

DocuWork: A Secure Escrow Platform for Document-Based Freelance Projects

Group 17

Aibek Murat, Syrymbek Zhumazhan, Dinmukhamed Assangeldi

Bakdaulet Omirzak

Project adviser: Askar Boranbayev

Executive Summary

DocuWork is the document-collaboration and secure escrow platform which brings together three primary roles of users into payers(clients), workers(freelancers) and admin(administrators). Its essence is to render documentation based works, such as translations, academic papers, legal drafts, reports, designs, and e.t.c. by keeping money in escrow until work is completed and accepted. The platform connects financial safety with a powerful record administration procedure, in having both sides to work together so that they can trust each other, trace the histories and audit their activities. The problem we set out to solve is the frequent breakdown of trust in remote , document based freelance work, such as late payments, plagiarised deliverables, also one sided processes. Our solution is an escrow based platform, with neutral admin, and versioned document storage.

Over the course of two semesters we designed, evaluated and implemented the full application. We used the Go for backend, SQLite for database, and flutter for mobile applications with payers and workers experience, also browser based web site for admin console.

The system was tested using the goals from the documentation we provided in Senior Project 1. All main features were checked, such as registration, verification, posting jobs, applying to those jobs, payment, sending documents, approval and handling disputes. We checked everything with different users, as payers and workers. Overall, the project achieved its goal and shows the full process: design, building, and testing a working system.

1. Introduction

1.1 Problem

Freelance work is mostly online and it is mostly based on documents, such as translations, essays, reports and e.t.c. Clients, who offer the work, and workers, who do the work, usually never meet each other. They only exchange files. As they do not know each other, it creates trust problems. Clients are afraid they might receive nothing or poor quality work for the payment. Workers are afraid that they might not even get paid for the work they have done. There are other applications that also work as a connection between buyers and freelancers, but their systems are not always clear. They also do not properly track document versions and might be time consuming.

1.2 Motivation

The main idea of DocuWork is to make the document the main part of trust. Every file that is uploaded is saved with its version and unique code, and all it is linked to a clear transaction record. Money is released only when the documents pass certain steps, like approval. If there is a problem, an admin can intercept it and review everything. Including file history, payment records, to make fair decisions.

1.3 Significance of the project

DocuWork is a platform that helps three groups of people at the same time and makes online work safer for everyone involved in it.

For freelancers, it gives confidence that they will be paid once they finish and deliver the work. By this it will remove the biggest fear of working online, doing the job and not getting your money. Because of this, freelancers are more willing to accept the projects from new or unknown clients and can focus more on the quality of work and not worrying about the payment.

For clients, it provides protection by making sure that they only pay when they receive the work, which is actually delivered. If there is a problem, if their work is poorly done or not complete, the client can get their money back. It reduces the fear of requesting the task for a new and unknown person. It helps to remove the fear of losing money.

For whole system, such platform creates a more honest and professional environment. All actions are recorded, and there is a clear process for solving disputes. This transparency encourages both sides to follow the rules, communicate better, and act responsible, which improves trust and overall quality of online freelance work.

1.4 Proposed solution

DocuWork is connection of three things:

- **Mobile app** - application for users. It has two sides: one for the client(buyer) and one for the worker, still both of them use the system inside. It connects the backend using secure login, and works on different devices like phones, emulators, or browsers simultaneously.
- **Admin panel** - a simple web dashboard for managing the system. It runs in the browser and is connected to the backend. From there, admin can check the users, jobs, payments, documents, disputes, and basic analytics.
- **Backend** - main server that stores the data, such as users, jobs, payments, and disputes. It is simple and fast, using the database so everything runs on a single file without extra setup. It is also designed so it can later switch to a bigger database, in future.

2. Identification of constraints

2.1 Economic

The application and web panel use simple tools, so it is easier to build and run, especially for demo versions. That is why it influenced the system to be lightweight and easy to scale by adding more servers in future. However, in future costs like payment processing and cloud storage would appear. Application and system overall is prepared for that improvement but for now it is not considered.

2.2 Environmental

If we are speaking about environmental constraints, we work using digital documents which reduces the need for physical paper and delivery. Our main environmental constraint is electricity used by servers, so we want it to be simple and with fewer resources. But in future, with the system growing it might slow down the performance of applications.

2.3 Social and Political

Freelance rules are different in each country, it might be different in European countries compared to Asian countries. That is why the taxing system, contracts and payment method might differ. To deal with it we will include flexible features such as country selection, so it can adapt to different rules later. This improves usability globally, but some legal features are not fully implemented yet.

2.4 Ethical

The main distinction of our application from others is that it focuses on fairness, transparency especially in disputes and document handling. Admin's roles in those disputes to choose the clear outcome, and all actions should be recorded, so no one can affect the results. Documents are protected with secure tracking.

2.5 Health and Safety

Since our application is just a software product, there are no physical safety risks. The main concern is to make sure the application runs without errors. To address this, we keep the setup simple and reliable, which reduces the chance of failure but may not handle large amounts of workloads.

2.6 Manufacturability

The application is easy to deploy, using a simple backend file, a simple database, application, written in flutter, and a simple admin panel. This makes demonstration simple and reduces technical problems. However, this level of simplicity doesn't satisfy the requirements for a very large scale of use.

2.7 Sustainability

The codebase is very small and uses standard tools, making it easy to understand, maintain and improve over time. This helps to improve the application in future for a new developer without any difficulties. Whereas the system might need improvement to future expansion for more advanced features.

2.8 Risks, Trade-offs and Limitations

The biggest is not incorrect handling of money, so the system should use strict rules and unchangeable transaction records to prevent errors. The key trade-off is between realism and simplicity, the demo does not use the real payment method, it just uses it as a simulation to show how it should work in future deployment. We have not deployed it yet, but we had an idea to send a 6 digit password to verify the email address during the registration process. Even though the demo shows how everything should work, we still need the work on our application to ensure that it works in each of testing situations of real world applications.

3. Background and Related Work

3.1 Background to related work

There are a lot of platforms for freelance work, such as Fiverr, Upwork, RemoteJobs and others, which also try to solve the trust process between users and workers with escrow method and milestones payments. We think that there are still problems with unclear payment methods(users still struggle with trusting to those application's working method), slow dispute resolution process(when there is a problem between users and worker, such as errors with documents or not well done job) and a lack of transparency. Delayed payment, weak certification system can lead to conflicts, communication gap, unclear ownership of documents might lower the trust between users.

3.2 Comparison

Nowadays, most freelance platforms act as an online marketplace that connects clients with independent workers. Some of those platforms work as a market where workers(freelancers) post their abilities and work they can do, so users can purchase and pay them to do those works. Also, there are other platforms where users post the jobs they need to be accomplished, and workers bid for it, like auctions. The main idea of our application is to remove the troubles between users and workers, and reduce transparency. In our project we chose a structured escrow method for payment and clear dispute resolution states. Compared to other platforms, our application emphasizes the simplicity of connection, control of financial actions, and reducing the risk of unfair outcomes.

3.3 Escrow method and Dispute Management

The escrow method is used to make payments safe for both parties. Instead of sending money directly to workers, it is deposited to the system, where it is held securely. After the worker finishes the job, the client should approve the result then money is released. To reduce the risks for both sides, the worker knows they will get paid, and the clients will know that they will not lose money without receiving the work. In our application, escrow method is strictly controlled by the backend, and all transactions are recorded in a

clear and unchangeable ledger. This way it improves the transparency and prevents errors or fraud.

The dispute management system is used when there is a conflict between clients and the worker. For instance, if the client is not satisfied with the result or when the worker claims that work is done correctly and the client is not approving it. In our application, disputes are handled in a structured way. Each dispute includes three sides: the client, the worker, and the admin. The admin is a one who solves those disputes and those decisions are automatically applied and recorded. This prevents hidden actions and ensures the fairness between users and workers. The application stores all the information related to documentation and works, so there is no misunderstanding and improves the trust between application users and the freelancers.

3.4 Computing-Based solutions

To analyse the existing literature with cloud based solutions, our project tries to imply stateless backend design, structured data management and secure authentication. We also try to protect the documentation and files by version control and hashing to ensure integrity. By analyzing the current applications we came to the conclusion that today's solutions have a common weakness and we want to address them with simple but reliable architecture. We think that computing-based solutions can improve trust, reduce risks, and create efficient workflow in freelance platforms.

4. Project Approach

4.1 System Architecture

The DocuWork platform is a modular service-oriented system, which provides an opportunity to exchange documents securely with the help of an escrow mechanism. There are three major parts of the architecture, a centralized backend, web-based administration panel, and cross-platform mobile application. All these elements collaborate to achieve consistency, reliability, and security throughout the platform.

Backend Layer:

The back-end is the heart of DocuWork and is the place where all the primary logic of the system is contained. It offers API endpoints which are accessed by the mobile application and the administration panel. The back-end is in charge of the user authentication, document management and above all the escrow process. The escrow logic is used to make sure that the documents can be released only under the condition that all the necessary conditions between users are fulfilled. It also handles transactions, monitors their status and updates the clients. To make the system maintainable and scalable, the backend is divided into distinct request handling, business logic, and data access layers.

Database:

Any data that is persistent in the system is stored in the database. This entails user accounts, document data, and escrow records and system logs. It keeps track of the state of each transaction and ensures data consistency. Here security is significant and as such access to data is managed according to the role of the user, and sensitive data is secured.

Web application (Admin panel):

The system administrators use the admin panel, which is developed on Next.js. It enables them to oversee platform activity, administer users, walk transactions, and deal with special cases such as disputes. The web application will be connected to the backend via secure APIs and will only be accessible to authorized individuals.

Mobile application:

The mobile app is built on Flutter and is consumed by the average users of the platform. It offers registration, login, document uploading, and taking part in escrow operations. Users are able to monitor the status of their transactions and get updated real time information. The app uses API requests to communicate with the backend and is compatible with iOS and Android.

4.2 Roles

- Payer - can post jobs, fund escrow, accept applications for posted job which were accepted by workers, request revisions and open disputes with the admin. Access controlled by the Role("payer").
- Worker - can browse open jobs, apply to those jobs, open disputes with admin. Access controlled by Role("worker").
- Admin - can see every user, every job, every transaction, every document. Approves or rejects new accounts and resolves the disputes between payer and worker. Access controlled by role("admin").

4.3 Key Features

- Registration with strong password enforcement(upper letter, lower letter, number, 8+ characters)
- Admin approval of new applicant
- JWT-based login
- Job creation, listing, detail view of jobs, filtering by status of work, category, budget
- Job application: Worker applies for job, payer accepts the worker, escrow funds.
- Document upload, workers can see after clients approve their application
- Escrow state, with transitions written to an append-only ledger
- Dispute Lifecycle: opened by user, resolved by admin
- Admin analytics: weekly escrow, project status, active disputes, message from users, verification queue

4.4 Third-party Components

4.4.1 Backend(GO)

The system has a backend that combines multiple third-party technologies that guarantee scalability, performance, and reliability.

Amazon Web Services (AWS):

The platform is cloud-based on Amazon Web Services. Specifically, Amazon S3 is used to store media files, such as big documents uploaded by the users. This will enable the system to manage file storage effectively without congesting the main server.

Redis:

Redis is in-memory cache that is used to enhance the performance of the system. It keeps common data in key-value format which enables it to perform more quickly the GET and SET operations and less load on the database.

PostgreSQL – is also used in the backend as a main database.

The main database of the system is PostgreSQL. It ensures consistency and reliability in storing the structured data that includes user information, transaction records, and system data.

Docker:

The backend application is containerized using Docker, together with its dependencies. This provides a stable environment among development, testing and production and makes deployment and scaling straightforward.

JSON Web Tokens (JWT):

Authentication and authorization are done with the help of JSON Web Token. Once successfully logged in, users are given a token to be used in future requests, and the backend can authenticate user identities safely without having to store server-side sessions.

4.4.2 Flutter Mobile Application

The mobile app is created with the help of Flutter that supports cross-platform development on iOS and Android based on the same codebase. This saves time in development and provides a similar experience to the user on different devices. The app interacts with the backend via RESTful APIs, managing user authentication, document management, and escrow transactions. The token-based authentication ensures secure communication, enabling users to use the platform safely. The reactive framework and built-in UI components of Flutter are employed to design an interface that is responsive and smooth. The mobile application architecture is created to be separated into presentation, business logic, and data handling, enhancing maintainability and scalability.

4.4.3 Admin Panel

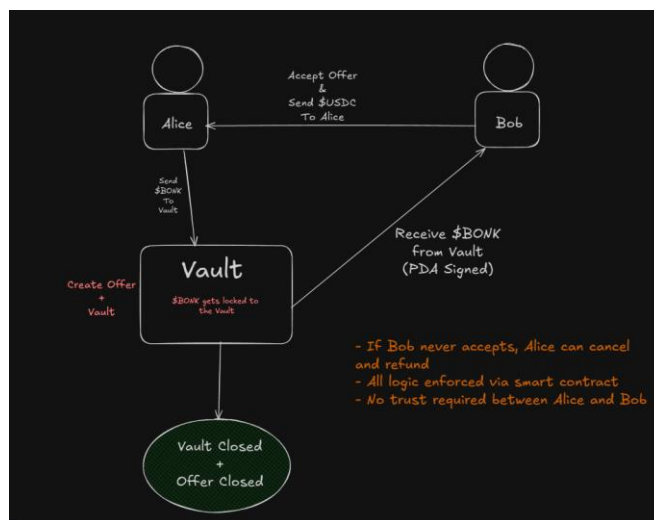
Next.js is used to create the admin panel, a web-based interface of the system administrators. It offers system activity monitoring and user management tools, as well as escrow management. The admin panel will communicate with the backend through secure API calls and can only be accessed by authorized staff. It enables administrators to look up transaction details, settle disputes, and keep the entire system in check. Next.js allows efficient rendering and performance optimization and allows scalable and maintainable frontend development.

4.5 Algorithms and Workflow

The DocuWork application does not rely on complex algorithmic techniques such as machine learning. Instead, the system is built around clearly defined workflows and state transitions that ensure correct execution of escrow-based document exchanges. The core logic focuses on managing processes, enforcing rules, and maintaining consistency between different stages of user interaction.

4.5.1 Escrow state machine

The escrow mechanism is applied to a state machine governing the lifecycle of every transaction. An escrow request is initiated by one of the users who sends an appeal and uploads a document. After the second party has accepted the terms, the transaction enters into an active state. In the process, the system will track various states that include pending, active, completed, cancelled, and disputed. Transfer of these states is highly regulated by predefined conditions, such as agreement by all parties or deadline expiry. The document is not issued until the transaction is in the completed state, and the parties are in their obligation.



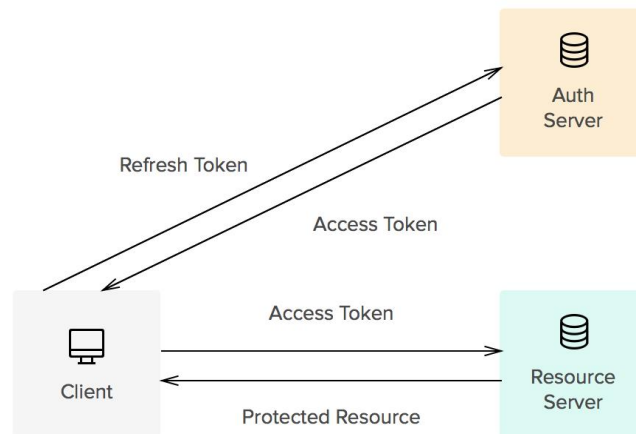
4.5.2 Job workflow

The job workflow represents the sequence of actions performed by users within the platform. A typical flow starts with a user creating a task or agreement, followed by document submission and escrow initialization. The second party then reviews the terms

and either accepts or rejects the request. If accepted, both users proceed with the agreement, and the system continuously updates the transaction status. Once all conditions are satisfied, the workflow concludes with the release of the document. This structured process ensures transparency and minimizes the risk of misuse.

4.5.3 Authentication

Authentication within the system is done through JSON Web Token. On successful user login, a token is created and sent to the client. This token is part of the later requests to confirm the identity of the user. The token-based authentication approach enables the system to be stateless whilst still being secure. The control of access to resources is determined by user roles, so that a user can only do what he or she is authorized to do.



4.5.4 Dispute

Dispute workflow is initiated when any of the parties is dissatisfied with the result of the transaction. In those situations, the transfer is placed in a contested state, which temporarily blocks the publication of the document. In this step, all the information related to the case including the user actions and timestamps are logged in the system to

aid in resolving the case. The conflict is not yet resolved and it has to be re-examined by an administrator. This will be a fair mechanism and a fallback in case of conflicts.

4.5.5 Admin Moderation Flow

The flow of the admins moderation enables the administrators to monitor and regulate activities in the platform. In case an irregular activity or a dispute is identified, the administrators are able to examine the details of the transaction, including documents, logs, and user actions. Depending on their decision, the administrators may either approve a transaction, cancel it, or decide to resolve disputes in favor of a party. This centralization of control makes the platform reliable and the problems dealt with in the same manner.

4.6 Use case diagrams

#	Actors	Use case
1	Payer + Worker	Register new account with strong password
2	Payer + Worker	Verify the account
3	Admin	Approve or reject the applicant
4	Payer	Post a job and fund escrow
5	Worker	Browse jobs and apply for jobs
6	Worker	Upload finished document
7	Payer	Approve the work or request revision
8	Admin	Resolve dispute

4.7 Team functionality

Every teammate of the group had their role on this project, we all functioned as a whole group team to try to accomplish our goals specified at the beginning of first semester and at the end of it. Work was tracked on a shared github projects board. We shared our thoughts on work done by our team members, we also used telegram chat as a source of communication, where we forced each other to contribute towards completion of our senior project. We used Figma to create the design of our web panel for admin and back end to overall work of everything. We write code in flutter for mobile application development, and use emulators on android studio to run it.

5. Project Execution

5.1 Work done by in the first semester

The team developed the architecture of DocuWork at the backend, mobile application, and the administration interface. On the back end, the base Go project architecture was established, rudimentary authentication systems were created, core database schemas were set up, document-uploading integration with the Amazon S3 was built, Redis configuration was run and basic API connections were written. Also, a prototype implementation of the escrow state machine and a simple notification system were implemented.

The app reached the prototype phase and included onboarding screens, a main dashboard, API connection, secure storage of tokens, and first versions of document preview and chat features.

Its administration panel, initial analytics dashboard, user and dispute management views and a responsive interface were integrated by the administrative panel. Design work included full Figma mockups, screen flows, and UI kits that could be used in the mobile and web platform. The results of documentation were functional requirements, architecture diagrams, UML representations, and draft API specifications, which provided a strong foundation to the further development.

5.2 Work done by in the second semester

In the second semester, we turned the mock up into a real working system. The main things we did:

Backend (Go + Postgres). We build everything from scratch, role based access, jobs, disputes, escrow, user accounts, login. The whole backend was developed by language Go, and we built the database using postgres.

Mobile Application - We built a whole mobile application using the programming language flutter(dart). In the first semester we only had mock data. In the second semester, we added a registration page for new applicants, a login page, a user page, a worker page, the ability to write to admin, job search, job post, job approval, profile, and other things we said at the end of the first semester we should do.

Admin panel - We build web pages using programming languages HTML and css, node js, react and typescript. We have a login page for admin, verification queue for new users, data of application users, support chat(two way chat between any user and the admin).

5.3 Design Decisions and Changes

When we began the project, we resolved to develop the system in a straightforward and understandable format. We separated it into parts: frontend, backend, and database. This assisted us to work simultaneously and the project was easier to handle.

We also selected technologies that were familiar to the majority of the team. This saved us the time of having to learn new tools and enabled us to concentrate more on the project construction.

The other significant decision was the addition of various user roles. Initially we intended to have a very simple system of logging in but we enhanced it later to allow various users to have varied permissions. This rendered the system more helpful and secure.

We needed to make some adjustments to the original plan during the work. We had to cut some features or delay them due to lack of time to complete all features. We also altered the database structure when we realized that the initial version was not efficient.

The user interface was altered as well. The initial design was more complex, yet based on the feedback, we simplified it so that it would be more accessible to users.

5.4 What went wrong/right

What went right:

The team was a team of good people. Work was divided equally, as everybody had their own activities. We also shared a lot, and this enabled us to solve problems quickly.

The other positive aspect was the system structure. As we separated it into parts, it was easier to work and test the parts separately.

An early working version was also made by us. This helped us to detect mistakes early and rectify them before the final copy.

What went wrong:

Time was one of the issues. We had not expected that it would require such a long time to get all the components of the system together and test them.

The other issue was that we added more features during the development. This made the project bigger than the initial and caused delays.

There were minor communication problems among team members working on other sections in some cases. Due to this, not all the parts fitted perfectly and needed to be repaired later.

In the end, we were able to work out these problems and complete the project successfully.

5.5 Teamwork aspects

Responsibilities were divided between members so that each member contributed towards development of the project. Each member did their part of work and towards completion of the project report and presentation.

Member	Primary Focus	Secondary
Aibek Murat	Mobile Application: flutter	Admin panel: next js and react
Syrymbek Zhumazhan	Go backend: for mobile app, admin panel	Next js and react (typescript)
Dinmukhamed Assangeldi	Mobile Application: flutter	Web page: admin panel
Bakdaulet Omirzak	Web page: Admin panel(html and css)	Go backend: admin panel

6. Evaluation

6.1 Evaluation Strategy

The testing of the DocuWork platform was aimed at confirming its efficiency as a secure document exchange system based on escrows. Because the project is not associated with complicated computational algorithms, the evaluation strategy was focus-based on the usability, trust in the working process, and the trust of the users in the escrow system.

In order to do so, a mixture of user testing and scenario validation was used. The system was also tested with real-world use cases which simulates how the users would interact with the platform in the real world. These situations involved the drafting of an escrow transaction, document uploading, acceptance of agreements, finalizing transactions and dispute resolution.

Besides functional validation, a sample of the potential users was invited to engage with the system using the mobile application created using Flutter and the administration panel created using Next.js. Participants were expected to give feedback on the platform after using it in the form of a structured questionnaire. By doing so, not only the system correctness was evaluated, but also user experience and workflow clarity. System performance was also assessed based on response time and stability during normal use, especially when connecting with the backend services as well as file storage using Amazon Web Services.

6.2 Clearly illustrate how the evaluation process validates the computing-based solution

The evaluation process showed that the system is able to serve its main purpose of facilitating secure and reliable exchange of documents based on an escrow mechanism. The escrow state machine made documents released only when all the conditions needed were fulfilled, preventing instances of illicit access or untimely completion of transactions. The scenario-based testing ensured that the key workflows are functioning as intended. The customers could make payments, post documents and go through with each phase without discrepancies. Change in various escrow statuses like pending, active, and completed was done in the right way depending on the user action.

Authentication based on JSON Web Token was confirmed by verifying that only authorized users have access to the authored endpoints and are allowed to do actions that are within their

authorizations. This ensured that the system has adequate access control and data security. The conflict management system was also tested through the simulation of conflict situations. In such situations, the transactions were properly transferred to a contested state, and before it could be resolved, it had to be sorted out by the administration. This proves that the system is capable of dealing with edge cases and ensuring fairness among the participants.

All in all, the analysis proves that the introduced architecture and workflows help to address the issue of trust in document exchange successfully as they enforce rules using the system instead of depending on the behavior of users.

6.3 Include any data collected for the evaluation and give a brief analysis of it. In some cases, the group may need to have potential users try out the developed system and provide feedback using questionnaires as part of the evaluation. This is NOT about what tests the group ran to help detect bugs in the system as part of the normal development process.

To back up the evaluation, a sample of 15 users who tried out the system was used to gather feedback. The participants were students and other people who are conversant with digital platforms and may not be conversant with escrow-based systems. The participants were given a series of predetermined tasks and asked to answer a questionnaire. The gathered information demonstrated that about 87 percent of the users managed to complete all the tasks without any help, which means that the system is fairly user-friendly. Approximately 80% of respondents indicated that the concept of escrow was simple to comprehend once they used the application, though some people felt that the concept was not known to them. Regarding user experience, 85% of users rated the mobile application as being easy to use and responsive.

Nevertheless, a few users noted confusion when creating a transaction, and it may be possible to make this part of the interface more user-friendly and easier to understand with better instructions or visuals. Feedback based on performance showed that the majority of actions, including uploading documents and transaction updates, were made within reasonable time frames. This implies that it can integrate with the external service (cloud storage) effectively and without harming user experience. There was also high trust in the system among the participants

when the tasks were done. About 78 percent indicated that they would be comfortable in such a platform to transact real documents, which demonstrates that the escrow mechanism could be effective in enhancing perceived reliability.

Overall, the data obtained demonstrates that the system achieves its functionality objectives and offers users a favorable experience. Although a few usability tweaks can be done, especially in the onboarding and transaction setups, the overall assessment proves that the solution is efficient, secure, and can be applied in real-life.

7. Conclusion and possible future work

7.1 Key findings and contributions

This project saw the design and implementation of DocuWork, a freelance collaboration based on documents, which is a secure platform. The primary aim was to overcome the issue of mistrust between clients and freelancers that is one of the most widespread problems in the field of online work.

We were able to show through our work that a combination of escrow-based payments, document version control, and the use of structured dispute resolution can go a long way towards enhancing transparency and reliability in freelance interactions. Our system does not operate solely on the basis of user to user communication but rather creates a transparent and traceable workflow in which all the actions, such as job creation, document submission and payment release are documented and connected to each other.

During the two semesters, we were able to come up with a complete functioning system consisting of a mobile application, an administrator panel, and a backend server. It was configured with all its core features and tested, such as user registration, role-based access, job workflows, state transitions of escrows, document uploads, and handling disputes.

Among the most significant contributions of this project can be viewed as demonstrating that with a simple and lightweight architecture it can be constructed a system that not only models processes that happen in the real world but also solves real-world problems. We use a

strategy of transparency, equality, and accountability, which are critical in establishing trust on the digital platforms.

7.2 Limitations

The project has a number of limitations, although it has fulfilled the primary goals.

To begin with, the present system incorporates a simulated escrow system instead of actual payment integration. Although it enables us to illustrate the workflow, it is not entirely realistic to real-world financial transactions and security requirements.

Second, the system is developed on a lightweight database and local deployment, which is appropriate for demonstration but might not scale up with a large number of users or heavy workloads.

The other weakness is associated with legal and regional diversities. The laws of freelancing, taxation, and contract laws differ across countries, and we are not fully dealing with these issues in our existing system.

Moreover, there are certain features like email verification, advanced analytics, and cloud-based storage which were to be implemented but not entirely realized in the existing version.

7.3 Future Work

DocuWork can be enhanced and extended in a number of directions.

Among the most significant measures is the integration of a real payment system, e.g., Stripe connect, to facilitate secure escrow payments and multi-currency payments. This would render the platform applicable in the real world.

The other enhancement is the moving of document storage to cloud computing such as AWS S3 that would enable easy scalability, reliability and versioning of large files.

We will also improve the dispute resolution system by introducing structured evidence timelines, improved communication tools, and pre-defined templates of decisions to admins.

Technically, the backend can be scaled up to a more scalable database like PostgreSQL and the system can be deployed to a cloud environment to support the increased traffic.

Last but not least, we believe in the possibility of implementing AI-based functions, including plagiarism detection or document quality analysis, to further assist with decision-making and enhance the trust between users.

8. References

- [1] Stripe, *Stripe Connect: Payments for platforms*. Stripe, 2025. [Online]. Available: <https://stripe.com/connect>
- [2] The Go Authors, *The Go Programming Language*. Go.dev, 2025. [Online]. Available: <https://go.dev/>
- [3] Kotlin Foundation, *Kotlin Multiplatform Overview*. Kotlin, 2025. [Online]. Available: <https://kotlinlang.org/>
- [4] Meta Open Source, *React: A JavaScript Library for Building User Interfaces*. React, 2025. [Online]. Available: <https://react.dev/>
- [5] DigitalOcean, *App Platform Documentation*. DigitalOcean, 2025. [Online]. Available: <https://www.digitalocean.com/docs/app-platform/>
- [6] J. F. Brazdil, “A Practical Guide to Escrow and Dispute Management in Online Marketplaces,” *ACM Digital Library*, 2021. [Online]. Available: <https://dl.acm.org>
- [7] OWASP Foundation, *Authentication and Access Control Best Practices*. OWASP, 2025. [Online]. Available: <https://owasp.org/>
- [8] Amazon Web Services, *Amazon S3 Security and Storage*. AWS, 2025. [Online]. Available: <https://aws.amazon.com/s3/>

