

Software Engineering Department  
ORT Braude College

Capstone Project Phase A – 61998

**TaskTenders**

An Application that Connects Users with Freelance Services

**Authors:**

Safa Khier – Safa.Khier@e.braude.ac.il

Layan Amer – Layan.Amer@e.braude.ac.il

**Supervisor:**

Dr. Zakharia Frenkel

**24-2-D-42**

**Table of Contents**

[**Abstract** 4](#_Toc176990290)

[**1. Introduction** 5](#_Toc176990291)

[**2. Related Work** 6](#_Toc176990292)

[*2.1 Current Demand for Freelancer Services* 6](#_Toc176990293)

[*2.2 Benefits and Challenges of Freelance Platforms* 6](#_Toc176990294)

[*2.3 Risks and Security Concerns in Freelancer Platforms* 6](#_Toc176990295)

[*2.4 Existing Apps that Offer Freelance Services* 6](#_Toc176990296)

[**3. Background** 8](#_Toc176990297)

[*3.1 Project Origin and Inspiration* 8](#_Toc176990298)

[*3.2 Key Features of Successful Freelancer Platforms* 8](#_Toc176990299)

[*3.3 Case Study: Fiverr* 10](#_Toc176990300)

[*3.4 Lessons Learned and Applied in TaskTenders* 10](#_Toc176990301)

[**4. Expected Achievements** 11](#_Toc176990302)

[*4.1 Outcomes* 11](#_Toc176990303)

[*4.2 Unique Features* 11](#_Toc176990304)

[*4.3 Criteria for Success* 11](#_Toc176990305)

[**5. The Process** 12](#_Toc176990306)

[*5.1 Research – User Needs Analysis* 12](#_Toc176990307)

[*5.2 Methodology and Development Process* 12](#_Toc176990308)

[*5.3 Development Phases* 12](#_Toc176990309)

[**6. Product** 13](#_Toc176990310)

[*6.1 Our Solution* 13](#_Toc176990311)

[*6.2 Requirements* 15](#_Toc176990312)

[*6.2.1 Functional Requirements* 15](#_Toc176990313)

[*6.2.2 Non-Functional Requirements* 16](#_Toc176990314)

[*6.3 System Functionality* 17](#_Toc176990315)

[*6.3.1 Client Functionality* 17](#_Toc176990316)

[*6.3.2 Freelancer Functionality* 18](#_Toc176990317)

[*6.3.3 Administrator Functionality* 18](#_Toc176990318)

[*6.4 Technology Choices* 20](#_Toc176990319)

[*6.4.1* *Back-end: Node.js* 20](#_Toc176990320)

[*6.4.2* *Back-end: TypeScript* 21](#_Toc176990321)

[*6.4.3 Database: MongoDB* 23](#_Toc176990322)

[*6.4.4 Front-end: Flutter* 25](#_Toc176990323)

[*6.4.5 Development Environment: Visual Studio Code* 26](#_Toc176990324)

[*6.5 Project Structure* 28](#_Toc176990325)

[*6.6 Algorithms* 31](#_Toc176990326)

[*6.7 Diagrams* 33](#_Toc176990327)

[*6.7.1 use case diagram* 33](#_Toc176990328)

[*6.7.2 Activity diagram* 33](#_Toc176990329)

[*6.8 Tests* 33](#_Toc176990330)

[*6.8.1 Login Tests* 33](#_Toc176990331)

[*6.8.2 User Tests* 34](#_Toc176990332)

[*6.8.3 Freelancer Tests* 34](#_Toc176990333)

[*6.8.4 Admin Tests* 35](#_Toc176990334)

[**7. GUI** 35](#_Toc176990335)

[*7.1 User Screens* 35](#_Toc176990336)

[*7.2 Freelancer Screens* 35](#_Toc176990337)

[*7.3 Admin Screens* 35](#_Toc176990338)

[**8. References** 35](#_Toc176990339)

# **Abstract**

There is an increasing demand for a platform that links freelancers to individuals who need various services. This situation is caused by the changing gig economy. TaskTenders aims at bridging this gap through its user-friendly mobile application which connects service providers with users seeking to get tasks done more effectively. For instance, it allows users to post job requests such as home repairs, cleaning or painting giving details of what they need and when they need it. Consequently, freelancers can look through these job postings in order to submit their bids and offer their services at competitive rates.

This platform exploits the latest technology to enable smooth communications between users and freelancers. Furthermore, it has strong functionalities like user registration and secure authentication among others. Customers can articulate what they need, while workers can present their skills and expertise in full detail. A safe payment gateway guarantees secure financial transactions; while a within app messaging system enhances lucidity between the involved parties. This mobile application also contains a method of rating and reviewing services delivered by freelancers as well as keeping up high service standards and promoting trustworthiness within its society.

TaskTenders also has the administrators panel where you can manage all user activities, check all transactions and handle disputes. The main aim here is to create an operational, dependable and secure ecosystem that will enhance user satisfaction thereby promoting gig economy. By providing this platform, TaskTenders enables users to find competent freelancers who suit their requirements while at the same time granting freelancers access to numerous job opportunities. This project is developed using a modern tech stack, including React Native for the frontend, Node.js and Express.js for the backend, and MongoDB for data management. Expected completion date of the application is 7th September; this will be after creating a comprehensive solution that bridges between freelance workers and clients.

# **1. Introduction**

In the dynamic landscape of the gig economy, the demand for reliable and skilled freelance workers is burgeoning. Individuals and businesses alike seek efficient and cost-effective solutions to fulfill their varied service needs, ranging from home repairs and cleaning to specialized tasks like wall painting. However, connecting these users with competent freelancers remains a significant challenge. Traditional methods of finding and hiring freelancers can be time-consuming, unreliable, and often lack transparency.

Exploring the intricacies of this problem, users often face difficulties in verifying the credibility and quality of freelancers. Miscommunications, inconsistent service quality, and concerns about secure transactions further complicate the process. On the other hand, freelancers struggle to find a steady stream of job opportunities that match their skills and availability. The absence of a streamlined platform that bridges this gap leaves both parties underserved and their needs unmet.

Given these challenges, we propose an innovative solution in the form of TaskTenders, a mobile application designed to connect users with freelance workers efficiently and securely. TaskTenders allows users to post detailed job requests, specifying their requirements, deadlines, and budget. Freelancers can then browse these postings and submit bids, offering their services at competitive rates. The application's advanced matching algorithm ensures that users are paired with freelancers who meet their specific needs and preferences.

TaskTenders incorporates robust features to enhance user experience and build trust. A secure authentication system safeguards user data, while a comprehensive review and rating system ensures accountability and maintains high service standards. The in-app messaging feature facilitates clear communication between users and freelancers, reducing the potential for misunderstandings. Furthermore, an integrated payment gateway ensures secure and seamless financial transactions.

By leveraging modern technology, TaskTenders aims to streamline the process of finding and hiring freelancers, providing a reliable platform where users can easily access skilled professionals, and freelancers can find job opportunities that suit their skills and schedules. This application not only simplifies the connection between service seekers and providers but also contributes to the growth and efficiency of the gig economy. Through TaskTenders, we aspire to foster a more interconnected and productive community, addressing the evolving needs of both users and freelancers in today's fast-paced world.

# **2. Related Work**

## *2.1 Current Demand for Freelancer Services*

## *2.2 Benefits and Challenges of Freelance Platforms*

## *2.3 Risks and Security Concerns in Freelancer Platforms*

## *2.4 Existing Apps that Offer Freelance Services*

**1. Fiverr:**

Function Details: Fiverr is an online marketplace where freelancers offer a wide range of services (called "gigs") across various categories like graphic design, digital marketing, writing, video editing, programming, and more. Clients can browse these gigs and hire freelancers to complete specific projects or tasks.

How it Works:

* Freelancers create profiles and list their services, set their prices, and outline what they offer.
* Clients search for services they need, compare freelancer profiles, read reviews, and select a freelancer whose skills and price fit their requirements.
* Once the client places an order, they communicate with the freelancer to finalize details. The freelancer completes the task, delivers the work through the platform, and the client reviews and accepts it.
* Payment is made through Fiverr, which holds the funds in escrow until the work is approved by the client. Fiverr takes a commission from each transaction.

Examples of Tasks:

* **Graphic Design:** Logo design, business cards, social media graphics
* **Writing & Translation:** Blog posts, articles, translations, proofreading
* **Digital Marketing:** SEO services, social media marketing, email marketing
* **Video & Animation:** Explainer videos, video editing, animations
* **Programming & Tech:** Website development, app development, tech support
* **Music & Audio:** Voice-over, music production, podcast editing

Pros and Cons:

Pros:

1. **Wide Range of Services:** Offers a vast variety of services across multiple categories.
2. **Affordable Options:** Services available at different price points to fit various budgets.
3. **Global Talent Pool:** Access to freelancers from around the world.
4. **Easy to Use:** User-friendly interface for both freelancers and clients.
5. **Escrow System:** Ensures payment security for both parties.

Cons:

1. **Quality Variability:** Quality of work can vary greatly among freelancers.
2. **Fees:** Fiverr takes a 20% commission from freelancers and charges additional fees for clients.
3. **Communication Barriers:** Working with freelancers from different time zones and cultures can sometimes lead to communication issues.
4. **Service Limitations:** Some freelancers may not offer revisions or additional support.
5. **Platform Dependency:** Freelancers are reliant on Fiverr for their client base, which can be limiting.

Usage:

* **Client Base:** Millions of users worldwide, including individuals, small businesses, and large corporations.
* **Availability:** Fiverr operates globally, providing services to clients and freelancers in numerous countries and languages.

Images of Fiverr application:

In Summary...

# **3. Background**

## *3.1 Project Origin and Inspiration*

The idea for **TaskTenders** arose from observing the growing demand for efficient and reliable freelance services in the gig economy. With the increasing number of individuals and small businesses seeking flexible, skilled workers for various tasks such as home repairs, cleaning, and other professional services, it became evident that the traditional methods of finding and hiring freelancers are often inefficient. Users are often left with unreliable service providers, while freelancers struggle to consistently find job opportunities that match their skills and schedules.

**Inspiration for the Project** came from analyzing existing freelance platforms and identifying key pain points faced by both service seekers and freelancers. Many platforms are plagued with issues such as poor communication, lack of transparency in the bidding process, and unreliable payment systems. Additionally, users often have to navigate multiple platforms or rely on informal networks to find freelancers for their specific needs, which further complicates the process.

**TaskTenders** was conceptualized to provide a solution to these issues by creating a platform where users can easily post jobs, freelancers can bid on tasks, and the selection process is transparent and secure. The inspiration also stems from the desire to empower both users and freelancers by giving them a reliable, streamlined platform to connect, negotiate deals, and complete tasks efficiently.

By addressing these issues, TaskTenders seeks to bridge the gap between users who need quality services and freelancers looking for consistent work opportunities. The project's vision is to make freelance service engagement easier, more transparent, and accessible to a broader audience, contributing to the evolving gig economy in a meaningful way.

## *3.2 Key Features of Successful Freelancer Platforms*

Successful freelancer platforms have revolutionized the gig economy by providing reliable, user-friendly solutions that efficiently connect service providers and seekers. The following key features are essential in creating a successful platform that both users and freelancers can trust:

1. **User-Friendly Interface:**

* One of the core elements of a successful platform is an intuitive and easy-to-navigate interface. Both users and freelancers should be able to post jobs, bid on tasks, and communicate effortlessly. A streamlined interface encourages user engagement and fosters a positive experience.

1. **Efficient Matching System:**

* A successful freelancer platform incorporates an advanced matching algorithm that connects users with freelancers based on specific criteria, such as location, skills, and availability. By automating the matching process, users can quickly find qualified freelancers, and freelancers are matched with jobs that fit their expertise.

1. **Transparent Bidding and Payment System:**

* The bidding process should be transparent, allowing users to compare offers based on price, freelancer ratings, and previous work history. Equally important is a secure payment gateway that ensures seamless transactions between users and freelancers. Many platforms integrate escrow systems to protect both parties until the job is completed satisfactorily.

1. **Profiles and Ratings:**

* Successful platforms offer comprehensive freelancer profiles where freelancers can showcase their skills, experience, and portfolio. Additionally, users and freelancers can provide ratings and reviews after each completed job. This system builds trust within the platform by allowing future users to make informed decisions based on past performance.

1. **In-App Messaging and Communication:**

* Real-time communication is essential for discussing project details, clarifications, and updates. In-app messaging facilitates direct and secure communication between users and freelancers, helping to avoid misunderstandings and ensure that both parties are on the same page.

1. **Secure Authentication and User Verification:**

* Successful platforms prioritize user security by implementing strong authentication mechanisms, such as two-factor authentication and verified profiles. This helps prevent fraud, ensures that users and freelancers are legitimate, and enhances trust across the platform.

1. **Escalation and Dispute Resolution:**

* An essential feature of any freelancer platform is the ability to handle disputes or conflicts. Many successful platforms have built-in dispute resolution mechanisms, providing users with a way to escalate issues and resolve them fairly and efficiently.

1. **Mobile Accessibility:**

* In today’s digital age, mobile accessibility is crucial for the success of any freelancer platform. A well-developed mobile app enables users and freelancers to manage their jobs and bids on the go, increasing engagement and accessibility.

1. **Task Categorization and Filtering:**

* Successful platforms allow users to easily categorize tasks (e.g., home repair, graphic design, web development) and provide robust filtering options. This makes it easier for users to find specific freelancers, and for freelancers to locate relevant jobs based on their skills.

1. **Support for Multiple Payment Methods:**

* Flexibility in payment options is key. Successful platforms support multiple payment methods such as credit cards, PayPal, and bank transfers, making it convenient for users to pay and freelancers to receive their earnings.

By integrating these key features, successful freelancer platforms can foster a strong user base, ensure high-quality services, and establish trust between users and freelancers. These features will be vital in the development of **TaskTenders**, ensuring its relevance and competitiveness in the freelance marketplace.

## *3.3 Case Study: Fiverr*

Fiverr is a leading freelance platform, known for its unique service-based marketplace model. Launched in 2010, Fiverr allows freelancers to offer "gigs" – pre-packaged services with clear pricing – making it easy for users to browse and purchase services without the need for bids or negotiations.

**Key Features of Fiverr:**

1. Service-Based Marketplace: Freelancers create detailed gigs with fixed prices, simplifying the hiring process for users.
2. Diverse Categories: Fiverr offers a wide range of services, from graphic design to programming, catering to various needs.
3. Transparent Pricing: Each gig lists its price upfront, and freelancers can offer different service tiers for flexibility.
4. Profile and Review System: Detailed freelancer profiles and a user review system ensure trust and transparency.
5. Secure Payment System: Fiverr uses an escrow payment model, protecting both users and freelancers.
6. Mobile Accessibility: Fiverr’s mobile app enables users and freelancers to manage tasks and communicate on the go.

Fiverr’s success lies in its simplicity, transparency, and global reach, offering valuable insights for the development of TaskTenders.

## *3.4 Lessons Learned and Applied in TaskTenders*

From analyzing successful platforms like Fiverr, several key lessons have been identified and are being applied to the development of **TaskTenders** to ensure its success in connecting users with freelancers efficiently.

1. **Simplified Service Offerings:**
   * Like Fiverr, TaskTenders will adopt a clear and simple service posting model, allowing users to post tasks with clear descriptions and budgets. Freelancers can offer services and bid on tasks in an organized manner, making the process straightforward for both parties.
2. **Transparent Bidding and Pricing:**
   * TaskTenders will implement a transparent bidding system, enabling users to compare offers based on price, reviews, and freelancer profiles. This ensures that users can make informed decisions, while freelancers can showcase their value competitively.
3. **Secure Payment System:**
   * Drawing inspiration from Fiverr's escrow model, TaskTenders will integrate a secure payment gateway, where funds are held until the job is completed to the satisfaction of both parties. This approach will foster trust and reduce the risk of disputes.
4. **Freelancer Profiles and Reviews:**
   * Like Fiverr, TaskTenders will feature detailed freelancer profiles that display skills, experience, and reviews from previous jobs. This will help build credibility and give users confidence when selecting a freelancer.
5. **Mobile Accessibility:**
   * Ensuring that TaskTenders is fully functional on mobile devices is crucial, just as Fiverr has done. This will allow users and freelancers to manage tasks, communicate, and track progress easily, even while on the go.
6. **Freelancer Progression System:**
   * A tiered system for freelancers, similar to Fiverr’s, will be introduced in TaskTenders, allowing top-performing freelancers to gain visibility and access to more job opportunities based on their performance and customer feedback.

By applying these lessons from Fiverr, TaskTenders aims to create a user-friendly, transparent, and secure platform that addresses the pain points of the current freelance marketplace while offering a streamlined solution for both users and freelancers.

# **4. Expected Achievements**

## *4.1 Outcomes*

## *4.2 Unique Features*

## *4.3 Criteria for Success*

# **5. The Process**

## *5.1 Research – User Needs Analysis*

## *5.2 Methodology and Development Process*

This section outlines the overall approach to managing and executing the development of TaskTenders, including the chosen development methodology and rationale.

* **Development Methodology**:
  + **Agile Development**: TaskTenders uses an Agile development methodology to allow for iterative development, continuous feedback, and flexibility to adapt to changing user needs or market conditions.
  + **Scrum Framework**: Implementation of the Scrum framework to facilitate frequent sprints, allowing the team to focus on delivering specific features or functionalities in short cycles.
* **Tools and Technologies**:
  + **Project Management Tools**: Use of tools like Jira for tracking progress, managing sprints, and logging issues.
  + **Version Control System**: Adoption of Git for version control, hosted on platforms like GitHub to facilitate collaborative coding and code review.

## *5.3 Development Phases*

The development of TaskTenders is divided into distinct phases, each focusing on different aspects of the platform:

* **Phase 1: Planning and Design**
  + **Activities**: Requirement gathering, system architecture design, wireframing, and prototyping.
  + **Deliverables**: Project plan, design documents, and initial prototypes.
* **Phase 2: Development**
  + **Activities**: Coding of the core functionalities for both the frontend and backend, based on the designs approved in Phase 1.
  + **Deliverables**: Working software with core functionalities such as user registration, job posting, bidding, and messaging.
* **Phase 3: Testing and Iteration**
  + **Activities**: Comprehensive testing including unit tests, integration tests, and user acceptance testing (UAT). Feedback from the initial user group is integrated into the development.
  + **Deliverables**: Test reports, final adjustments based on feedback, and preparation for deployment.
* **Phase 4: Deployment and Maintenance**
  + **Activities**: Deployment of the platform to a production environment. Continuous monitoring and regular updates based on user feedback and system performance.
  + **Deliverables**: Deployed application, maintenance schedules, and ongoing support plans.

# **6. Product**

## *6.1 Our Solution*

**TaskTenders** revolutionizes the freelancing market by providing a robust, intuitive, and secure platform that connects clients and freelancers across various service categories. Our solution is crafted to tackle common challenges in the freelancing ecosystem such as job matching precision, transparent bidding processes, and secure payment transactions. Here’s how **TaskTenders** stands out:

**Comprehensive Job Matching**

**TaskTenders** employs advanced algorithms to match freelancers with job postings, ensuring that clients receive bids from the most qualified freelancers. This matching is based on several criteria including skill set, past job performance, and client feedback. This targeted matching system not only enhances the relevance of job matches but also increases the chances of job satisfaction for both parties.

**Transparent Bidding Mechanism**

Our platform introduces a transparent and fair bidding system where freelancers can submit bids for job postings. Clients can view detailed profiles of bidding freelancers, including their ratings, previous work, and reviews. This open system helps clients make informed decisions while ensuring freelancers are fairly compensated.

**Secure Payment Gateway**

**TaskTenders** integrates a secure payment system that holds funds in escrow until job completion, safeguarding both clients and freelancers. Payments are only released when both parties confirm that the job has been completed to their satisfaction. This mitigates the risk of non-payment and disputes, creating trust and reliability within the platform.

**Real-Time Communication Tools**

The platform features built-in messaging and notification systems that allow seamless communication between clients and freelancers. These tools facilitate clear and continuous dialogue from the initial job posting to final delivery, enhancing collaboration and efficiency.

**Mobile and Cross-Platform Accessibility**

Designed using **Flutter**, **TaskTenders** offers a responsive and user-friendly interface accessible on both iOS and Android devices. This cross-platform accessibility ensures that users can manage their projects and respond to job updates from anywhere, at any time.

**Robust Admin Dashboard**

For platform administrators, **TaskTenders** provides a powerful dashboard that offers insights into user activity, financial transactions, and system health. Admins can manage user accounts, resolve disputes, and generate detailed reports, all from a single interface.

**Benefits of Our Solution**

* **Increased Efficiency**: Automated job matching and real-time updates streamline the process from job posting to completion.
* **Enhanced Security**: Advanced security protocols and escrow payment systems protect user information and funds.
* **Greater Market Reach**: Easy access through mobile apps expands the user base, providing more opportunities for freelancers and a broader selection of talent for clients.
* **Improved User Experience**: Intuitive design and comprehensive support services ensure user satisfaction and platform loyalty.

**TaskTenders** is not just a freelancing platform; it is a complete ecosystem designed to optimize the interaction between those who need services and those who provide them. By addressing key pain points such as payment security, job matching accuracy, and user experience, **TaskTenders** sets a new standard in the freelancing market.

## *6.2 Requirements*

### *6.2.1 Functional Requirements*

Functional requirements specify what the system should do. They describe the specific behaviors, functions, and capabilities that **TaskTenders** must offer:

1. **User Account Management**
   * Users can create, edit, and delete their accounts.
   * Users can recover forgotten passwords through a secure process.
   * Users can update profile information including skills, contact details, and preferences.
2. **Job Posting and Management**
   * Clients can post new jobs with specific details such as job description, required skills, budget, and deadlines.
   * Clients can edit, close, or delete their job postings.
   * System must provide a search and filter mechanism for finding specific jobs.
3. **Bidding System**
   * Freelancers can browse job listings and submit bids.
   * Freelancers can withdraw or modify their bids before acceptance.
   * Clients can view all bids, compare them, and accept the most suitable one.
4. **Payment Processing**
   * Integration with secure payment gateways to handle transactions.
   * Implementation of an escrow system to hold payments until job completion.
   * Provide detailed transaction histories for users.
5. **Messaging and Notifications**
   * Real-time messaging capability between clients and freelancers.
   * Automated notifications for job updates, bid statuses, and payment alerts.
6. **Admin Functions**
   * Admin dashboard for managing user accounts, job posts, and dispute resolution.
   * Ability to generate reports on user activity, financial data, and platform usage.
   * Tools for monitoring and managing platform security.

### *6.2.2 Non-Functional Requirements*

Non-functional requirements define how the system operates at a higher level. They are crucial for ensuring usability, reliability, performance, and security:

1. **Performance**
   * The system should load any user interaction within 2 seconds.
   * Handle up to 10,000 simultaneous users without performance degradation.
2. **Reliability**
   * The platform should be available 99.9% of the time.
   * Implement robust error handling and data recovery processes to prevent data loss.
3. **Usability**
   * User interfaces must be intuitive and accessible to users with varying levels of technical proficiency.
   * Support for mobile devices and cross-platform accessibility ensuring consistent user experience.
4. **Security**
   * Compliance with data protection regulations such as GDPR.
   * All user data must be encrypted at rest and in transit.
   * Regular security audits and vulnerability assessments to safeguard platform integrity.
5. **Scalability**
   * The system should be designed to easily scale up to accommodate growing numbers of users and data.
   * Use of cloud services to dynamically allocate resources based on demand.

## *6.3 System Functionality*

### *6.3.1 Client Functionality*

Clients on **TaskTenders** are primarily job posters who seek freelancing services. Their core functionalities include:

1. **Account Management**
   * Register, log in, and manage account settings.
   * Customize profile information including contact details, payment methods, and preferences.
2. **Job Management**
   * Post new jobs with detailed descriptions, required skills, budget, and deadlines.
   * Edit, close, or delete job postings as necessary.
   * View and manage applications and bids from freelancers.
3. **Communication**
   * Use in-app messaging to communicate with freelancers about job specifics.
   * Receive notifications about job status updates, bids, and messages.
4. **Payment and Invoicing**
   * Make secure payments through the integrated payment system.
   * View transaction histories and receive invoices for completed jobs.
5. **Feedback and Ratings**
   * Rate and review freelancers after the completion of a job.
   * Receive recommendations based on past hiring and feedback.

### *6.3.2 Freelancer Functionality*

Freelancers are the service providers who bid on job postings. Their functionalities are designed to facilitate finding and completing jobs:

1. **Account Management**
   * Register, log in, and customize their profiles with skills, qualifications, and work portfolios.
2. **Job Search and Bidding**
   * Browse and search job listings.
   * Submit bids for job postings with proposed terms and rates.
   * Withdraw or modify bids before acceptance.
3. **Project Management**
   * Track ongoing projects and deadlines.
   * Submit work directly through the platform for client approval.
4. **Communication**
   * Communicate with clients through in-app messaging for clarifications or negotiations.
   * Receive real-time notifications about job updates and communications.
5. **Financial Management**
   * Manage payments received via the secure escrow system.
   * Access financial records and transaction histories.
6. **Ratings and Feedback**
   * Provide feedback on client cooperation.
   * Build a reputation through client ratings and reviews.

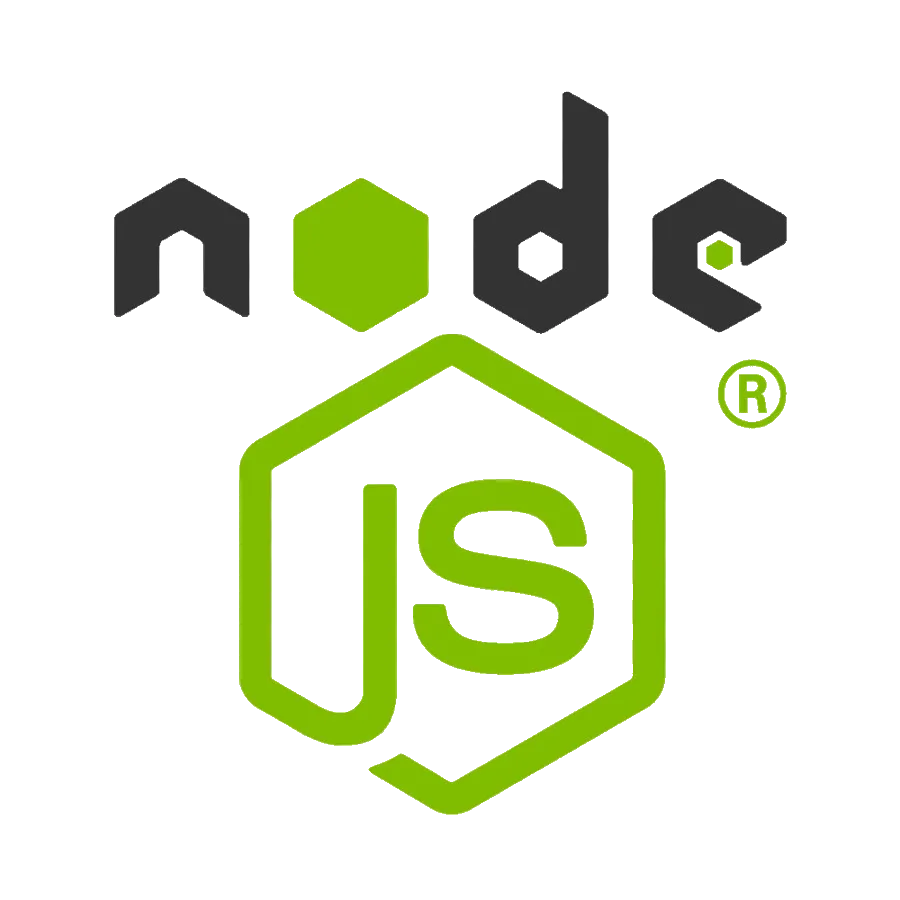
### *6.3.3 Administrator Functionality*

System administrators manage and maintain the platform, ensuring smooth operation and compliance with regulations:

1. **User Management**
   * Monitor and manage user accounts.
   * Handle user verifications and approve or deny registrations.
2. **Content Moderation**
   * Review and manage job postings and bids for adherence to platform policies.
   * Moderate user interactions to ensure compliance with community guidelines.
3. **System Monitoring and Reports**
   * Access detailed analytics and reports on user activity, financial transactions, and system performance.
   * Monitor system health and manage backups, updates, and security protocols.
4. **Dispute Resolution**
   * Mediate disputes between clients and freelancers.
   * Make decisions based on evidence from communication logs and job documentation.
5. **Policy and Compliance Management**
   * Update and enforce platform policies.
   * Ensure the platform complies with legal standards and data protection regulations.

## *6.4 Technology Choices*

### *6.4.1* *Back-end: Node.js*

Node.js is an open-source, cross-platform JavaScript runtime environment that enables developers to execute JavaScript on the server side. Built on Chrome's V8 JavaScript engine, Node.js is designed for efficiency and scalability, making it an ideal choice for building network applications. Its event-driven, non-blocking I/O model allows Node.js to handle a large number of concurrent connections with minimal overhead, making it particularly well-suited for real-time, data-intensive applications like TaskTenders.

In TaskTenders, Node.js powers the server-side logic, handling tasks such as job postings, bids, and real-time communication. Additionally, TypeScript is used in conjunction with Node.js to add static typing, improving code reliability, scalability, and maintainability.

**Node.js advantages:**

1. High Scalability: Node.js excels at managing numerous concurrent connections, making it ideal for real-time applications like messaging, live updates, and bidding in TaskTenders.
2. Fast Performance: Powered by the V8 engine, Node.js executes JavaScript code at high speeds, resulting in better performance for web applications.
3. Non-blocking I/O: The event-driven, non-blocking architecture enables Node.js to handle multiple requests without waiting for operations to complete, increasing efficiency.
4. Single Language (JavaScript/TypeScript): Node.js allows developers to use JavaScript (or TypeScript) for both frontend and backend development, promoting code consistency and easier collaboration between different layers of the stack.
5. Large Ecosystem: The Node.js ecosystem, supported by npm (Node Package Manager), provides access to thousands of open-source libraries and modules, speeding up development and adding flexibility.
6. Real-time Capabilities: Node.js is highly efficient for real-time applications like chat systems, push notifications, and live bidding, all essential features of TaskTenders.
7. RESTful APIs: Node.js is a great choice for building lightweight, fast RESTful APIs that enable smooth communication between the frontend (mobile app) and backend.

**Node.js disadvantages:**

1. CPU-Intensive Tasks: Node.js’s single-threaded nature makes it less efficient for handling heavy computational tasks, which can slow down the event loop.
2. Callback Hell: The asynchronous nature of Node.js can lead to complicated nested callbacks, though modern JavaScript features like async/await mitigate this problem.
3. Maturity of Modules: Although Node.js has a vast ecosystem, some third-party libraries and modules may be less stable or mature than those in more established languages.
4. Single Thread Limitations: While Node.js is excellent for I/O operations, it may face performance issues for CPU-heavy tasks, as it operates on a single thread.

**Express.js Framework**

Node.js is paired with Express.js, a minimal and flexible web application framework that simplifies the development of APIs and web applications. Express.js provides robust routing, middleware, and request-handling capabilities, making it easy to build a scalable and maintainable backend for TaskTenders.

**Database Integration**

Node.js works seamlessly with various databases, both relational and NoSQL. For TaskTenders, we use MongoDB, a NoSQL database, which offers the flexibility required to handle dynamic, unstructured data such as job postings, user profiles, and bids. MongoDB’s document-based structure makes it ideal for handling large volumes of complex data in a scalable manner.

**Real-time Communication with Socket.io**

Node.js integrates with Socket.io to provide real-time, bi-directional communication between the client and the server. This is crucial for features like instant messaging, notifications, and live bidding within TaskTenders, allowing users and freelancers to interact in real-time without delays.

**Node.js for RESTful APIs**

Node.js is well-suited for building lightweight, fast, and scalable RESTful APIs. In TaskTenders, these APIs enable efficient communication between the mobile app and the server, handling tasks like user authentication, job postings, bidding, and payment processing. The non-blocking architecture ensures that requests and responses are handled quickly and efficiently, improving the overall user experience.

### *6.4.2* *Back-end: TypeScript*

A blue square with white letters

Description automatically generatedTypeScript is a statically-typed superset of JavaScript that enhances the development process by adding static typing, interfaces, and advanced programming features to JavaScript. In the TaskTenders project, TypeScript is used alongside Node.js to ensure more reliable and maintainable server-side development. TypeScript’s type safety helps developers catch errors early in the development phase, leading to fewer bugs in production and making the codebase easier to scale as the project grows.

TypeScript integrates seamlessly with JavaScript and the Node.js runtime, allowing developers to build and maintain a robust backend while leveraging JavaScript’s flexibility. The TypeScript compiler converts TypeScript code into JavaScript, ensuring compatibility with the Node.js ecosystem while adding the benefits of strong typing and enhanced tooling.

**TypeScript advantages:**

1. **Static Typing:** TypeScript introduces static types to JavaScript, allowing developers to define the types of variables, function parameters, and return values. This helps catch potential bugs during development rather than at runtime, improving the overall quality of the code.
2. **Early Error Detection:** With TypeScript, errors related to type mismatches or undefined behaviors are caught at compile time. This reduces the likelihood of runtime errors, resulting in more stable applications.
3. **Improved Code Readability and Maintainability:** By explicitly stating types, TypeScript makes the code more readable and easier to understand. This is particularly beneficial when working in teams or on large codebases, as it provides clear expectations about the data and behavior of functions.
4. **Enhanced Tooling:** TypeScript provides better support for IDE features like autocompletion, navigation, and refactoring. These features, especially in Visual Studio Code, improve developer productivity by providing real-time feedback and intelligent suggestions.
5. **Seamless Integration with JavaScript Libraries:** TypeScript is fully compatible with existing JavaScript libraries and frameworks. In the TaskTenders project, TypeScript integrates easily with Node.js and Express, ensuring that the development workflow remains smooth while adding type safety and error prevention.
6. **Scalability:** As TaskTenders grows, TypeScript’s type system ensures that new features can be added without introducing unforeseen bugs. The strong typing and modular nature of TypeScript make it easier to refactor and scale the application as its functionality expands.
7. **Better Refactoring:** TypeScript’s static typing allows for safer and more efficient refactoring of code. Since types are explicitly defined, developers can modify code with confidence, knowing that type-checking will catch any potential issues introduced during refactoring.

**TypeScript disadvantages:**

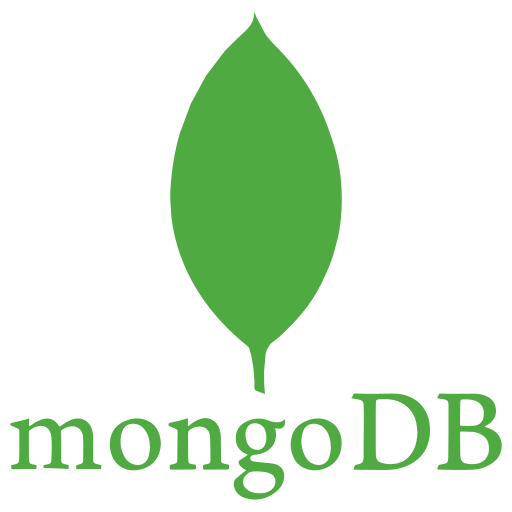
1. **Learning Curve:** Developers familiar with JavaScript may need time to adjust to TypeScript’s strict type system and new features. However, the long-term benefits of using TypeScript often outweigh the initial learning curve.
2. **Longer Development Time:** Writing type definitions and handling TypeScript’s strict rules may slow down development in the short term. However, this can result in fewer bugs and less debugging time later in the process.
3. **Additional Compilation Step:** Unlike JavaScript, TypeScript needs to be compiled into JavaScript before it can be executed by Node.js. This introduces an additional step in the development workflow, though modern tools and build processes can automate this.

**Why TypeScript for TaskTenders?**

TypeScript was chosen for TaskTenders to ensure that the backend code is more maintainable and scalable. By catching errors during the development phase and enforcing a consistent structure across the codebase, TypeScript helps improve code quality and developer productivity. The integration with Node.js and Express ensures that the application benefits from modern JavaScript features, with the added security and reliability of a statically-typed language.

In the TaskTenders backend, TypeScript enhances the development process by providing better code structure, clearer data flow, and a more reliable foundation for building RESTful APIs and handling real-time communication.

### *6.4.3 Database: MongoDB*

MongoDB is a popular, open-source NoSQL database known for its scalability, flexibility, and ability to handle large volumes of data in a non-relational format. Unlike traditional relational databases that use tables and rows, MongoDB stores data in flexible, JSON-like documents, making it ideal for applications that need to handle dynamic and unstructured data.

For **TaskTenders**, MongoDB provides the flexibility needed to manage various types of user-generated data, including job postings, bids, user profiles, and messages between users and freelancers. Its document-based structure allows us to store this information in a way that adapts easily to changes in the app’s data requirements.

**MongoDB advantages:**

1. **Schema Flexibility:** MongoDB’s flexible schema allows for easy updates and changes to data structures without the need for complex migrations, making it ideal for rapidly evolving applications like TaskTenders.
2. **Scalability:** MongoDB is designed for horizontal scalability, allowing it to handle large amounts of data across multiple servers. This ensures the app can grow as the user base expands.
3. **High Performance:** MongoDB’s efficient querying and indexing system enables fast read and write operations, ensuring that TaskTenders delivers quick responses to user actions such as posting jobs, bidding, and messaging.
4. **Document-Based Storage:** MongoDB stores data in JSON-like documents (BSON), which are more intuitive and flexible compared to rigid relational tables. This allows TaskTenders to manage complex data relationships efficiently.
5. **NoSQL Model:** Being a NoSQL database, MongoDB doesn’t require fixed schemas or complex joins, simplifying the management of large datasets and making it easy to store diverse types of information.
6. **Cloud Integration:** MongoDB integrates seamlessly with cloud platforms like AWS, GCP, and MongoDB Atlas, ensuring easy scaling and management in the cloud.
7. **Sharding for Scalability:** MongoDB’s sharding feature helps distribute data across multiple machines, enhancing both storage capacity and performance for large-scale applications.

**MongoDB disadvantages:**

1. **Memory Usage:** MongoDB can require significant memory, as it keeps frequently accessed data in RAM for fast performance.
2. **Limited Support for Complex Transactions:** Although MongoDB introduced multi-document ACID transactions, it still may not be as robust as relational databases when it comes to handling complex transactions.
3. **Less Mature Tooling:** Compared to mature relational databases like MySQL or PostgreSQL, some MongoDB tools, especially for analytics and reporting, are less developed.
4. **Consistency:** MongoDB offers eventual consistency in distributed systems, meaning data may not be immediately updated across all nodes, which could be a challenge in specific use cases.

**Why MongoDB for TaskTenders:**

* **Flexible Data Storage:** The dynamic nature of the data in TaskTenders, such as job postings, user profiles, freelancer bids, and messaging, fits well with MongoDB’s document-oriented approach. This allows the application to evolve without requiring major changes in the database structure.
* **Scalability:** As TaskTenders grows, MongoDB can scale horizontally to handle increased data loads and user traffic, ensuring smooth performance even with a large user base.
* **Fast Prototyping:** MongoDB’s flexible schema design allows rapid prototyping and deployment, making it perfect for the iterative development process required for TaskTenders.

**Integration with Node.js:** MongoDB integrates seamlessly with Node.js through the **Mongoose** library, which provides an elegant, object-oriented interface for interacting with the database. Mongoose simplifies database operations like querying, validation, and data modeling, making it easier to work with MongoDB’s flexible documents in the context of Node.js.

**Real-time Updates:** Using MongoDB’s ability to efficiently handle frequent read and write operations, **TaskTenders** can provide real-time updates for job postings, bid submissions, and user interactions, ensuring a responsive user experience.

### *6.4.4 Front-end: Flutter*

Flutter is an open-source UI software development kit created by Google, enabling developers to build natively compiled applications for mobile (iOS and Android), web, and desktop from a single codebase. Flutter uses the **Dart programming language**, which is designed for fast development and high-performance applications. Dart is optimized for building user interfaces and compiles to native machine code, ensuring efficient performance across platforms.

Flutter provides a rich set of pre-designed widgets that allow for fast and flexible UI development. It uses a reactive framework, like React, but with its own customizable widgets to render the UI, ensuring consistency and fluidity on any platform.

**Flutter advantages:**

1. **Cross-Platform Development:** Write once, deploy on iOS, Android, web, and desktop.
2. **High Performance:** Flutter compiles to native machine code, providing near-native performance.
3. **Rich Widgets Library:** A vast library of pre-built widgets ensures a consistent look across platforms.
4. **Fast Development:** Hot reload allows real-time updates without restarting the app, enhancing productivity.
5. **Customizable UI:** Flutter’s flexibility allows for complex UI designs with smooth transitions and animations.
6. **Single Codebase:** A single codebase for all platforms reduces development time and effort.
7. **Growing Community and Ecosystem:** Flutter’s popularity has fostered a large community with extensive resources and support.

**Flutter disadvantages:**

1. **Large App Size:** Flutter apps tend to have larger file sizes due to the Flutter engine being bundled with the app.
2. **Limited Third-Party Libraries:** While Flutter has a growing ecosystem, it may lack some niche libraries compared to older frameworks.
3. **Limited Native Functionality:** Some advanced native features may require custom platform-specific code.
4. **Learning Curve:** Developers may need time to get familiar with Dart, Flutter's primary programming language.
5. **Platform-Specific UI Elements:** Additional customization may be required to achieve a native look and feel for platform-specific UI elements.

**Flutter's DevTools** offer a suite of performance and debugging tools that enable developers to inspect the widget tree, analyze layout and rendering, debug code, and track performance bottlenecks, making development faster and more efficient.

### *6.4.5 Development Environment: Visual Studio Code*

A blue logo with a cross

Description automatically generatedVisual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft, and it is one of the most widely used development environments for modern web and mobile application development. It provides a lightweight yet powerful environment with support for numerous programming languages, including JavaScript, Dart (for Flutter), and Node.js, making it an ideal choice for the development of **TaskTenders**.

VS Code’s flexibility and extensive ecosystem of extensions make it well-suited for full-stack development, ensuring seamless integration between frontend and backend development tasks.

**Visual Studio Code advantages:**

1. **Lightweight and Fast:** VS Code is lightweight, meaning it can run on almost any system without consuming excessive resources, while still providing robust functionality for large projects like TaskTenders.
2. **Rich Extension Ecosystem:** With a vast marketplace of extensions, developers can enhance VS Code with tools for linting, debugging, Git integration, and language support. Popular extensions like Prettier, ESLint, and Flutter/Dart extensions improve code quality and efficiency.
3. **Built-in Git Integration:** VS Code has built-in support for Git version control, allowing developers to easily manage source code repositories, commit changes, and track project history without leaving the editor.
4. **Debugging Capabilities:** VS Code provides a powerful, built-in debugger for JavaScript and Node.js, and with extensions, it supports Dart for Flutter. This enables developers to debug frontend and backend code within the same environment.
5. **Cross-Platform:** VS Code is available on Windows, macOS, and Linux, making it a flexible choice for developers working in different environments.
6. **Terminal Integration:** The integrated terminal allows developers to run shell commands, npm scripts, and interact with Git directly within VS Code, streamlining the development workflow.
7. **Code Autocompletion:** The IntelliSense feature provides intelligent code suggestions and autocompletion for JavaScript, Dart, and other languages, making coding faster and reducing errors.
8. **Customizable:** VS Code is highly customizable, with themes, keybindings, and configuration options that allow developers to tailor the editor to their workflow.

**Visual Studio Code disadvantages:**

* **Resource Usage:** Although lightweight compared to traditional IDEs, VS Code can become resource-intensive when running multiple extensions and large projects.
* **Learning Curve for Extensions:** While extensions greatly enhance functionality, configuring and managing them can add complexity, especially for new users.
* **Limited Native Features:** Some features available in full Integrated Development Environments (IDEs) might require extensions to be enabled in VS Code, potentially making it less straightforward compared to IDEs designed specifically for a particular language.

**Why Visual Studio Code for TaskTenders:**

* **Frontend and Backend Development:** VS Code’s excellent support for both **Node.js** (backend) and **Flutter/Dart** (frontend) makes it an ideal choice for developing TaskTenders. Developers can switch seamlessly between backend services in Node.js and the frontend interface in Flutter using a single editor.
* **Git Integration:** With the built-in Git integration, the development team can easily collaborate on the project, manage source code changes, and keep track of version history for TaskTenders’ codebase.
* **Real-time Collaboration:** Using extensions like **Live Share**, VS Code allows developers to collaborate in real-time, making pair programming and team collaboration easier during the development process.
* **Customization for TaskTenders:** By configuring custom tasks, snippets, and key bindings, the development team can streamline repetitive workflows and automate common tasks related to building, testing, and deploying the app.

**Extensions for TaskTenders Development:**

* **Flutter & Dart:** For front-end development, the **Flutter** and **Dart** extensions provide support for building and debugging the mobile app.
* **Node.js Extension Pack:** Includes tools like **npm**, **Express Snippets**, and **REST Client** for managing the backend API and services.
* **Prettier and ESLint:** Code formatting and linting tools to ensure code quality and consistency across the team.
* **MongoDB Extension:** Enables developers to interact with the MongoDB database directly from VS Code, making it easier to test queries and visualize data.

## *6.5 Project Structure*

The **TaskTenders** project is organized to facilitate efficient development, scalability, and maintainability. It follows a modular structure to separate concerns between the frontend, backend, and shared resources, ensuring that each part of the system is independently manageable and easy to collaborate on. This organization also allows for better testing, debugging, and future expansions.

**1. Frontend (Flutter)**

The frontend of TaskTenders is built using **Flutter**, a framework for building cross-platform mobile applications. The Flutter project structure is organized to separate UI components, business logic, and services, making it easier to maintain and scale.

* **lib/**: Main directory for all Flutter app code.
  + **screens/**: Contains all the screen widgets (e.g., login screen, job posting screen, freelancer bidding screen).
  + **widgets/**: Reusable UI components, such as buttons, cards, or form fields.
  + **models/**: Data models that define the structure of objects such as user profiles, job postings, and bids.
  + **providers/**: Handles the state management and business logic using **Provider** or **Riverpod** for managing state across different screens.
  + **services/**: Includes code for interacting with backend APIs, authentication, and other external services.
  + **utils/**: Utility functions such as formatters, validators, or constants used throughout the app.
  + **assets/**: Contains images, fonts, and other static resources used in the app.

**2. Backend (Node.js/Express.js)**

The backend is responsible for handling server-side logic, managing data storage, processing requests, and providing APIs for the frontend. It is built using **Node.js** with the **Express.js** framework.

* **src/**: Main directory for backend code.
  + **controllers/**: Contains logic for handling incoming API requests (e.g., user registration, job posting, bidding process).
  + **routes/**: Defines the routes and endpoints for the APIs (e.g., /api/users, /api/jobs).
  + **models/**: Defines data schemas and models using **Mongoose** to interact with the MongoDB database.
  + **middleware/**: Contains middleware functions for authentication, authorization, and validation.
  + **services/**: Handles services like sending emails, processing payments, and interacting with third-party APIs.
  + **config/**: Contains configuration files (e.g., environment variables, database connection strings).
  + **utils/**: Utility functions, such as for data validation, formatting responses, or logging.
  + **tests/**: Unit and integration tests for the backend logic to ensure code reliability.

**3. Database (MongoDB)**

MongoDB is the NoSQL database used to store and manage dynamic data for TaskTenders. The database is organized with a focus on scalability and flexibility, and is accessed through the backend using **Mongoose**.

* **Collections**:
  + **Users**: Stores user profiles for freelancers and clients, including credentials, preferences, and roles.
  + **Jobs**: Contains job postings, including descriptions, budget, deadlines, and job status.
  + **Bids**: Manages freelancer bids on job postings, including bid amounts, timelines, and freelancer profiles.
  + **Transactions**: Tracks payments and transaction histories between users and freelancers.
  + **Messages**: Stores conversation data between users and freelancers for in-app communication.

**4. Shared Resources**

This section covers assets and shared utilities that are used across the entire project.

* **docs/**: Project documentation, including requirements, API specs, and architectural diagrams.
* **logs/**: Log files generated by the backend for monitoring application performance and debugging.
* **tests/**: Contains test cases for both frontend and backend code to ensure the reliability of features.

**5. Version Control**

TaskTenders uses **Git** for version control, with the source code hosted on **GitHub**. The project structure supports branching strategies such as feature branches, bugfixes, and hotfixes to streamline team collaboration and ensure code stability.

* **master/main branch**: Contains the stable code that is ready for production.
* **development branch**: Used for integrating new features and performing testing before merging into the main branch.
* **feature branches**: Each new feature or functionality is developed in its own branch, allowing developers to work in isolation without affecting the main project.
* **bugfix branches**: For resolving issues and bugs reported in the system.

**6. Testing**

Both the frontend and backend include tests to ensure the reliability and stability of the application.

* **Frontend Tests**: Uses Flutter’s testing framework for unit tests and widget tests, ensuring that UI components and business logic function correctly.
* **Backend Tests**: Uses **Mocha/Chai** for unit and integration tests on the Node.js backend, testing API endpoints, data processing, and user authentication.

## *6.6 Algorithms*

**1. Matching Algorithm**

The matching algorithm is fundamental to the TaskTenders platform, ensuring that user's job postings are accurately matched with freelancers who have the requisite skills and experience. This algorithm analyzes various factors such as:

* **Skill Relevance**: Matching based on the skills and qualifications listed by freelancers against the requirements of the job posting.
* **Geographical Location**: Preferentially matching freelancers who are in close proximity to the job location, if relevant.
* **Availability**: Aligning freelancer availability with the job timeline.
* **Rating and Performance History**: Prioritizing freelancers with higher ratings and a history of successful job completions.

This algorithm uses a weighted scoring system where each factor is assigned a score based on its importance. The scores are then aggregated to rank freelancers for each job posting.

**2. Bidding Algorithm**

The bidding algorithm facilitates fair and competitive bidding by allowing freelancers to place bids on job postings. It includes features such as:

* **Dynamic Pricing Suggestions**: Based on historical data, the algorithm suggests a competitive bid amount that freelancers might consider.
* **Time Decay Factor**: Bids might be adjusted based on the urgency of the job posting, with more immediate needs potentially driving higher bid values.
* **Bid Ranking**: Sorting bids not only by price but also considering factors like freelancer rating and the number of previously completed jobs.

**3. Rating System**

The rating system algorithm is designed to ensure fairness and accuracy in how freelancers' performance is assessed. It considers:

* **Client Feedback**: Ratings provided by clients after a job is completed.
* **Complexity of Jobs**: Weighing ratings according to the complexity of the tasks completed.
* **Timeliness and Communication**: Factors such as adherence to deadlines and communication quality affect the final rating.

**4. Dynamic Job Recommendation Engine**

For users, a recommendation engine suggests potential jobs based on their previous search behaviors, skills, and ratings. This uses machine learning techniques to learn user preferences over time and improve the relevance of job recommendations.

**5. Fraud Detection Algorithm**

To maintain the integrity of the platform, a fraud detection algorithm continuously analyzes user activities to flag potentially fraudulent or abnormal behaviors such as:

* **Multiple Accounts**: Identifying users who operate multiple freelancer or client accounts to manipulate bidding or ratings.
* **Payment Anomalies**: Detecting unusual patterns in payment transactions that could indicate fraudulent activity.

**Implementation Considerations**

These algorithms are implemented using a combination of traditional programming techniques and modern machine learning methods, depending on the complexity and requirements of each task. The backend, developed with Node.js and TypeScript, integrates these algorithms to ensure that data handling is efficient and secure. The algorithms are regularly updated based on feedback and new data to improve accuracy and performance.

**Conclusion**

The algorithms section outlines the critical computational processes that support the TaskTenders platform. By elaborating on these key algorithms, stakeholders can understand how the platform operates dynamically to match freelancers with jobs, facilitate competitive bidding, ensure fair ratings, recommend relevant jobs, and prevent fraud.

## *6.7 Diagrams*

### *6.7.1 use case diagram*

### *6.7.2 Activity diagram*

## *6.8 Tests*

This section outlines the test cases designed to validate the functionality, security, and performance of the TaskTenders platform, segmented by different user roles and system administration.

### *6.8.1 Login Tests*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Method | Expected Outcome | Status |
| L1 | Correct credentials login | Automated Test | User is logged in successfully | Pending |
| L2 | Incorrect password | Automated Test | Login fails with an error message | Completed |
| L3 | Password reset process | Manual Testing | User can reset password and log in | Pending |
| L4 | Login session expiration | Automated Test | User session expires as expected | Completed |
| L5 | Login with two-factor authentication | Automated Test | User is prompted for secondary authentication method | Pending |
| L6 | Login via social media accounts | Integration Test | User can log in using external social media accounts | Pending |
| L7 | Account lockout after multiple failed attempts | System Test | Account is temporarily locked after consecutive failed login attempts | Pending |
| L8 | Captcha verification on login | Manual Testing | Captcha is required and works correctly after failed login attempts | Completed |
| L9 | Handling of session tokens during login | Security Test | Session tokens are handled securely during and after login | Pending |
| L10 | Response to SQL injection attempts during login | Security Test | System securely rejects SQL injection in login fields | Completed |

**Purpose**: To ensure that the login mechanism is secure, user-friendly, and robust against unauthorized access attempts.

### *6.8.2 User Tests*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Method | Expected Outcome | Status |
| U1 | Update profile information | Integration Test | Profile is updated with new information | Pending |
| U2 | Delete user account | Manual Testing | User account is deleted correctly | Completed |
| U3 | View job history | Automated Test | User can view past jobs accurately | Pending |
| U4 | Account registration verification | Automated Test | User receives verification email | Pending |
| U5 | Change password functionality | Manual Testing | Password change is processed correctly | Completed |
| U6 | Mobile app login with biometrics | System Test | User can log in using biometric data | Pending |
| U7 | Handling of user session timeouts | Automated Test | User is logged out after inactivity | Completed |
| U8 | Search and filter jobs | Integration Test | Search returns accurate results | Pending |

**Purpose**: These tests ensure that all user-related functionalities are robust, secure, and user-friendly, enhancing the overall experience of the clients and freelancers on the platform.

### *6.8.3 Freelancer Tests*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Method | Expected Outcome | Status |
| F1 | Bid on job posting | Automated Test | Freelancer can submit bids successfully | Pending |
| F2 | Withdraw from bid | Manual Testing | Freelancer can withdraw bid correctly | Completed |
| F3 | Complete job and receive payment | Integration Test | Job completion triggers payment correctly | Pending |
| F4 | Manage portfolio | Manual Testing | Freelancer can update their portfolio | Completed |
| F5 | Receive and respond to client queries | Integration Test | Communication with clients is seamless | Pending |
| F6 | Notification settings adjustment | System Test | Freelancer can customize notifications | Completed |
| F7 | Review and rate clients | Automated Test | Freelancer can submit reviews and ratings | Pending |

**Purpose**: To test and verify that all freelancer-specific functionalities are efficient, reliable, and facilitate their engagement with job postings and clients seamlessly.

### *6.8.4 Admin Tests*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Method | Expected Outcome | Status |
| A1 | Add new job categories | Manual Testing | Admin can add new categories successfully | Pending |
| A2 | Ban a user | Automated Test | User is banned and cannot access system | Completed |
| A3 | Generate reports | Integration Test | Admin can generate accurate reports | Pending |
| A4 | Manage user complaints | Manual Testing | Complaints are resolved efficiently | Completed |
| A5 | Update platform policies | System Test | Policy updates are deployed correctly | Pending |
| A6 | Monitor user activity | Automated Test | Admin can monitor and review user activity | Completed |
| A7 | Backup and data recovery procedures | Integration Test | Data recovery processes function properly | Pending |

**Purpose**: To ensure that administrative functionalities are robust, secure, and allow for efficient management and oversight of the TaskTenders platform.

# **7. GUI**

## *7.1 User Screens*

## *7.2 Freelancer Screens*

## *7.3 Admin Screens*

# **8. References**

1. Microsoft. (n.d.). *Visual Studio Code - Code Editing. Redefined.* Retrieved from <https://code.visualstudio.com/>
2. Microsoft. (2021). *Using Git in Visual Studio Code.* Retrieved from <https://code.visualstudio.com/docs/editor/versioncontrol>
3. Fekete, A. (2020). *Mastering Visual Studio Code.* Packt Publishing.
4. Node.js — Introduction to Node.js. (n.d.). <https://nodejs.dev/en/learn/>
5. GeeksforGeeks. (2023, December 1). MongoDB Advantages Disadvantages. GeeksforGeeks. <https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/>
6. Google. (n.d.). Flutter - Beautiful native apps in record time. Retrieved from <https://flutter.dev/>
7. Google. (n.d.). Dart Language Overview. Retrieved from <https://dart.dev/guides>
8. Marco, E. (2019). Beginning Flutter: A Hands-On Guide to App Development. Apress.