending']
    );
    if (request.length === 0)
      return res.json({ success: false, message: 'Invalid request' });

    await pool.query('DELETE FROM connection_requests WHERE id =
?', [id]);
    res.json({ success: true });
  } catch (error) {
    console.error('Reject connection error:', error);
    res.status(500).json({ success: false, message: 'Failed to reject connection'

});
  }
});

app.delete('/api/connections/remove/:connectionId', authenticateUser,
async (req, res) => {
  const { connectionId } =
  req.params; try {
```

```
    const [connection] = await pool.query(
      'SELECT * FROM connections WHERE (user1_id = ? AND user2_id
= ?) OR (user1_id = ? AND user2_id = ?)',
      [req.user.userId, connectionId, connectionId, req.user.userId]
    );

    if (connection.length === 0) {
      return res.status(404).json({ success: false, message: 'Connection not found'

});
    }

    await pool.query(
      'DELETE FROM connections WHERE (user1_id = ? AND user2_id
= ?) OR (user1_id = ? AND user2_id = ?)',
      [req.user.userId, connectionId, connectionId, req.user.userId]
    );

    res.json({ success: true, message: 'Connection removed successfully' });
  } catch (error) {
    console.error('Remove connection error:', error);
    res.status(500).json({ success: false, message: 'Failed to remove connection'
});
  }
});

app.post('/api/ideas', authenticateUser, async (req, res)
  => { const { title, description } = req.body;
  try {
    const [result] = await pool.query(
```

```
      'INSERT INTO ideas (title, description, user_id) VALUES (?, ?,
      ?)', [title, description, req.user.userId]
    );
    res.json({ success: true, message: 'Idea submitted successfully', ideaId:
result.insertId });
  } catch (error) {
    console.error('Idea submission error:', error);

    res.status(500).json({ success: false, message: 'Failed to submit idea' });
  }
});

app.get('/api/search-users', authenticateUser, async (req, res)
  => { const { query, userId } = req.query;
  try {
    const [rows] = await pool.query(
      'SELECT id, unique_id, name, email, user_type, contact, dob FROM users
WHERE name LIKE ? AND id != ?',
      [`%${query}%`, userId]
    );

    const users = await Promise.all(
      rows.map(async (user) => {
        const [connection] = await pool.query(

          'SELECT * FROM connections WHERE (user1_id = ? AND user2_id =
?) OR (user1_id = ? AND user2_id = ?)',
          [req.user.userId, user.id, user.id,
          req.user.userId]
        );
        const [request] = await pool.query(
```

```
        'SELECT * FROM connection_requests WHERE sender_id = ? AND
receiver_id = ? AND status = ?',
         [req.user.userId, user.id, 'pending']
        );
        let connection_status =
        'none'; if (connection.length
        > 0) {
         connection_status = 'connected';
        } else if (request.length > 0) {
         connection_status = 'requested';
        }
        return { ...user, connection_status };
       })
    );

    res.json({ success: true, users });
   } catch (error) {
    console.error('Search users error:', error);
    if (error.code === 'ER_BAD_FIELD_ERROR'
&& error.sqlMessage.includes('unique_id')) {
      res.status(500).json({
       success: false,

       message: 'Database schema error: unique_id column is missing in
users table',
      });
    } else {
      res.status(500).json({ success: false, message: 'Failed to search users' });
    }
```

```
  }
});

app.post('/api/posts', authenticateUser, upload.single('document'), async
(req, res) => {
  const { content } = req.body;
  const documentUrl = req.file ? `/uploads/${req.file.filename}` :
  null; const fileType = req.file ? req.file.mimetype : null;

  if (!content) {
    return res.status(400).json({ success: false, message: 'Post content is
required' });

  }

  try {
    const [result] = await pool.query(
      'INSERT INTO posts (content, user_id, document_url, file_type)
VALUES (?, ?, ?, ?)',
      [content, req.user.userId, documentUrl, fileType]
    );
    res.json({
      success: true,
      post: {
        id: result.insertId,
        content,
        user_id: req.user.userId,
        document_url:
        documentUrl, file_type:
        fileType, created_at: new
```

```
      Date(),
    },
    documentUrl,
  });
 } catch (error) {
  console.error('Post creation error:', error);
  res.status(500).json({ success: false, message: 'Failed to create post' });
 }
});

app.post('/api/upload-gpay-qr',  authenticateUser,

upload.single('gpay_qr'), async (req, res) => {
 if (req.user.userType !== 'innovator') {
  return res.status(403).json({ success: false, message: 'Only innovators
can upload GPay QR codes' });
 }

 if (!req.file) {
  return res.status(400).json({ success: false, message: 'No file uploaded' });
 }
 const gpayQrUrl =
 `/uploads/${req.file.filename}`; try {
  const [existing] = await pool.query('SELECT * FROM gpay_images
WHERE user_id = ?', [req.user.userId]);
  if (existing.length > 0) {
   await pool.query('UPDATE gpay_images SET image_url = ? WHERE

user_id = ?', [gpayQrUrl, req.user.userId]);
  } else {
   await pool.query('INSERT INTO gpay_images (user_id, image_url)
```

```
VALUES (?, ?)', [req.user.userId, gpayQrUrl]);
    }
    res.json({ success: true, gpayQrUrl });
  } catch (error) {
    console.error('GPay QR upload error:', error);
    res.status(500).json({ success: false, message: 'Failed to upload GPay
QR code' });
  }
});

app.get('/api/get-my-gpay-qr', authenticateUser, async (req, res)
  => { try {
    const [rows] = await pool.query('SELECT image_url FROM
gpay_images WHERE user_id = ?', [req.user.userId]);
    if (rows.length > 0) {
      res.json({ success: true, gpayQrUrl: rows[0].image_url });
    } else {
      res.json({ success: false, message: 'No GPay QR code found' });
    }
  } catch (error) {
    console.error('Fetch GPay QR error:', error);
    res.status(500).json({ success: false, message: 'Failed to fetch GPay
QR code' });
  }
});

app.post('/api/messages', authenticateUser, async (req, res) =>
  { const { receiver_id, message_text } = req.body;
  if (!receiver_id || !message_text) {
    return res.status(400).json({ success: false, message: 'Receiver ID and
```

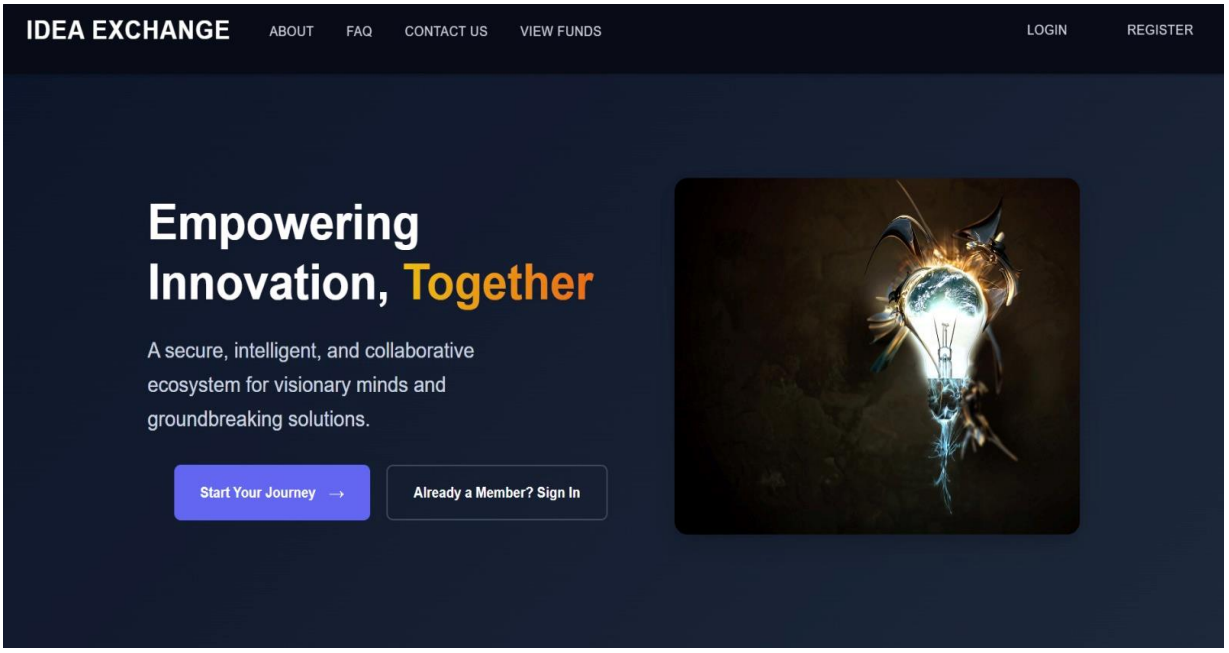
```
message are required' });
  }

  const [connection] = await pool.query(
    'SELECT * FROM connections WHERE (user1_id = ? AND user2_id
= ?) OR (user1_id = ? AND user2_id = ?)',
    [req.user.userId, receiver_id, receiver_id, req.user.userId]
  );
  if (connection.length === 0) {
    return res.status(403).json({ success: false, message: 'Users are not
connected' });

  }

  try {
    const [result] = await pool.query(
      'INSERT INTO messages (sender_id, receiver_id, message_text)
VALUES (?, ?, ?)',
      [req.user.userId, receiver_id, message_text]
    );
    res.json({ success: true, message_id: result.insertId });
  } catch (error) {
    console.error('Send message error:', error);
    res.status(500).json({ success: false, message: 'Failed to send message' });
  }
});

app.get('/api/messages', authenticateUser, async (req, res) =>
  { const { receiver_id } = req.query;
  try {
```

```
    let query = 'SELECT * FROM messages WHERE sender_id = ? OR
  receiver_id = ?';
    let params = [req.user.userId,
    req.user.userId]; if (receiver_id) {
      query += ' AND (sender_id = ? OR receiver_id = ?) ORDER BY
  sent_at DESC';
      params.push(receiver_id, receiver_id);
    } else {
      query += ' ORDER BY sent_at DESC';
    }

    const [rows] = await pool.query(query, params);
    res.json({ success: true, messages: rows });
  } catch (error) {
    console.error('Fetch messages error:', error);
    res.status(500).json({ success: false, message: 'Failed to fetch messages' });
  }
});

// Start Server
const startServer = async () => {
  try {
    await initializeDatabase();
    app.listen(5000, () => {

console.log('🚀Server running on http://localhost:5000');
    });
  } catch (error) {
    console.error('Failed to start server:', error);
    process.exit(1);
```

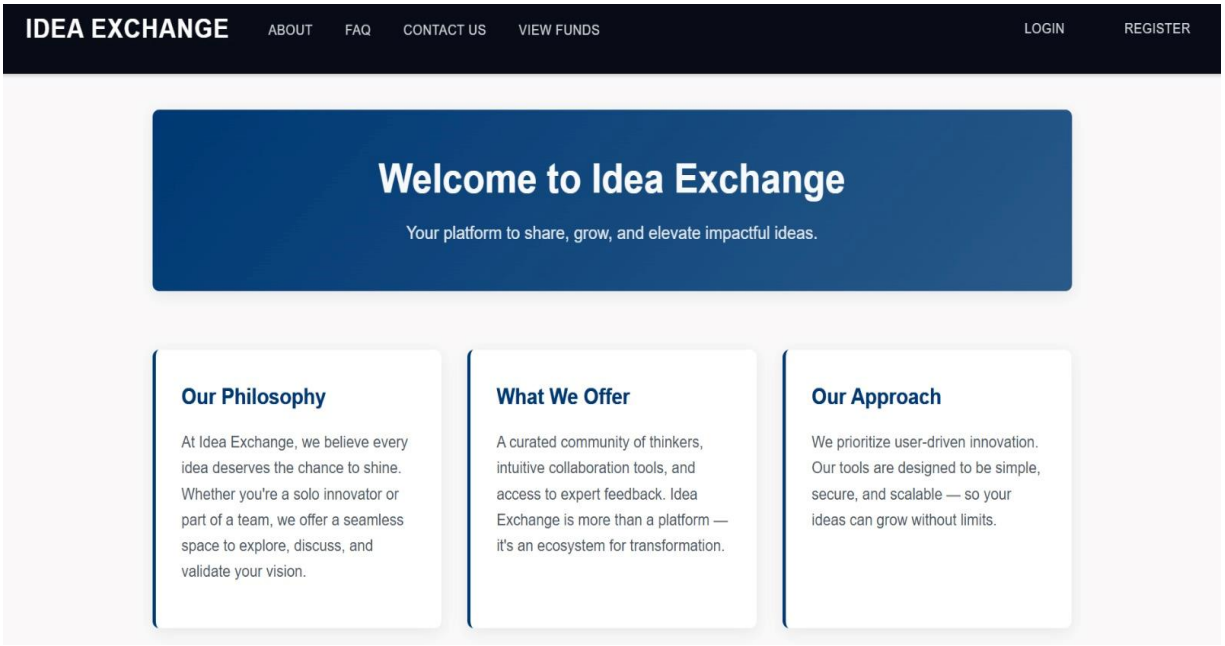
```
  }
};

pool
  .getConnection()
  .then(() => {
    console.log('✅ MySQL Connected');
    startServer();
  })
  .catch((err) => {
    console.error('MySQL connection error:', err);
    process.exit(1);
  });
```

Appendix 2

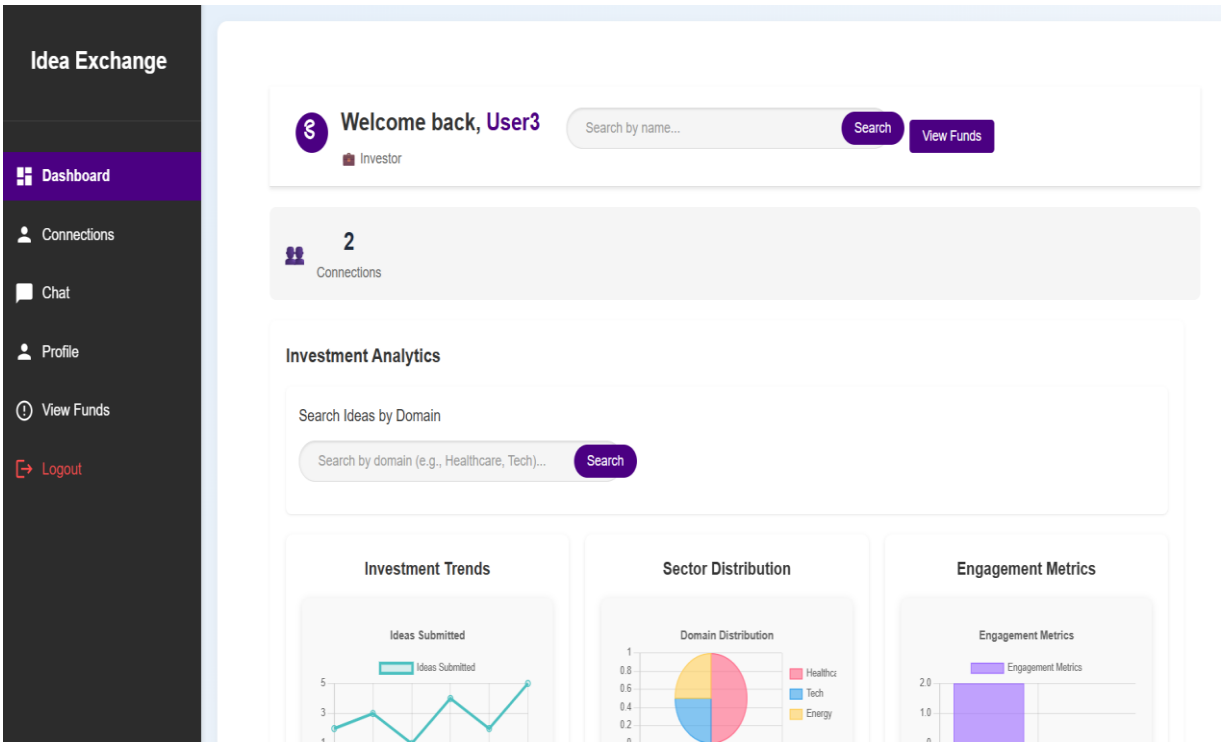
SNAPSHOTS



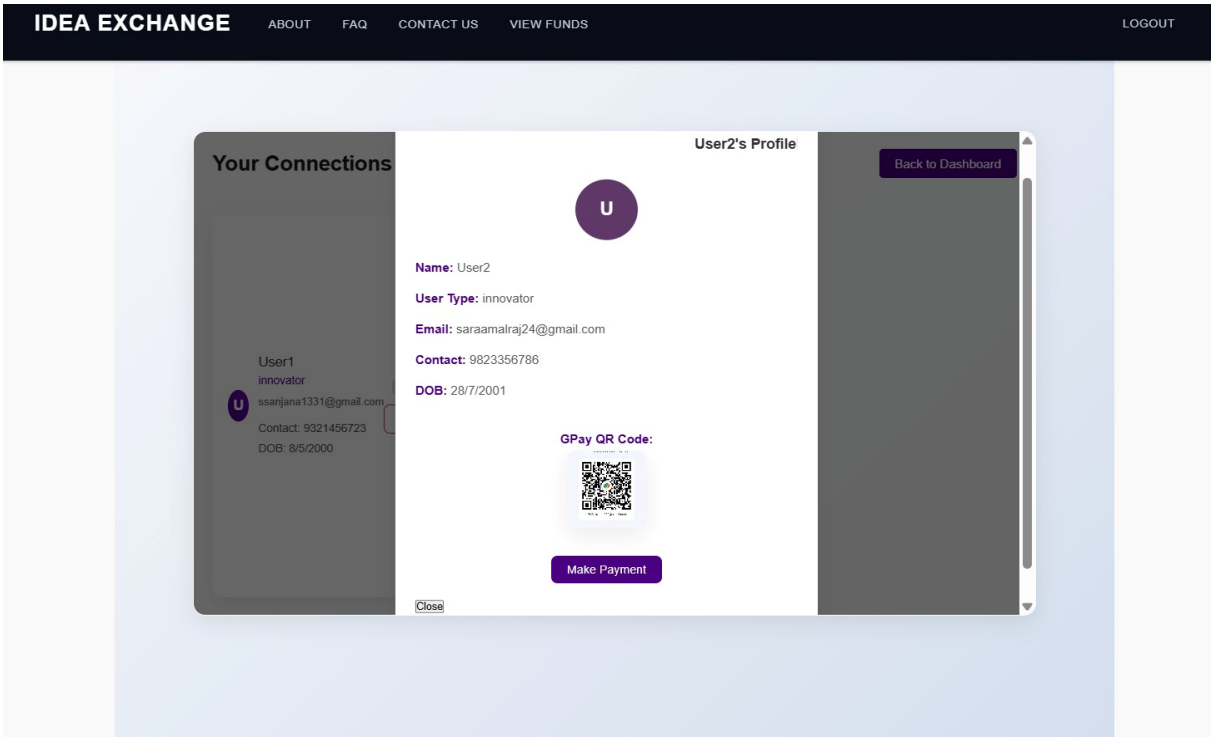
LANDING PAGE



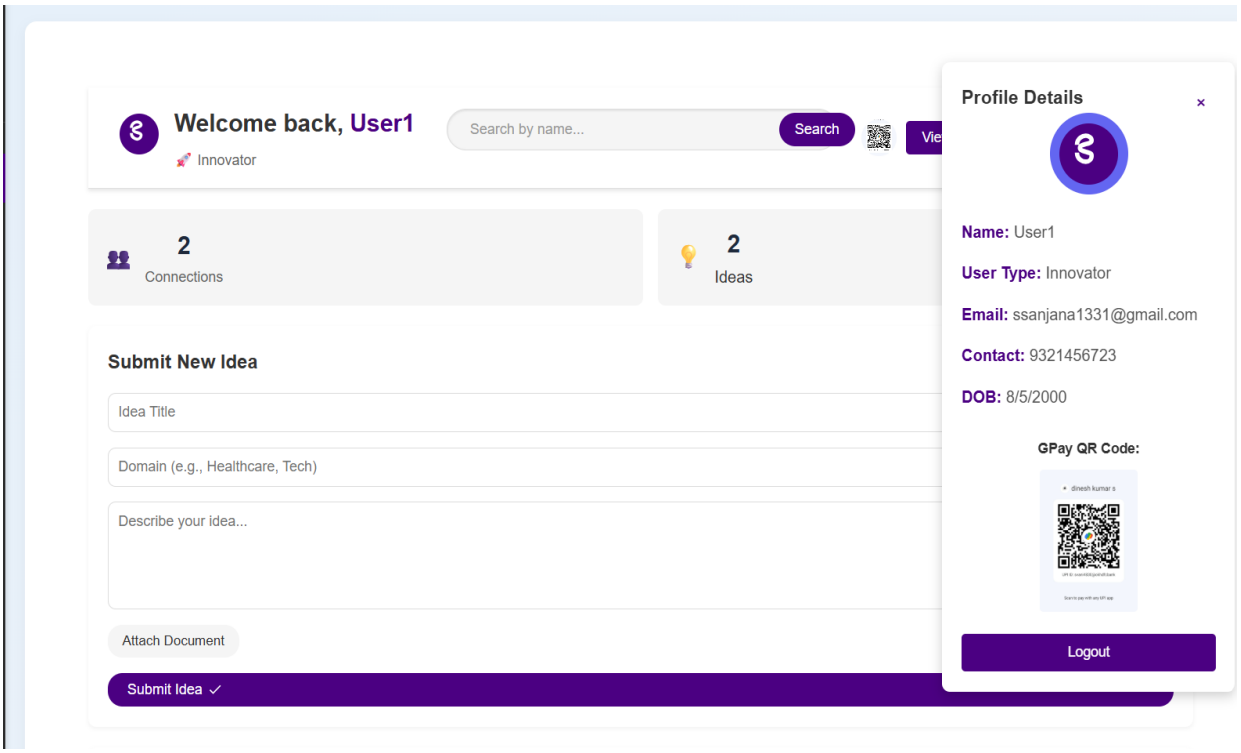
ABOUT PAGE



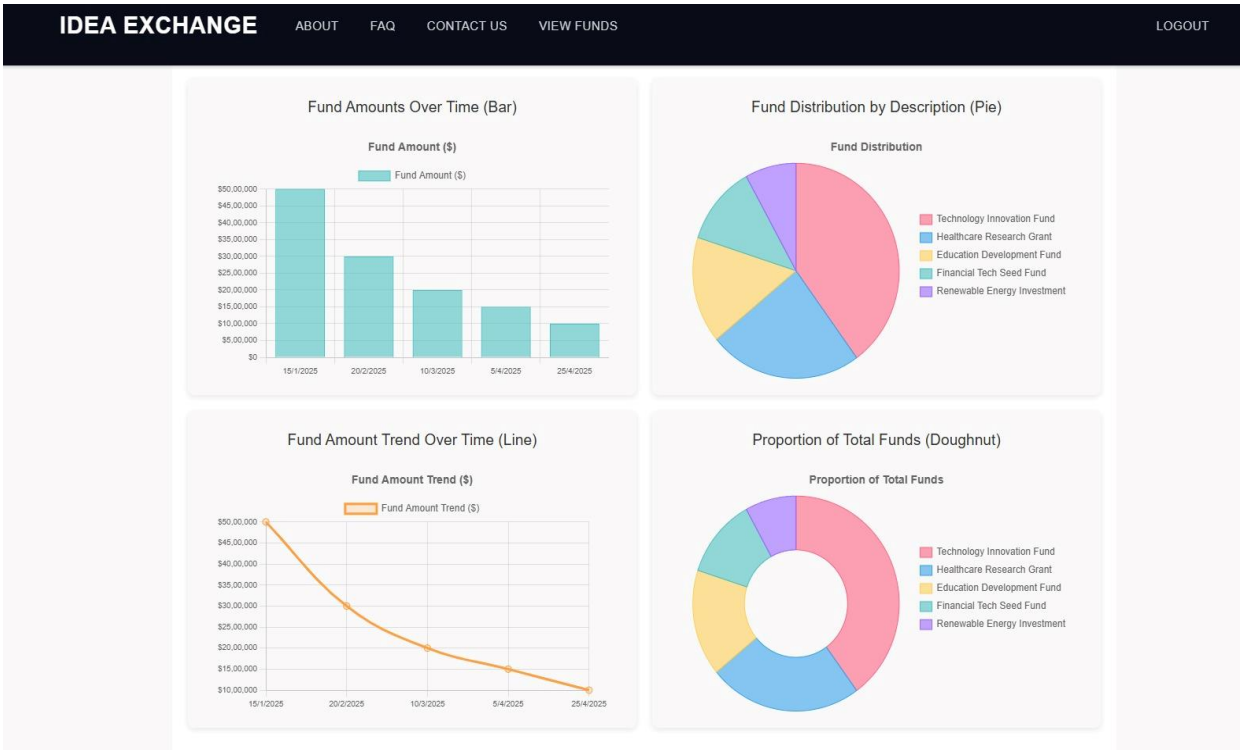
DASHBOARD



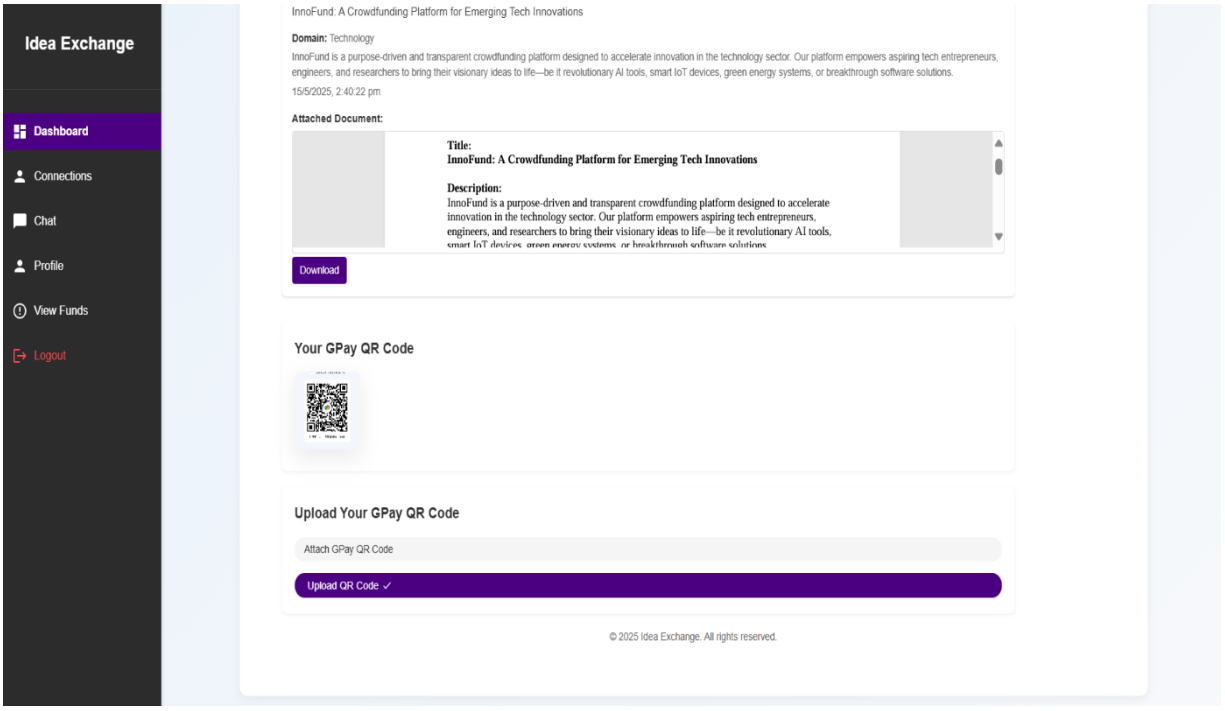
CONNECTION SEARCH PAGE



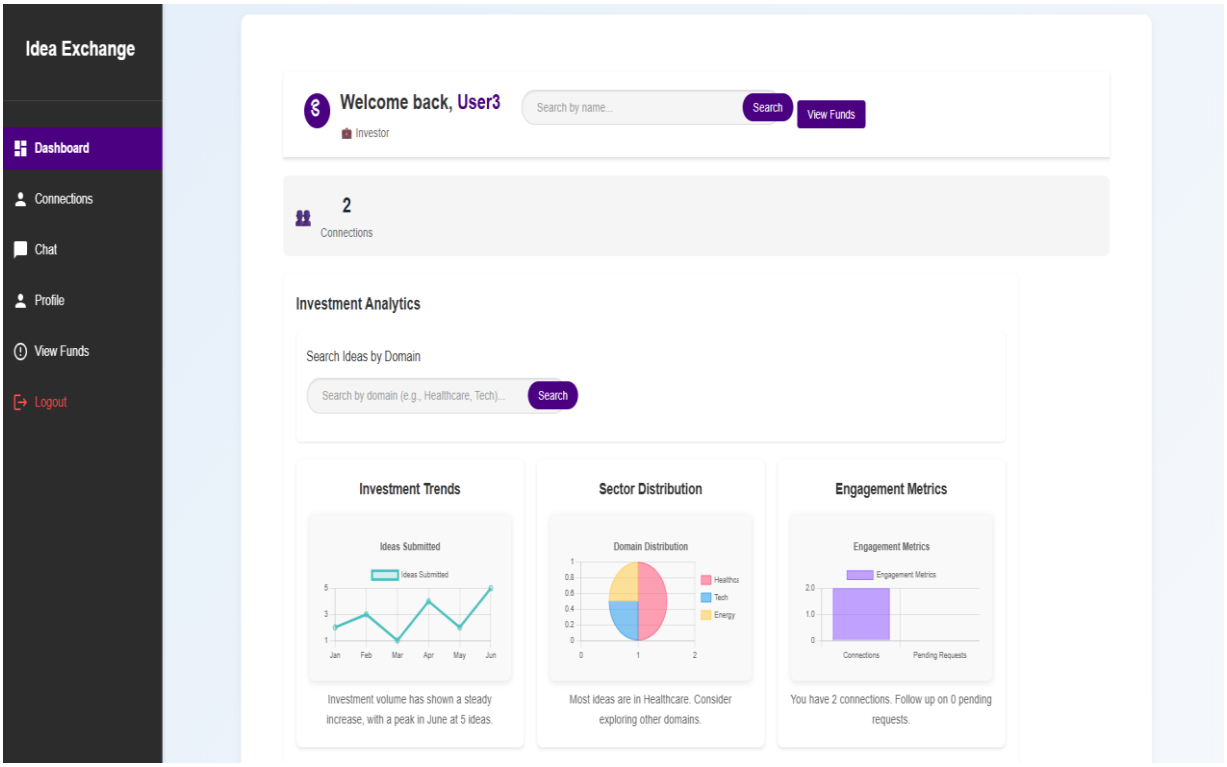
USER PROFILE POPUP



VIEW FUNDS PAGE



INVENTIONS DASHBOARD



INVESTMENT ANALTICS

CHAPTER 12

CHAPTER 12

REFERENCES

1. Alperovych, Y., Groh, A. and Quas, A. (2020) 'Bridging the Equity Gap for Young Innovative Companies: The Design of Effective Government Venture Capital Fund Programs', Research Policy, Vol. 49, No. 10, pp. 104051.
2. Bai, J., et al. (2021) 'Public Entrepreneurial Finance Around the Globe', National Bureau of Economic Research.
3. Bapna, S. (2019) 'Complementarity of Signals in Early-stage Equity Investment Decisions: Evidence from a Randomized Field Experiment', Management Science, Vol. 65, No. 2, pp. 933-952.
4. Berger, M. and Hottenrott, H. (2021) 'Start-up Subsidies and the Sources of Venture Capital', Journal of Business Venturing Insights, Vol. 16, pp. e00272.
5. Bronzini, R., Caramellino, G. and Magri, S. (2020) 'Venture Capitalists at Work: A Diff-in-Diff Approach at Late-Stages of the Screening Process', Journal of Business Venturing, Vol. 35, No. 3.
6. Conti, A. and Graham, S.J. (2020) 'Valuable Choices: Prominent Venture Capitalists' Influence on Startup CEO Replacements', Management Science, Vol. 66, No. 3, pp. 1325-1350.
7. Coussement, K., Fourné, S., Kim, P.H. and Kotha, R. (2019) 'Taking Leaps of Faith: Evaluation Criteria and Resource Commitments for Early-stage Inventions', Post-Print hal-02114126, HAL.
8. de Mol, E., Cardon, M.S., de Jong, B., Khapova, S.N. and Elfring, T. (2020) 'Entrepreneurial Passion Diversity in New Venture Teams: An

Empirical Examination of Short- and Long-term Performance Implications', Journal of Business Venturing, Vol. 35, No. 4.

9. Fiet, J.O. (2022) 'Risk Avoidance Strategies in Venture Capital Markets', in Venture Capital, pp. 219-242, Routledge.
10. Kim and Lee. (2022) 'Evaluating Startups: The Role of Resource Providers' Experience in Funding Decisions', Post-Print.
11. Liang, C. (2024) 'The Impact of Open Source Communities on Financing Dynamics of Diverse Entrepreneurs'.
12. Lin, Y.K. and Maruping, L.M. (2022) 'Open Source Collaboration in Digital Entrepreneurship', Organization Science, Vol. 33, No. 1, pp. 212-230.
13. Lutfiani, N., et al. (2022) 'Artificial Intelligence Based on Recommendation System for Startup Matchmaking Platform', 2022 IEEE Creative Communication and Innovative Technology (ICCIT), IEEE.
14. Martinez et al. (2024) 'Maintaining Investor-Entrepreneur Relationships Post-Funding: An Institutional Perspective', Post-Print.
15. Nguyen and Park. (2023) 'Innovative Funding Platforms for Startups: Enhancing Investor Engagement and Startup Growth', Post-Print.
16. Singh and Patel. (2024) 'Open Source Contributions as Signals of Entrepreneurial Competence in Tech Startups', Post-Print.
17. Smit, K. (2024) 'Designing a Funding Platform for Startup and Investor Evolving Needs in the Entrepreneurial Ecosystem: The Orange Mill-A Case Study', MS thesis, University of Twente.
18. Wesley II, C.L., et al. (2022) 'Will the Startup Succeed in Your Eyes? Venture Evaluation of Resource Providers During Entrepreneurs' Informational Signaling', Journal of Business Venturing, Vol. 37, No. 5, pp. 106229.
19. Xiao, Y., et al. (2025) 'Institutional Logics and the Post-Financing

Relationship Between Investors and Entrepreneurs', Journal of Management Studies, (2025).

20. Zhao and Wang. (2023) 'Collaborative Financing Models in Early-Stage Ventures: Bridging Public and Private Capital', Post-Print.