Software Requirement Specification

Problem Statement: A few sentences to describe the problem you are trying to solve, i.e., justify why this software is needed.

Hopkins students often need temporary or flexible work opportunities, while others may require assistance with tasks like tutoring, dorm cleaning, or custom dorm decor. However, there is currently no dedicated platform that connects students for these peer-to-peer services in a trusted, structured way. This app aims to provide a centralized platform for students to post and accept job listings, ensuring a seamless, secure payment process through smart contracts. The goal is to foster a community-driven, convenient solution for students to offer and find services within the Hopkins ecosystem.

Potential Clients: Who are influenced by this problem and would benefit from the proposed solution? (i.e. the potential users)

1. Hopkins students seeking flexible work opportunities.
2. Students looking for peer-provided services such as tutoring, dorm cleaning, etc.
3. University organizations or student groups that might need temporary help from peers.

All of these respective groups would benefit from our solution, as our centralized platform for student job postings would bring these groups together to solve a common problem. Additionally, the students seeking the provided services would save valuable time, leading to decreased stress and overall a better state of well-being.

Proposed Solution: Write a few sentences that describe how a software solution will solve the problem described above.

The web app we propose is a freelance marketplace that will support a secure payment process, like other existing freelance marketplace platforms. The difference is that this web app will require that users be Hopkins students. This will result in an experience tailored to Hopkins students, with job and freelancer pools localized to the Homewood campus.

Functional Requirements: List the (functional) requirements that software needs to have in order to solve the problem stated above. List these in role-goal-benefit format. It is useful to try to group the requirements into those that are essential (must have), and those which are non-essential (but nice to have).

Must have

(1) User Profile

Role: Provider/Requester users

Goal: To create and manage a user profile with basic info(name, bio, contact details)

Benefit: Allows users to find information regarding other users, making it easier to connect with other users

(2) Job Listings

Role: Provider

Goal: To create task listings with relevant details (summary, price, availability / how long job will take)

Benefit: Provides detailed information to other users allowing them to perform job responsibilities

(3) Job Listing Acceptance

Role: Requester

Goal: Users can accept a listed job from the Job Listing Board that they want a provider to complete

Benefit: Allows users to accept jobs from providers

(4) Escrow System for Payments

Role: To pay for tasks upfront with the payment held in escrow until both parties confirm job completion.

Benefit: Increases trust and security by ensuring payment is only released after the task is satisfactorily completed.

(5) Ratings and Reviews

Role: Requester and Provider

Goal: To leave feedback for completed tasks

Benefit: Gives rating system allowing users to make informed decisions when selecting whether to hire a person or complete a job for someone

(6) Job Search

Role: Requester

Goal: Allows users to search for task listing

Benefit: Allows users to quickly find relevant services or tasks that meet their needs without extensive browsing

(7) Smart Contracts and Crypto Wallet Integration

Role: Provider and Requester

Goal: To facilitate a decentralized payment through smart contracts and cryptocurrencies

Benefit: Eliminates the need for middlemen or worry about money withholding, providing a secure, efficient payment process

(8) In-App Messaging

Role: Provider and Requester

Goal: Can Communicate directly through the app

Benefit: can communicate timing and clarification details without the need of external tools

Nice to have

(1) Job Filtering

Role: Requester

Goal: Can filter based on service type, price, availability, rating, proximity, and experience level.

Benefit: Further improves the way Requesters can discover job listings

(2) Dispute Resolution

Role: Provider and Requester

Goal: To resolve disputes between users

Benefit: ensures fairness and provides a safety net for users in case of disputes

(3) Task Photos

Roles: Provider

Goal: Upload photos of completed jobs or previous work

Benefit: Helps requester make more informed decisions by showcasing the provider's work

(4) Job Desire Posting

Roles: Requester

Goal: Allow Requester to post jobs they want and for what price they are offering if there is not anything they see available

Benefit: can ensure any job desired can get completed

(5) Push Notifications

Role: Provider and Requester

Goal: To receive notification for important updates(job requests, job acceptance, payment, etc.)

Benefit: Keep users well informed and ensures timely action

(6) User tutorials

Role: Provider and Requester

Goal: To provide tutorial for first-time users

Benefit: ensures that those who want to use the app understand its functionality and features

(7) Multiple Payment Options

Role: Provider and Requester

Goal: To offer other forms of payment beyond crypto like Venmo or credit card

Benefit: Allows for people to pay in whatever method they have the most comfort with

(7) Job Scheduling

Role: Provider and Requester

Goal: Allow scheduling for jobs based on availability of both parties

Benefit: Improves coordination between providers and requesters

(8) Skill Verification

Role: Provider and Requester

Goal: Allow providers to verify their specific skills for requesters.

Benefit: Further improves trust and safety within the application.

Continues on the next page

Non-functional Requirements:

(1) Performance

Role: Requester

Goal: The system should be able to maintain a smooth response time under high concurrency.

Benefit: More smooth user use, improve the user experiences.

(2) Security

Role: Provider and Requester

Goal: The system must ensure the security of user data and prevent unauthorized access and data leakage.

Benefit: Ensure user information security.

(3) Usability

Roles: Provider and Requester

Goal: A clear website UI that allows users to get started quickly and provides good navigation.

Benefit: Improve the user experience

(4) Compatibility

Roles: Provider and Requester

Goal: The system should be compatible with multiple devices and browsers, including desktop, mobile, tablet, etc., and maintain consistent performance on major browsers.

Benefit: Make sure the site still looks good for users across different browsers and devices

Software Architecture & Technology Stack: Will this be a Web/desktop/mobile (all, or some other kind of) application? Would it conform to specific software architecture? What programming languages, frameworks, databases, …, will be used to develop and deploy the software?

This application will be a web app. It will be a client-server architecture, where both the client and the server can interact with the blockchain. Using the MetaMask chrome extension, the frontend will be able to directly create new instances of the job smart contract and submit them to the blockchain. Using the Infura API, the backend will be able to receive updates on outstanding smart contracts. When a contract is created, it's address will be stored in the Postgres database, and served from the backend to the frontend when receiving additional user inputs regarding that contract.

Frontend:
- React.js for user interface
- Ether.js for initializing smart contract / interacting with blockchain
- MetaMask chrome extension required for user's wallet address to be available in smart contract
- Vercel for deployment

Backend:
- Python Django web server (AWS EC2)
- Infura API to listen for smart contract updates on blockchain given smart contract address
- Postgres for database (AWS RDS)

Smart Contract:
- Solidity

Docker:
- Containerize frontend and backend for deployment

Similar Apps: List a few similar applications to the one you are developing. Don't be eager to conclude no similar app exists! There is always something similar to what you are building! Finding those will help you to better specify your project. ***You must be prepared to explain how your app is different from the existing ones.***

Ethlance, LaborX, CanWork, DeFiHub, Upwork, Fiverr, Freelancer.com, Blackbear, Superside, TopTal, Malt.