Fpt University

TRƯỜNG ĐẠI HỌC FPT

Capstone Project Report

Report 1 – Project Introduction

– Can Tho, December 2022025–

Table of Contents

I. Project Report

1. Status Report

#	Work Item	Status	Notes (Work Item in Details)
1		Pending	
2		In Progress	
3		Completed	

2. Team Involvements

#	Task	Member	Notes (Task Details, etc.)
1		KienNT	
2		TuanTV	
3		AnhLM	

3. Issues/Suggestions

#	Issue	Status	Notes (Solution, Suggestion, etc.)
1		Pending	
2		In Progress	
3		Completed	

II. Project Introduction

1. Overview

1.1 Project Information

- Project name: InterDev - Freelance Software Project Brokerage Platform for SMEs in Vietnam
- Project code: InterDev
- Group name: SEP490_06
- Software type: Web Application

1.2 Project Team

a. Supervisor

Full Name	Email	Phone Number	Title
Luong Hoang Huong	Huonglh3@fe.edu.vn	0941666545	Supervisor

b. Team Members

Full Name	Email	Mobile	Role
Ngo Thai Son	SonNTCE180624@fpt.edu.vn	0939124198	Leader
Dang Chi Bang	bangdcce181999@fpt.edu.vn	0964682850	Member
Nguyen Phuc An	annpce181888@fpt.edu.vn	0383718494	Member
Nguyen Gia BaBao	baongce18		Member
Pham Tri Trong An	anpttce181254@fpt.edu.vn	0944613448	Member

2. Product Background

In the context of digital transformation in Vietnam, many Small and Medium Enterprises (SMEs), shop owners, and local businesses have a high demand for building websites, mobile apps, or management systems to digitize their operations.

However, these non-technical clients face significant challenges:

- They lack technical knowledge and cannot write professional software requirements (specifications).
- They struggle to communicate effectively with developers, leading to misunderstandings, "feature creep," and products that do not match their vision.
- They fear being scammed or overcharged when working with anonymous freelancers.

On the other side, IT Freelancers face issues such as vague requirements, scope changes without extra pay, and payment risks.

Current freelance platforms only focus on connecting two parties via messaging but lack a quality control layer. Therefore, there is a strong need for a platform with a **"Broker"** (acting as a Business Analyst/Project Manager). This Broker will serve as an intermediary to standardize requirements,

2.1 React

React is a declarative, efficient, and flexible open-source JavaScript library for building user interfaces, maintained by Meta (formerly Facebook). It lets developers compose complex UIs from small and isolated pieces of code called "components." React creates a Virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM. This process allows for high performance and a smoother user experience, especially for single-page applications (SPAs) where data changes over time.

2.1.1. Characteristics

- **Component-Based:** Builds encapsulated components that manage their own state, then composes them to make complex UIs.
- **Declarative:** Makes it painless to create interactive UIs by designing simple views for each state in the application.
- **Virtual DOM:** Uses an in-memory data structure cache to compute the resulting differences, and then updates the browser's displayed DOM efficiently.
- **JSX Syntax:** Allows HTML quoting and tag syntax to render subcomponents within JavaScript.
- **Unidirectional Data Flow:** Implements one-way data binding which ensures that changes in child components do not affect parent data.

2.1.2. Advantages

- **High Performance:** The Virtual DOM minimizes direct access to the standard DOM, speeding up rendering.
- **Reusability:** Components can be reused across different parts of the application, reducing code duplication.
- **Strong Community Support:** Massive ecosystem of libraries, tools, and community-driven solutions.
- **SEO Friendly:** React can run on the server and render the virtual DOM to the browser as a regular web page.

- **Easy to Learn:** For developers already familiar with JavaScript, React's syntax and concepts are straightforward to pick up.

2.1.3. Disadvantages

- **High Pace of Development:** The environment changes so fast that some developers feel it is hard to keep up with all new features and best practices.
- **Poor Documentation:** With so many tools and libraries updates, documentation can sometimes be outdated or incomplete.
- **View Layer Only:** React covers only the UI layers of the app and nothing else; developers still need to choose other technologies for HTTP requests, state management, etc.
- **JSX Complexity:** Some developers may find JSX (JavaScript XML) confusing initially as it mixes HTML with JavaScript logic.

Figure 1. React Library[1]

2.2 NestJS

NestJS is a progressive Node.js framework for building efficient, reliable, and scalable server-side applications. It is built with and fully supports TypeScript (yet still enables developers to code in pure JavaScript) and combines elements of OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming). Under the hood, NestJS makes use of robust HTTP Server frameworks like Express (the default) and optionally Fastify.

2.2.1. Characteristics

- **Modular Architecture:** Encourages organizing code into modules, making the application structure organized and easy to maintain.
- **TypeScript Support:** Built from the ground up with TypeScript, providing strong typing and modern JavaScript features.
- **Dependency Injection:** Has a powerful built-in Dependency Injection (DI) container to manage class dependencies.

[1] https://react.dev/

- **Extensible:** Easy to use with other libraries, allowing for flexible integration of third-party modules.
- **Decorator-based:** Uses decorators heavily (like Java Spring Boot) to define modules, controllers, and services.

2.2.2. Advantages

- **Scalability:** The modular architecture allows the application to scale easily as complexity grows.
- **Maintainability:** Strong typing and clear structure make the code easier to read, test, and maintain.
- **Familiarity:** Developers coming from Angular or Spring Boot will find NestJS's architecture very familiar.
- **Rich Ecosystem:** Provides out-of-the-box solutions for microservices, WebSockets, GraphQL, and more.
- **Testing:** Dependency injection system makes unit testing and integration testing straightforward.

2.2.3. Disadvantages

- **Learning Curve:** The heavy use of decorators and architectural patterns can be overwhelming for developers used to simple Node.js/Express apps.
- **Verbosity:** Can require more boilerplate code compared to minimalist frameworks like Express.
- **Performance Overhead:** The abstraction layers may introduce a slight performance overhead compared to raw Node.js/Fastify, though usually negligible for most apps.
- **Debugging:** Sometimes debugging TypeScript compilation issues or complex dependency injection errors can be tricky.

Figure 2. NestJS[2]

2.3 PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system (ORDBMS) with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. It extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

2.3.1. Characteristics

- **ACID Compliance:** Fully ACID (Atomicity, Consistency, Isolation, Durability) compliant, ensuring valid database transactions.
- **Extensibility:** Users can define their own data types, index types, and functional languages.
- **Advanced Data Types:** Supports JSON/JSONB, XML, Hstore, and geometric data types natively.
- **Concurrency Control:** Uses Multi-Version Concurrency Control (MVCC) to manage concurrent access to data efficiently.
- **Standard Compliance:** Highly compliant with the SQL standard.

2.3.2. Advantages

- **Reliability and Stability:** Known for being rock-solid and preventing data corruption.
- **Open Source:** Free to use, with no licensing costs and a vibrant community.
- **Complex Queries:** Excellent support for complex queries, joins, and foreign keys.
- **Scalability:** Capable of handling large volumes of data and high concurrent user loads.
- **Geospatial Support:** With the PostGIS extension, it is the industry standard for geospatial data.

2.3.3. Disadvantages

- **Performance:** For simple read-heavy operations, it might be slower than some NoSQL databases or lighter SQL alternatives like MySQL (in certain scenarios).
- **Memory Usage:** Can be memory-intensive as every new client connection forks a new process.
- **Complexity:** The vast feature set can make it complex to configure and optimize for beginners.
- **Data Migration:** Migrating from other database systems to PostgreSQL can be challenging due to strict standard compliance.

[2] https://nestjs.com/

Figure 3. Postgre SQL[3]

2.4 Docker

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization. It allows developers to package an application with all of its dependencies (libraries, runtime, system tools, etc.) into a standardized unit called a container. This ensures that the application runs consistently regardless of the environment, solving the "it works on my machine" problem.

2.4.1. Characteristics

- **Containerization:** Wraps software in a complete file system that contains everything it needs to run.
- **Isolation:** Containers are isolated from one another and bundle their own software, libraries, and configuration files.
- **Portability:** Docker containers can run on any computer, on any infrastructure, and in any cloud.
- **Lightweight:** Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies.
- **Version Control:** Docker supports versioning of container images (Docker Hub).

2.4.2. Advantages

- **Consistency:** Ensures the application runs the same way in development, testing, and production.
- **Efficiency:** Uses fewer resources than traditional virtual machines (VMs) because containers share the OS kernel.
- **Rapid Deployment:** Containers can be spun up or down in seconds, facilitating rapid scaling and deployment.

[3] https://medium.com/codex/intro-to-postgresql-c8da31335c34

- **Microservices Support:** Ideal for microservices architecture where each service runs in its own container.
- **DevOps Integration:** Plays a crucial role in modern CI/CD pipelines.

2.4.3. Disadvantages

- **Complexity:** Can add a layer of complexity to the infrastructure and requires learning new tools and commands.
- **Data Storage:** Managing persistent data in containers (volumes) can be tricky for beginners.
- **Security:** If not configured correctly, containers can have security vulnerabilities, especially if running with root privileges.
- **Performance:** While faster than VMs, there is still a slight overhead compared to running natively on the host OS.

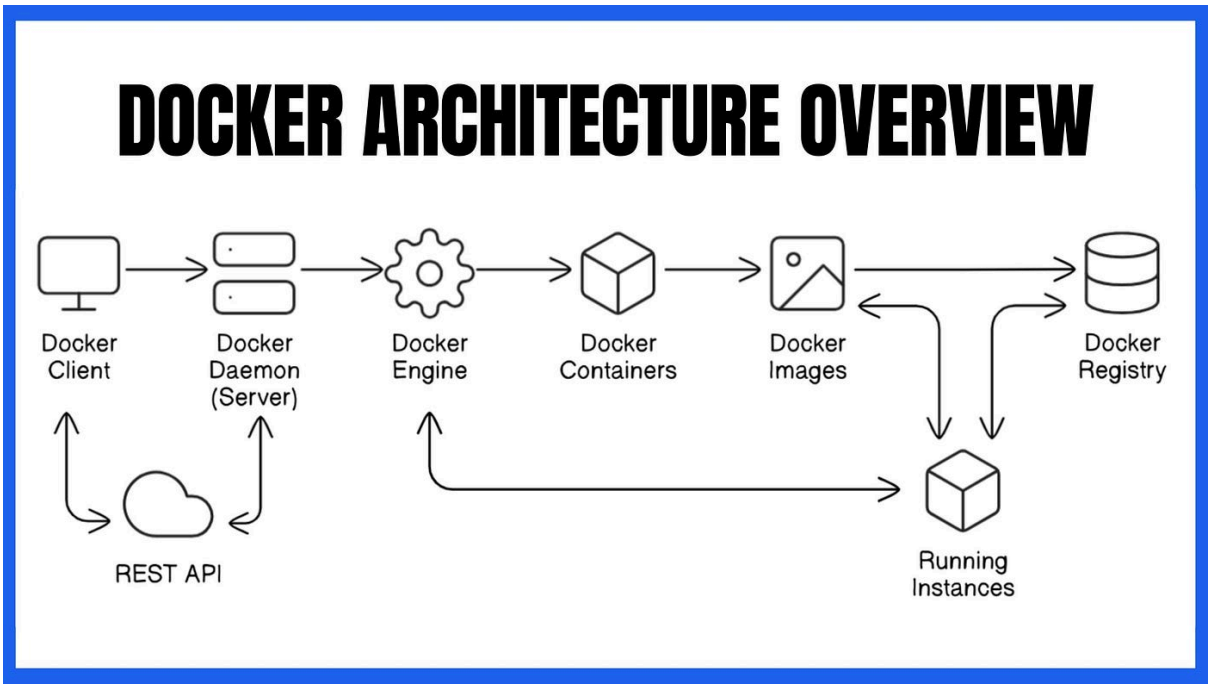

Figure 4. Docker Architecture[4]

2.5 Firebase Cloud Storage

Firebase Cloud Storage is a powerful, simple, and cost-effective object storage service built for Google scale. It is designed to store and serve user-generated content, such as photos, videos, and contractual documents. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for Firebase apps, regardless of network quality.

2.5.1. Characteristics

[4] https://medium.com/m/global-identity-2?redirectUrl=https%3A%2F%2Ffaun.pub%2Fthe-ultimate-guide-to-docker-from-novice-to-expert-cf961536c147

- **Robust Security:** Integrates with Firebase Authentication to provide simple and intuitive security for files.
- **High Scalability:** Built on Google Cloud Storage infrastructure, capable of handling exabytes of data.
- **Resumable Uploads:** Automatically pauses and resumes transfers if the app loses network connection, saving users time and bandwidth.
- **Global Edge Caching:** Files are stored in Google's global network, ensuring fast delivery.
- **SDK Integration:** Provides client-side SDKs for easy integration with web and mobile apps.

2.5.2. Advantages

- **Ease of Use:** Extremely simple API to upload and retrieve files from client-side code.
- **Cost-Effective:** Generous free tier and pay-as-you-go pricing for scaling.
- **Reliability:** Backed by Google's SLA and infrastructure.
- **Serverless:** No need to manage your own storage servers or file systems.
- **Seamless Integration:** Works perfectly with other Firebase services like Firestore and Auth.

2.5.3. Disadvantages

- **Vendor Lock-in:** Migrating away from Firebase to another provider can be difficult.
- **Limited Querying:** Unlike a database, you cannot complexly query the metadata of stored files directly; usually requires a companion database.
- **Cost at Scale:** While the free tier is good, costs can increase significantly with very high bandwidth or storage usage.
- **Cold Starts:** In some serverless contexts, there might be slight latency in initial connections.

Figure 5. Firebase[5]

2.6 Redis

Redis (Remote Dictionary Server) is an open-source, in-memory data structure store, used as a database, cache, and message broker. It supports various data structures such as strings, hashes, lists, sets, and more. Because it resides in memory, Redis delivers sub-millisecond response times, making it ideal for real-time applications.

Figure 9. Redis data structures diagram

2.6.1. Characteristics

- **In-Memory Storage:** Keeps all data in RAM for extremely fast read and write access.
- **Key-Value Store:** Data is stored as key-value pairs, but supports complex data types.
- **Persistence:** Can optionally persist data to disk (RDB or AOF) to prevent data loss.
- **Replication:** Supports master-slave replication for high availability.
- **Atomic Operations:** Operations on data structures are atomic, ensuring data integrity.

2.6.2. Advantages

- **Exceptional Performance:** Extremely fast, capable of handling millions of requests per second.
- **Flexible Data Structures:** Not limited to simple strings; supports lists, sets, maps, etc.
- **Simple Usage:** Easy to install, configure, and use.
- **Versatility:** Can be used for caching, session management, real-time analytics, and pub/sub messaging.
- **Community:** Large community and support for almost every programming language.

2.6.3. Disadvantages

- **Memory Limitation:** Data size is limited by the available RAM on the server.
- **Persistence Overhead:** Configuring persistence correctly to balance performance and data safety can be complex.
- **No SQL Support:** It is a NoSQL store, so it lacks the complex querying capabilities of relational databases.
- **Data Consistency:** In clustered setups, ensuring strong consistency can be challenging (CAP theorem trade-offs).

[5] https://firebase.google.com/docs/guides?hl=vi

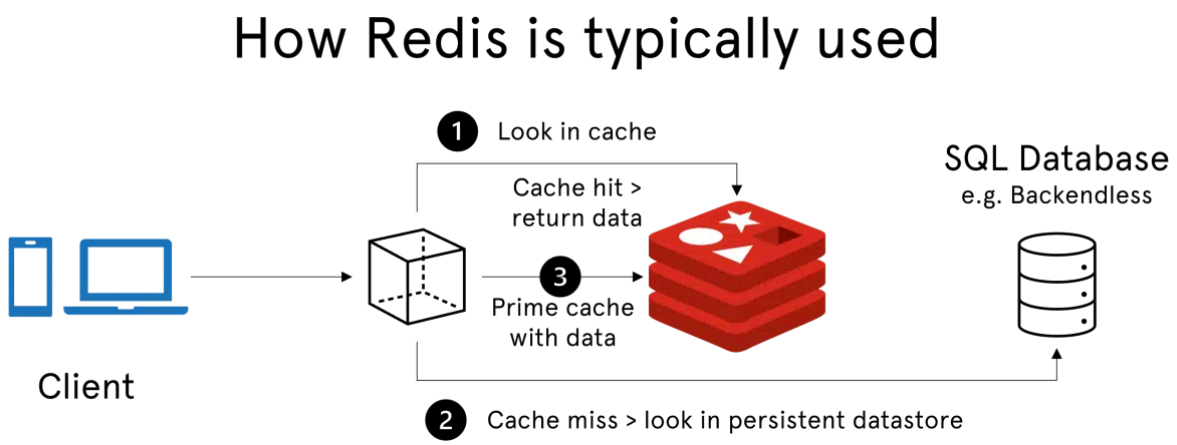

Figure 6. Regis Architect[6]

2.7 GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows developers to automate their build, test, and deployment pipeline. It enables the creation of workflows that build and test every pull request to a repository, or deploy merged pull requests to production.

2.7.1. Characteristics

- **Workflow Automation:** Define custom workflows using YAML files directly in the repository.
- **Event-Driven:** Workflows can be triggered by GitHub events (push, pull request, release) or on a schedule.
- **Matrix Builds:** Run tests across multiple operating systems and runtime versions simultaneously.
- **Live Logs:** View real-time logs of the workflow execution.
- **Marketplace:** Access to thousands of pre-built actions created by the community.

2.7.2. Advantages

- **Integrated with GitHub:** No need for external CI/CD tools; everything is in one place.
- **Free Tier:** Generous free minutes for public repositories and a decent amount for private ones.
- **Customizable:** Highly flexible; can run practically any command or script.
- **Community Actions:** Easy to drop in pre-made actions for standard tasks (e.g., "Checkout", "Setup Node", "Deploy to Docker").
- **Container Support:** Native support for Docker container-based jobs.

[6] https://backendless.com/redis-what-it-is-what-it-does-and-why-you-should-care/

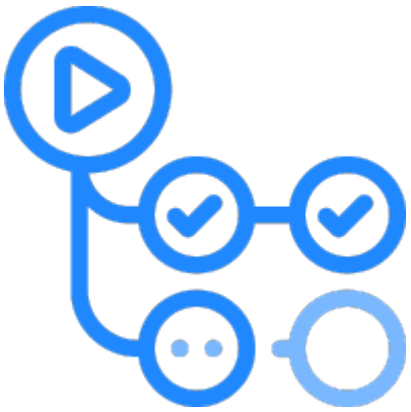

Figure 7. Github actions[7]

2.7.3. Disadvantages

- **Vendor Lock-in:** Workflows are specific to GitHub and not easily portable to GitLab CI or Jenkins.
- **Debugging:** Debugging failed workflows can sometimes be tedious compared to local scripts.
- **Complex Syntax:** YAML syntax for complex logic (conditionals, loops) can become messy.
- **Resource Limits:** Execution time and concurrency limits apply, especially on the free plan.

2.8 Tailwind CSS

Tailwind CSS is a utility-first CSS framework for rapidly building custom user interfaces. Unlike component-based frameworks like Bootstrap, Tailwind does not provide pre-styled components (like buttons or navbars). Instead, it provides low-level utility classes that let developers build completely custom designs without ever leaving the HTML.

2.8.1. Characteristics

- **Utility-First:** Detailed classes for almost every CSS property.
- **Responsive Design:** Extensive prefix system for easy responsive layouts.
- **Configuration:** Highly customizable via tailwind.config.js.
- **JIT (Just-In-Time) Compiler:** Generates styles on demand as you author your templates, resulting in lightning-fast build times.
- **PurgeCSS Integration:** Automatically removes unused CSS in production builds.

2.8.2. Advantages

- **Rapid Development:** Speeds up the styling process by reducing the need to switch between HTML and CSS files.
- **Small Bundle Size:** With unused styles purged, the final CSS file is incredibly small.

---

[7] https://github.com/actions

- **Consistency:** Utility classes ensure spacing, colors, and typography are consistent throughout the app.
- **No Naming Struggles:** Eliminates the need to invent class names.
- **Mobile-First:** Encourages mobile-first design patterns naturally.

2.8.3. Disadvantages

- **HTML Clutter:** Class strings in HTML can become very long and clutter the code readability.
- **Learning Curve:** Requires learning the specific utility class names for standard CSS properties.
- **Lack of Components:** Developers must build components (buttons, inputs) from scratch or use a UI library like Headless UI.
- **Consistency Enforcement:** Without strict guidelines, different developers might solve the same styling problem with different utilities.

Figure 8. Tailwind CSS[8]

2.9 Google reCAPTCHA

Google reCAPTCHA is a security service that protects your website from fraud and abuse. It uses an advanced risk analysis engine and adaptive challenges to keep automated software (bots) from engaging in abusive activities on your site, while letting valid users pass with ease.

2.9.1. Characteristics

- **Bot Detection:** Distinguishes between human users and automated bots.
- **Adaptive Challenges:** Presents challenges (like identifying images) only when necessary (v2) or runs invisibly (v3).

[8] https://github.com/tailwindlabs/tailwindcss-intellisense

- **Risk Analysis:** Assigns a score to user interactions to determine the likelihood of them being a bot (v3).
- **Security:** Protects login, registration, and contact forms from spam and brute-force attacks.

2.9.2. Advantages

- **Enhanced Security:** Effectively blocks most automated spam and abuse.
- **User Experience:** "Invisible" and "Checkbox" versions are much less intrusive than old text-warping captchas.
- **Easy Integration:** Simple API and libraries available for most frameworks (including React).
- **Free Service:** Free for low to moderate volume usage.

2.9.3. Disadvantages

- **Privacy Concerns:** Involves sending user data to Google for analysis.
- **User Friction:** Even simple challenges can annoy some users or be difficult for users with disabilities (though accessibility options exist).
- **False Positives:** Occasionally may flag a legitimate user as a bot.
- **Dependency:** Creates a dependency on an external Google service.

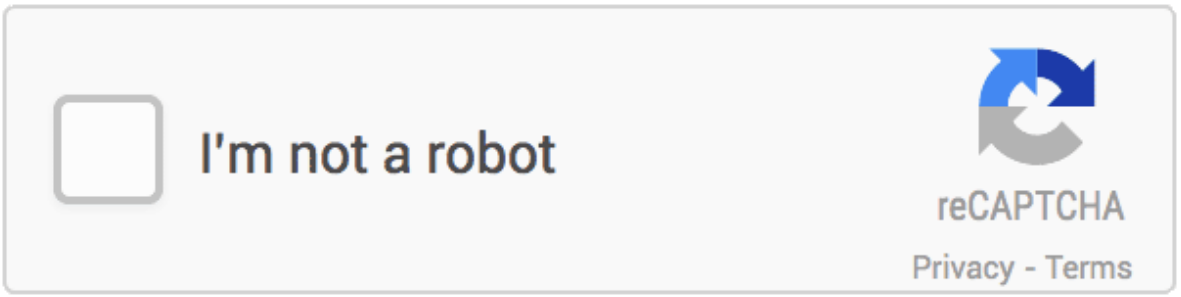

Figure 9. Google reCAPTCHA[9]

2.10 HTML (HyperText Markup Language)

HTML is the standard markup language for documents designed to be displayed in a web browser. It defines the structure and meaning of web content using "tags" to wrap different parts of the content to make it appear or act a certain way.

2.10.1. Characteristics

- **Structure:** Defines the skeleton of web pages (headings, paragraphs, lists, links).
- **Semantics:** Provides semantic meaning to content improving accessibility and SEO.
- **Hypertext:** Enables linking between pages and resources via hyperlinks.
- **Standardized:** Maintained by the W3C and WHATWG.

2.10.2. Advantages

[9] https://developers.google.com/recaptcha

- **Universal:** The fundamental building block of the web, supported by all browsers.
- **Easy to Learn:** Simple syntax and structure.
- **Accessibility:** Semantic HTML helps screen readers interpret content correctly.
- **Integration:** Seamless integration with CSS and JavaScript.

2.10.3. Disadvantages

- **Static:** HTML alone cannot create dynamic functionality (needs JavaScript).
- **Styling Limitations:** Plain HTML looks basic; requires CSS for visual appeal.
- **Browser Inconsistencies:** Older browsers may render some newer HTML5 tags differently (though rare nowadays).

Figure 10. HTML5[10]

2.11 System Architecture

2.11.1 Client-Server Architecture (Overall) The overall system follows the Client-Server architecture model. This is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

- **Client (Frontend):** The ReactJS application running in the user's browser. It is responsible for the user interface, presentation logic, and sending requests to the server.

[10] https://www.oxfordwebstudio.com/en/did-you-know/what-is-html

- **Server (Backend):** The NestJS application running on a server (or container). It processes business logic, interacts with the database (PostgreSQL/Supabase) and external services (Firebase, Redis), and sends responses back to the client.
- **Communication:** The client and server communicate over the HTTP/HTTPS protocol using RESTful APIs.

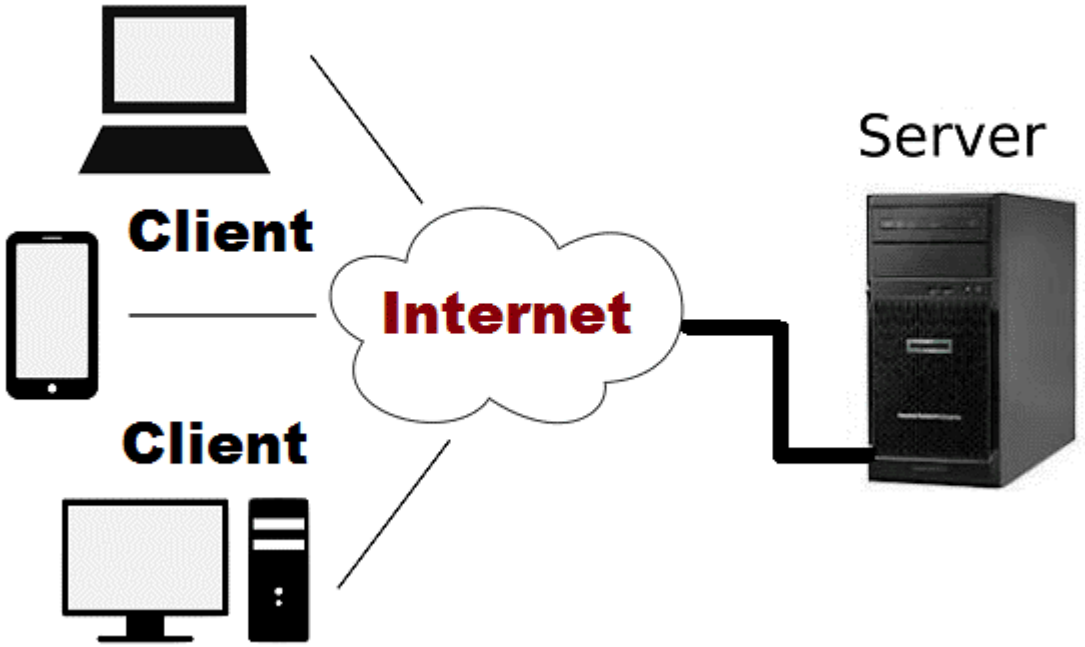

Figure 11. Client Server Architech[11]

2.11.2 Layered Architecture (Backend) The backend (NestJS) is designed using a Layered Architecture. This pattern organizes the code into horizontal layers, each with a specific responsibility. This separation of concerns improves maintainability and testability.

- **Controller Layer:** Handles incoming HTTP requests, validates input, and sends responses. It acts as the entry point for the application.
- **Service Layer (Business Logic):** Contains the core business rules and logic. It processes data received from the controllers and interacts with the Data Access Layer.
- **Data Access Layer (Repository):** Responsible for communicating with the database (PostgreSQL). It performs CRUD operations and abstracts the database implementation details from the rest of the application.
- **DTO (Data Transfer Object):** Defines the shape of data being sent over the network, ensuring strict typing and validation.

[11] https://teachcomputerscience.com/client-server-architecture/

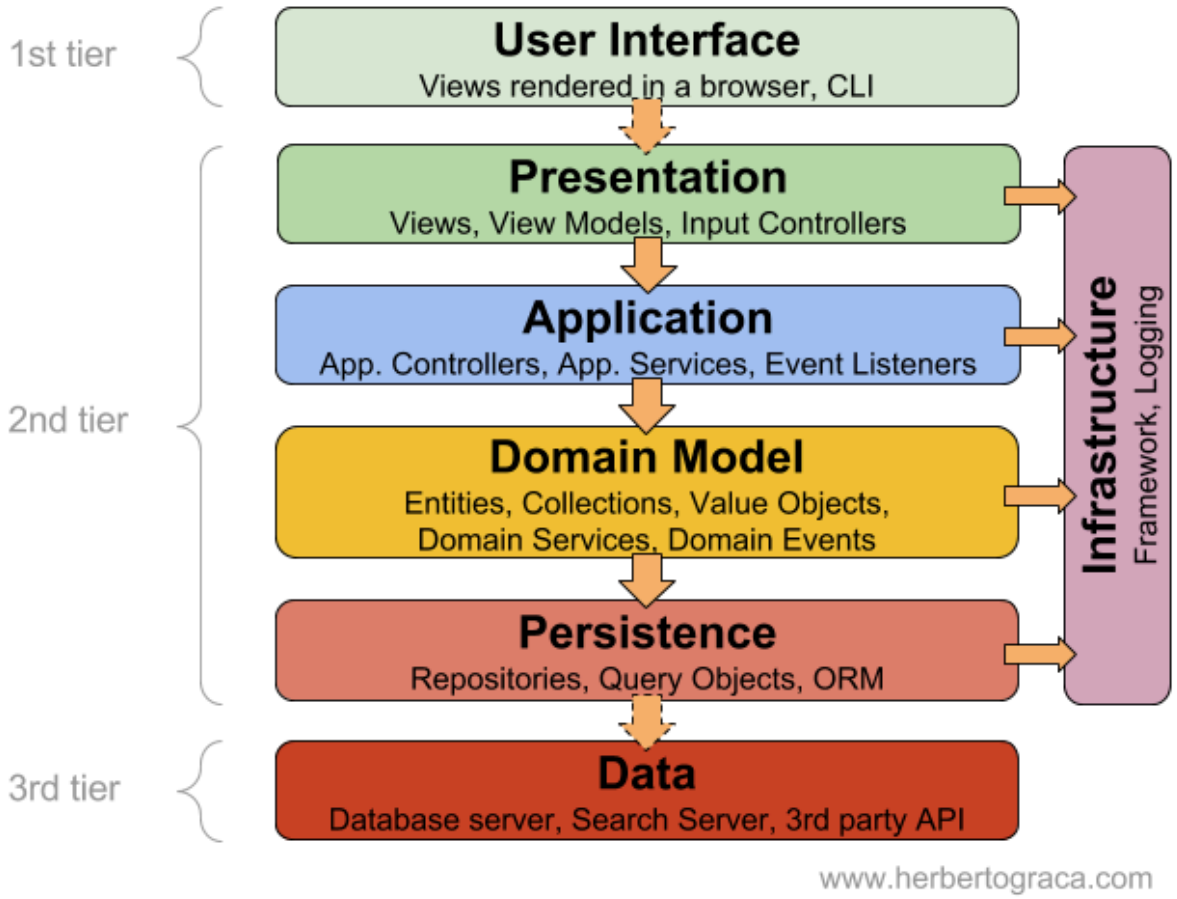

Figure 12. Layered Architecture[12]

2.11.3 Component-Based Architecture (Frontend) The frontend (React) utilizes a Component-Based Architecture. The user interface is decomposed into independent, reusable pieces called components.

- **Atomic Design:** Components are often organized from smallest (Atoms: buttons, inputs) to largest (Pages: Login Page, Dashboard).
- **Reusability:** A component (e.g., a "Product Card") can be written once and used in multiple places.
- **Encapsulation:** Each component manages its own structure (HTML/JSX), style (Tailwind), and behavior (Logic), reducing side effects on other parts of the application.

[12] https://topdev.vn/blog/kien-truc-phan-lop-layered-architecture/

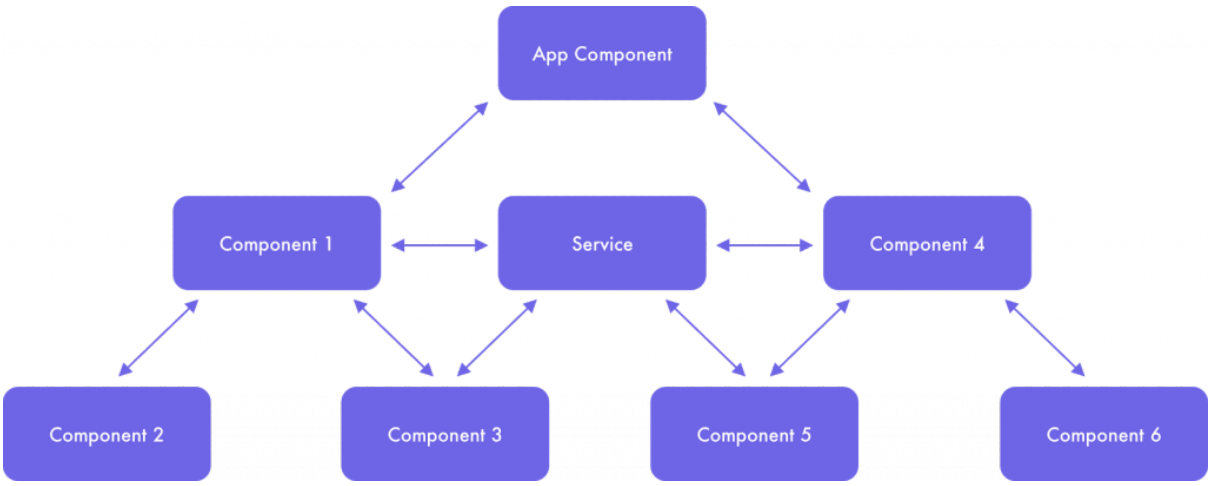

Figure 13. Component Based Architect[13]

3. Existing Systems

3.1 Existing systems

3.1.1 vLance

Introduction

vLance is currently one of the largest freelance marketplaces specifically targeting the Vietnamese market. It connects businesses with freelancers across various fields, including IT, design, and marketing, operating primarily on a bidding system.

Characteristics

- **Operating Model:** Clients post jobs, and freelancers bid with a price and timeline.
- **Local Focus:** The interface and support are entirely in Vietnamese, catering to local users.
- **Payment:** Supports local payment gateways and bank transfers suitable for Vietnam.

Advantages

- **Language Accessibility:** Being a domestic platform, it eliminates the language barrier, making it accessible for Vietnamese SMEs.
- **Local Payment:** Easy integration with domestic banking systems, avoiding international transaction fees.
- **Large User Base:** Has a significant pool of local Vietnamese freelancers.

Disadvantages

- **Lack of Quality Assurance (The "Broker" Gap):** While it connects parties, vLance operates more like a classifieds board. It does not provide a specialized intermediary

[13] https://blog.pixelfreestudio.com/implement-component-based-architecture-in-frontend-development/

(Broker) to verify technical requirements. Non-technical SMEs often struggle to evaluate the quality of bids, leading to "low-balling" and poor project outcomes.

- **Business Culture Disconnect:** Vietnamese business culture heavily relies on personal trust and clear accountability ("làm việc phải có người đảm bảo"). vLance's open bidding model often feels too impersonal and risky for traditional shop owners who prefer a managed service approach.
- **Limited Business Support:** It lacks mechanisms to deeply understand or enforce specific Vietnamese business rules (e.g., specific workflow requirements for local F&B or retail) during the project execution phase.

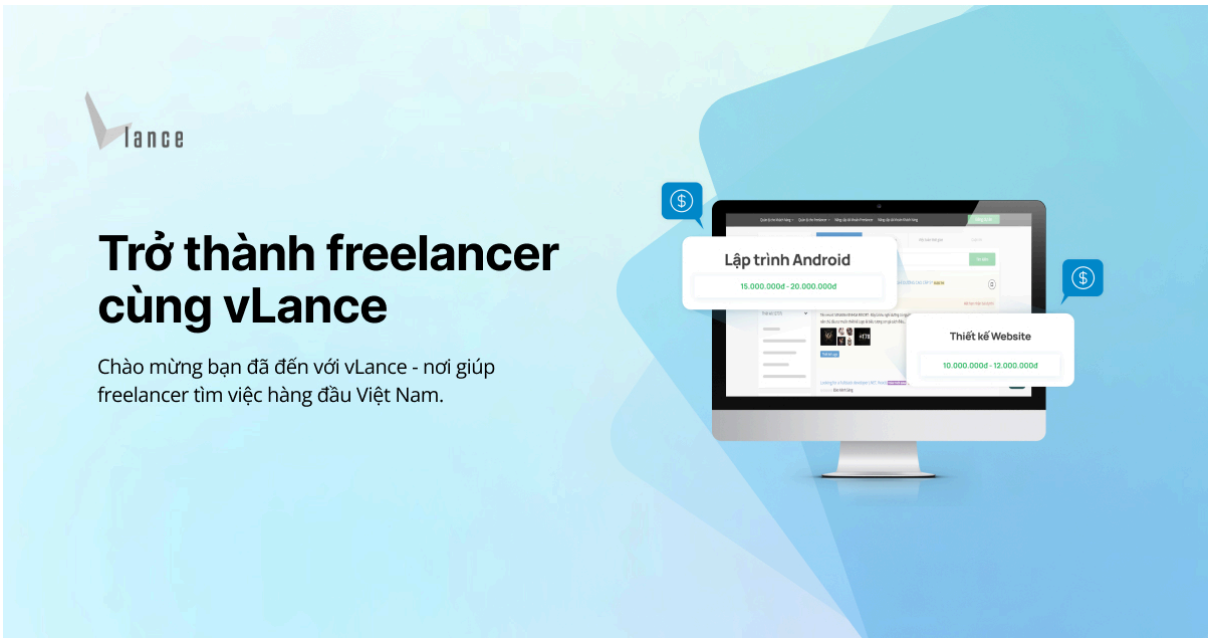

Figure 14. vLance[14]

3.1.2 Freelancer.com

Introduction

Freelancer.com is a global crowdsourcing marketplace website, allowing potential employers to post jobs that freelancers can then bid to complete. It is one of the oldest and most recognized platforms worldwide.

Characteristics

- **Global Scale:** Connects millions of employers and freelancers from over 240 countries, regions, and territories.
- **Diverse Contest Mode:** Allows clients to host contests (e.g., logo design) where freelancers submit work first, and the winner gets paid.
- **Currency:** Primarily operates in USD and major international currencies.

Advantages

[14] https://www.vlance.vn/

- **Talent Pool:** Access to a massive, global talent pool with diverse skill sets that might be scarce locally.
- **Platform Maturity:** Highly advanced features for time tracking, bidding, and dispute resolution at a global standard.

Disadvantages

- **Cultural & Language Barriers:** The platform is not "for Vietnamese by Vietnamese." The interface and communication flow do not align with the local tradition of close, high-context communication. SMEs often face difficulties conveying nuanced business requirements to international (or even local) freelancers on a global platform.
- **Regulatory & Payment Friction:** Vietnamese businesses often require VAT invoices and strict adherence to local financial regulations. Freelancer.com's international payment structure (PayPal, international wire) is cumbersome for local SMEs and often does not fully implement Vietnamese business regulations regarding taxation and contracts.
- **Disconnect from Local Market:** The platform does not cater to the specific "street-smart" business rules of the Vietnamese market. It treats a project in Vietnam the same as one in the US, missing the local context of how small businesses actually operate here.

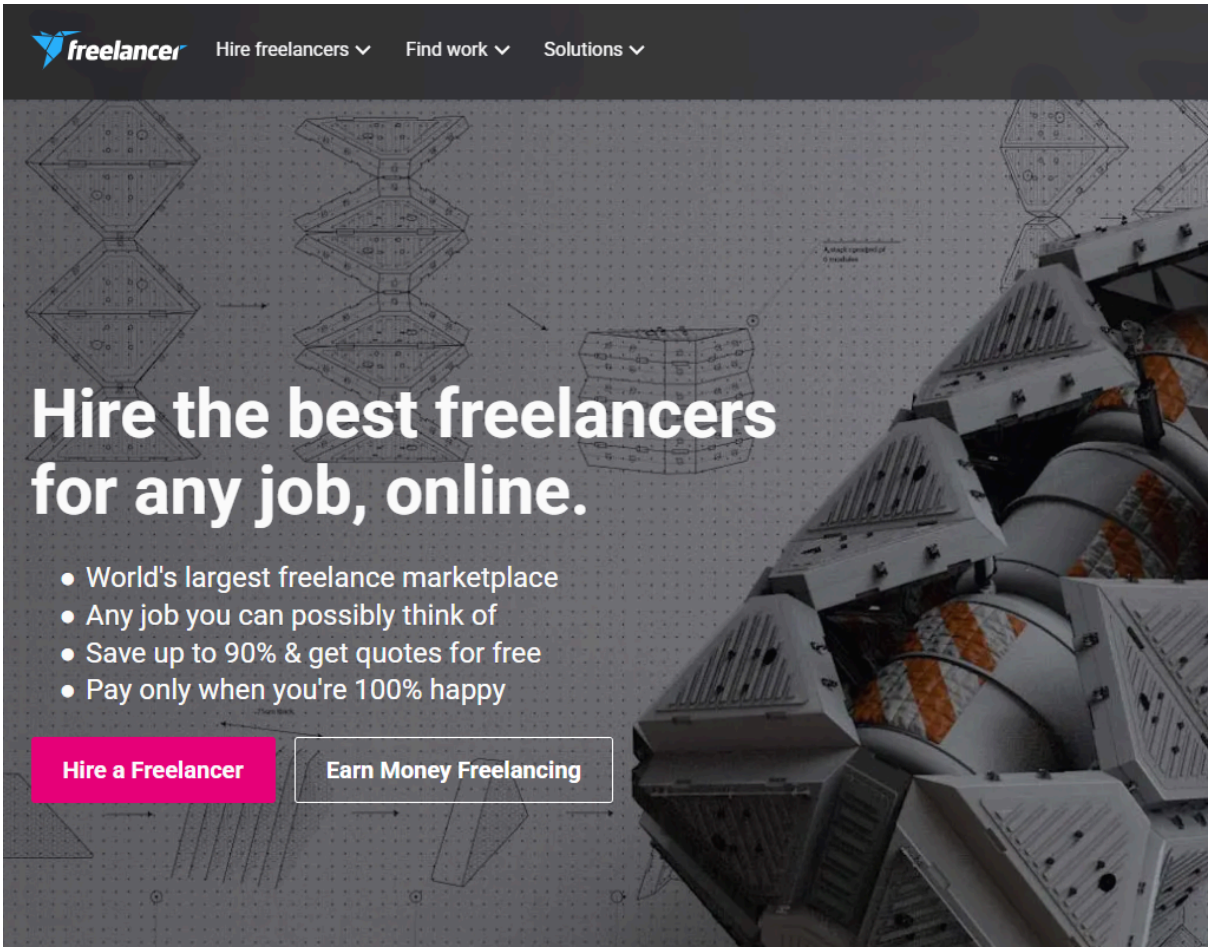

Figure 15. Freelancer.com website[15]

3.1.3 Fiverr

Introduction

Fiverr is a global online marketplace for freelance services, famous for its "Service as a Product" (SaaP) model. Unlike traditional bidding platforms, freelancers on Fiverr list their services (Gigs) at fixed prices, allowing clients to browse and purchase directly.

Characteristics

- **Gig-Based Model:** Services are packaged as products with defined scopes and prices (e.g., "I will build a landing page for $50").
- **Tiered Pricing:** Offers Basic, Standard, and Premium packages for different levels of service.
- **Speed & Transactional Focus:** Designed for quick turnaround and high-volume transactions.

Advantages

- **Simplicity & Transparency:** Clients see the exact price upfront without going through a lengthy negotiation or bidding process.
- **Ease of Use:** Buying a service feels as easy as shopping on an e-commerce site.

Disadvantages

- **"Productized" Inflexibility:** Fiverr encourages standardized, "cookie-cutter" deliverables. However, Vietnamese SMEs often operate with unique, non-standard workflows ("làm theo ý anh") that require deep customization. A pre-packaged Gig rarely fits the messy reality of a local shop's operations.
- **Transactional Nature vs. Relationship Culture:** The platform promotes quick, one-off transactions. This clashes with the Vietnamese business tradition of building long-term relationships ("làm ăn lâu dài") and expecting ongoing support after the product is delivered.
- **Payment & Compliance Issues:** Similar to other international platforms, Fiverr does not support Vietnamese tax invoices (VAT/Red Invoice), which is a critical requirement for local businesses to claim expenses. The reliance on international payment methods also creates friction for smaller traditional merchants.

[15] https://www.freelancer.com/

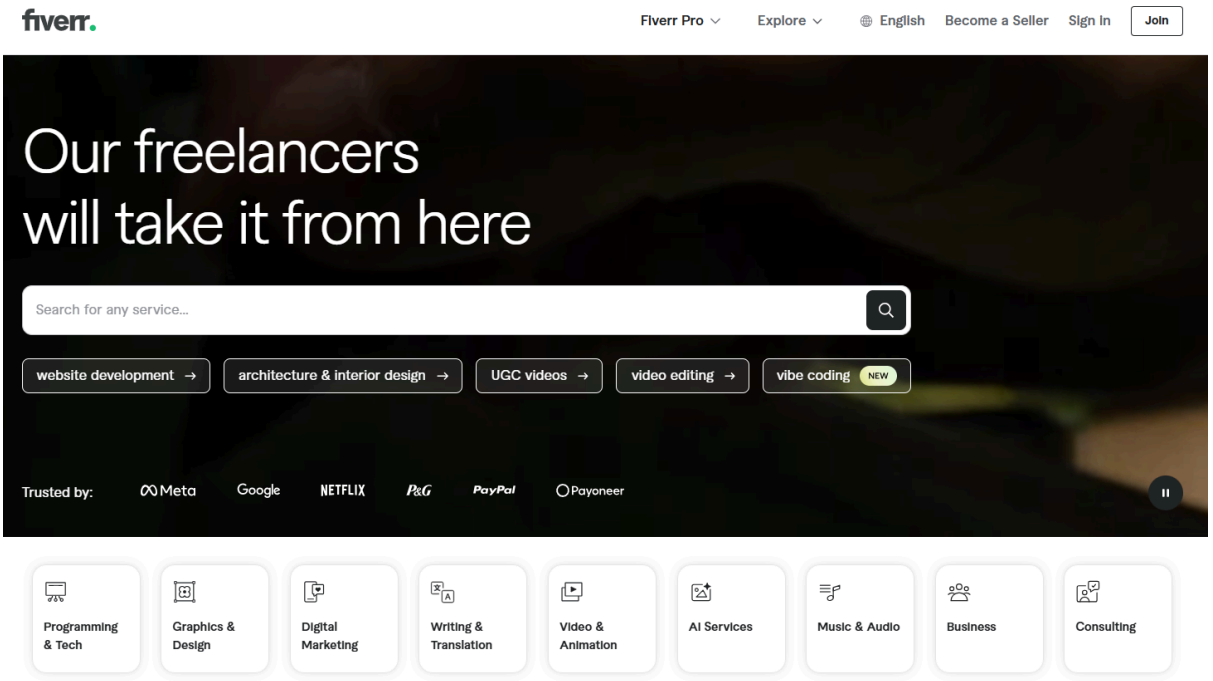

Figure 16. Fiverr[16]

4. Business Opportunity[2]

Market Context: In the current wave of digital transformation in Vietnam, there is a surging demand from Small and Medium Enterprises (SMEs), local shop owners, and F&B businesses to digitize their operations through websites, mobile applications, and management software. These businesses often have a budget for technology (typically in the range of 10-50 million VND) but completely lack the technical expertise to manage software development.

The Problem Gap: Currently, these non-technical clients are underserved by existing solutions:

- **Freelance Platforms (vLance, Freelancer, Fiverr):** These platforms operate on a "self-service" model where the client must explicitly define their technical requirements. For a non-tech shop owner, writing a clear software specification is impossible, leading to vague job postings, "low-ball" bidding, and ultimately, project failure due to misunderstandings.
- **Traditional Outsourcing Agencies:** While they offer high-quality management and Business Analyst (BA) services, their cost structure and minimum project size are often too high for small local businesses.

Furthermore, the Vietnamese market operates heavily on "trust" and accountability. The open bidding model of current platforms often feels risky to local business owners who fear scams, scope creep, or lack of support after payment.

[16]https://www.fiverr.com/

The Solution & Opportunity: The **Freelance Software Brokerage Platform** seizes this opportunity by introducing a **"Managed Freelance" model** tailored for Vietnam. By integrating a **Broker (BA/PM)** role and a **Guided Input Wizard**, the platform solves the core friction points:

1. **Bridging the Knowledge Gap:** The Guided Input Wizard allows clients to define needs via simple business questions (e.g., "Do you sell clothes or food?", "Do you need online payment?"). The Broker then "translates" these answers into professional technical specifications.
2. **Building Trust:** The Broker acts as a guarantor of quality and clarity, while the **Virtual Escrow** system ensures fair payment—money is only released when the Broker verifies the technical quality and the Client approves the business functionality.
3. **Standardizing the Process:** Unlike the chaotic "chat-and-hope" workflow of current platforms, this system enforces a structured workflow: **Draft -> Spec -> Contract -> Milestone -> Payment**.

5. Software Product Vision

The website is designed to support four main user roles: Guest, Client (SME), Freelancer, Broker, and Administrator.

Guests are visitors who have not logged in yet but can still explore the platform. They can view the landing page to understand the service flow, browse public freelancer profiles (limited view), and read about the benefits of the "Broker" model. Guests can register a new account as a Client or apply to become a Freelancer/Broker.

Clients represent Small and Medium Enterprises (SMEs) or shop owners who need software but lack technical expertise. They can register, perform KYC (identity verification), and create project requests using a "Guided Input Wizard" (instead of writing raw specs). Clients can chat with Brokers to clarify requirements, review and approve the Project Specification (Spec) and Milestones created by the Broker. They can sign simulated E-Contracts, deposit funds into a Virtual Escrow, and monitor project progress via a workspace. Clients approve milestones to release payments and rate Freelancers/Brokers upon project completion.

Brokers are the intermediaries (Business Analysts/Project Managers) who bridge the gap between Clients and Freelancers. After logging in and being approved by the Admin, Brokers receive raw requests from Clients. Their main task is to analyze these requests, communicate with Clients to refine them, and generate a standardized Project Spec. Brokers shortlist suitable Freelancers using the system's matching suggestions, manage the E-Contracting process, and oversee the project timeline. They act as quality controllers before the Client conducts the final acceptance.

Freelancers are developers who execute the projects. They create profiles showcasing their skills (Tech Stack) and experience. Freelancers receive project invitations from Brokers, view the Spec, and accept or reject offers. Once a project starts, they work within the Project Workspace, update task statuses (Todo, In Progress, Done), and submit evidence (source

code, demo links) for Milestone acceptance. They receive payments into their internal wallet once milestones are approved by the Client.

Administrators are responsible for the overall integrity of the platform. They manage user accounts, approve KYC requests, and review applications for the Broker role. Administrators oversee the entire system statistics, manage categories (skills, project types), and handle Dispute Resolutions if conflicts arise between Clients and Freelancers regarding payments or quality.

Generally, the system offers a unified digital environment where the complex process of software outsourcing is simplified for non-tech businesses through a dedicated Broker layer, ensuring transparency, trust, and quality delivery.

6. Project Scope & Limitations

6.1 Major Features

FE-01: Authentication & Authorization:

- Guest, Client, Freelancer, Broker, Staff, and Admin can register and log in using email or Google accounts.
- Users can recover forgotten passwords and manage their profiles.
- **Role-Based Access Control (RBAC):** Strict permission checks to ensure Clients cannot see Broker-only views, and Freelancers cannot access Escrow settings.

FE-02: KYC Verification & Identity Management:

- Users (Client, Freelancer, Broker) must upload ID documents (CMND/CCCD) and facial photos.
- Admin reviews and approves KYC requests to grant "Verified" badges, increasing Trust Scores.

FE-03: Broker Application & Vetting:

- Experienced users can apply to become Brokers by submitting CVs/Portfolios and experience details.
- Admin reviews Broker applications to ensure they have sufficient BA/PM skills before approving them to handle projects.

FE-04: Guided Input Wizard (for Clients):

- Clients create project requests via a step-by-step wizard (Multiple choice: Web/App, Industry, Budget, Features) instead of writing technical documents.
- The system generates a structured "Draft Requirement" based on Client inputs.

FE-05: Requirement Standardization (for Brokers):

- Brokers view draft requests, chat with Clients to clarify business needs.

- Brokers convert drafts into formal **Project Specifications (Spec)** and define detailed **Milestones** (timeline, budget allocation).

FE-06: Matching & Freelancer Selection:

- The system suggests Freelancers based on matching skills (extracted from Spec), Trust Score, and Rating.
- Brokers curate a **Shortlist** of the best candidates for the Client to review and select.

FE-07: E-Contract & Virtual Escrow:

- The system automatically generates a digital contract containing the Spec, Timeline, and Costs.
- Clients and Freelancers perform a digital "Sign" action to validate the contract.
- Clients deposit funds into a **Virtual Escrow** (System Wallet) to "lock" the deal before work begins.

FE-08: Project Workspace & Team Management:

- A centralized dashboard for the project team (Client, Broker, Freelancer).
- **Member Management:** Clients can invite internal **Staff/Supervisors** to join the workspace with "View Only" or "Reviewer" permissions.

FE-09: Task Management:

- Brokers or Freelancers break down Milestones into smaller **Tasks**.
- **Kanban Board:** visual tracking of task status (Todo -> In Progress -> Done).

FE-10: Real-time Communication (Chat):

- Built-in chat system linked to specific projects.
- Supports file attachments (images, docs) for quick clarification between Broker and Freelancer.

FE-11: Milestone Submission & Evidence:

- Freelancers submit "Proof of Work" (Source code link, Demo URL, Screenshots) when marking a Milestone as completed.
- System locks the milestone status to "Pending Review".

FE-12: Multi-step Acceptance Process:

- **Broker Review:** The Broker checks technical quality (Code, Logic) first.
- **Staff/Client Review:** Once the Broker approves, the Client (or invited Staff) checks business functionality.
- **Escrow Release:** Funds are only transferred to the Freelancer's wallet after Client final approval.

FE-13: Change Request Management (Scope Control):

- If the Client wants to add features mid-project, the Broker creates a "Change Request".
- The system calculates the additional cost/time, and both parties must digitally sign an addendum to the contract.

FE-14: Wallet & Transaction Management:

- Users can view their wallet balance and transaction history (Deposit, Withdrawal, Payment, Refund, Brokerage Fee).
- Freelancers and Brokers can request withdrawals to their real bank accounts (simulated by Admin approval).

FE-15: Automatic Fee Calculation:

- The system automatically splits the payment upon Milestone release: % for Freelancer, % for Broker (Service Fee), and % for Platform Fee.

FE-16: Rating & Trust Score System:

- After project completion, parties rate each other (Client rates Freelancer/Broker; Freelancer rates Client).
- **Trust Score Engine:** Automatically calculates user reliability based on Ratings, Project Completion Rate, and Dispute history.

FE-17: Dispute Resolution System:

- If a milestone is rejected repeatedly, either party can raise a **Dispute**.
- Admin accesses the "Dispute Dashboard" to review Specs, Chat Logs, and Evidence to make a final ruling (Refund Client or Pay Freelancer).

FE-18: Notification System:

- In-app and Email notifications for key events: New Request, Milestone Submitted, Money Released, New Message.

FE-19: User Dashboard & Portfolio:

- **Freelancer/Broker:** Display Portfolio, Skills, Verified Badges, and Earnings.
- **Client:** Overview of active projects, total spent, and action items (Approvals needed).

FE-20: Admin Dashboard & System Configuration:

- **Statistics:** View total users, active projects, total revenue (platform fees).
- **Category Management:** Admin can add/edit Tech Stacks, Project Types, and Skill Sets available in the system.

6.2 Limitations & Exclusions

LI-01: The E-Contract features are simplified; contracts are generated using static templates.

LI-02: The communication module supports only basic messaging. Real-time chat, WebSockets, file sharing, voice/video calls, and push notifications are excluded.

LI-03: Notifications are limited to in-app messages. Email, SMS, or mobile push notifications are not implemented.

LI-04: Deployment uses a single server instance with simple Docker setup. Load balancing, auto-scaling, CI/CD pipelines, and high-availability architecture are out of scope.

LI-05: The admin dashboard includes only basic CRUD and user management. Advanced analytics, reporting, charts, and export tools are not provided.

LI-06: The platform does not include a mobile application. Only the web version is developed for this project.

LI-07: Security is limited to authentication and role-based access control. Features like two-factor authentication, audit logs, penetration testing, encryption-at-rest, or SSO are excluded.

LI-08: The freelancer–client matching system uses simple filters (skills, budget, timeline) and does not support advanced recommendation engines, behavioral scoring, or learning-based ranking.

LI-09: The system cannot handle multi-tenant, enterprise-level scalability. It is optimized for small-scale use only during the capstone demo period.