I.2. System Specifications

Table Of Contents 👇

---

2. System Specifications

In the **system specifications document**, we are going to speak about the five pieces of information. First we discuss what are the reasons that such a system is needed from the business perspective in the Business Case section. Then, in the System Description section, we describe the main features of the application. Later, in the Assumption section, we discuss the pre-existing conditions that we assumed are there in place before starting the development. In the fourth part, actually Requirements we reflect the whole requirements that we have received from the customer. Finally, in the Wish List section, we determine which part of requirements are going to be planed for the future revisions and are not going to be delivered in our implementation.

2-1. Business Case

By spreading the internet and the smart phones, everyday some new applications are appeared to change the way that we deal with daily basis problem. Although, the need for new applications are growing constantly through out the world, however, the number of IT companies are not even in different part of the world. Therefore, it is needed to have some systems to connect clients to providers through out the world. Our **B2B Research Matchmaking** is a bridge between clients and providers.

Providers can reveal their abilities by providing their resume and some topics that they are interested in and have enough power for implementing projects in that area.

In the other side our **Clients** can search in our database, review the resume of different providers, select them and bid a price for a specific project.

If any deal happen during this process, then the selected **Provider** can enjoy *70%* of the deal, the **Client** can have a nice functional application on time, and **the System** can continue its operation using the rest of the *30%* of the deal.

2-2. System Description

The System contains four agents. First, **the Host** or as it has been called in the requirements **the System**. This agent is responsible to *store* data, *moderate* messages and *connect* the other agents. Then **Providers**, which *provide* services by *introducing* themselves via yellow pages. In contrast, we have **Clients** which *search* and *consume* services. Finally, we have **Guests** that are a limited version of **Clients** which are more for *discovering* and *evaluating* the main functionality and examining the quality of data in our database. If a **Guest** find the

information and services interesting, they *can register* and *become* a **Client**. Being a **Client** means they can start *biding* **Providers**.

As mentioned before, **the System** *charges* providers with *30%* commission. Therefore, besides of registration and searching modules, there must be an accounting module which *keeps track of* financial transactions.

All actors, can access to their dashboard via a login mechanism which is not delivered in the first phase. In fact, we mock the login by simulating the sign in mechanism.

Last but not least, we are not going to deliver the chat, commenting and rating features. We will discuss them in our design phase, however, in the implementation phase we exclude them for the first phase.

2-3. Assumption

In this section, we discuss the pre-existing conditions that we assumed are there in place before starting the development and we build our design by assuming them.

1. In our GUI, we won't provide any login mechanism for our actors. We would have a database which keeps records of our **Providers** and **Clients**. When already registered users want to login, they can just select their name from a list and press on the login key and then sign in. We assumed they don't abuse this situation and each use only login to his or her account.
2. For persisting the data, we will use one of the embedded databases in Java. A few of the dominant providers are available for choosing (i.e., H2, HyperSQL, Apache Derby, Berkley DB, Java DB, ObjectDB, and so forth. However, in this stage we don't know which database will be selected. We will select one due to the performance, price, and the license.
3. We have a specific policy and guideline for using any OSS component during our implementation. If we need to leverage any OSS component, we must make sure that we fulfill these steps:
    1. First of all we only accept libraries and components that have MIT license.
    2. If any OSS component does not have MIT license we must not use them.
    3. We must provide an static copy of the jar file in a separate folder in the lib directory. For example, for using a jar file as `xyz.jar`, we must make a sub folder `xyz` inside the `\lib` directory and put the jar file in it.
    4. We must provide the license file beside of the jar file.
    5. We must avoid copy/paste any snippet of code from any website or mailing list, specially Stackoverflow.com.

2-4. Requirements

In this section, you can find the requirements. Each actor (i.e. agent) has been coded by **bold** style. Each action/verb has been declared by *italic* style. Finally, each attribute has been identified by underline style.

1. Ability to *sign up* as **Provider** and **Client**.
2. Ability to be a **Guest** and *visit* the app.
3. For **Providers**: ability to *submit* name, website, logo, resume, special keywords, hourly compensation.
4. **Providers** *can get* a verified icon if they *send* their proof of business to **the System**.
   The System should *make sure* that every piece of information *is correct* and then *accept* the request.
5. For **Guests**: ability to *search* keywords and *get* a list of available Providers.
6. A contract should be *sent* to a **Provider** the moment they *sign up*.
7. **Provider** should be able to *accept* or *reject* the contract.

8. Upon rejection, the **Provider** will be automatically *converted* to a **Client**, *losing* their resume, website, special keywords and hourly compensation information.
9. When a **Guest** *visits* the **App**, they can only *see* the name, website, logo, resume, and special keywords of **Providers**. They *cannot see* their hourly compensation. Also, they *cannot place* a bid for the **Provider**.
10. Signed-up Agents (**Providers** and **Clients**) *can see* every piece of information available on **the system**.
11. **Providers** *can choose* between Basic and Premium plans.
    Premium subscribers will *appear* first on the search list, regardless of their approval ratings or hourly compensation.
12. The sorting algorithm always *puts* Premium **Providers** on top, then verified **Providers**, and then the rest. Between each group, **Providers** should be *sorted* based on their approval ratings by default (can be changed).
13. A **Client** is able to *change* the sorting of results upon searching a keyword (e.g. **Clients'** approvals, number of projects done, the amount of hourly compensation).
14. A **Client** *can request* a **Provider** and *place* a bid. The bid *can be a different* value than the hourly compensation of the **Provider**.
15. **Provider** *can accept* or *reject* a bid.
16. A *rejection* from the **Provider** will be directly *sent* to the **Client**.
17. *Accepting* a request from the **Provider** will go through **the System** first, and not directly to the **Client**.
18. **The System**, upon *receiving* an accept confirmation, will *pull up* a contract and *send* it to both **Provider** and **Client**.
19. **Provider** and **Client** *can accept* or *reject* the contract.
20. Any *money transfer* will be *handled* by **the System**.
    The System will *receive* 30% of any transaction. This info should *be* in the contract.
21. Ability to *watch* the progress of the project for the both sides (**Provider** and **Client**).
22. The tracking page will *show* the tentative deadline, progress so far, and estimated time of completion based on the current pace.
23. A chat room page will *be created* for **Client** and **Provider** once a project *gets accepted*.
24. Any change request from the **Client** must first *get accepted* by the **Provider** after a project begins.
    Deadlines could *change* based on **Provider's** request
25. When a project *is done*, the **Client** *can leave* a comment and rating for the **Provider**,
26. A **Provider** *can also leave* a comment and rating for the **Client**. **Provider**s *can see* the past ratings of a **Client** when there is a new bid.
27. The app **must** have a GUI.

2-5. Wish List (Not implemented)

It seems we can implement all requirements, except the number 21 till 26. Basically, it means we assume all parties are satisfied from each other and there is no need for monitoring or rating. Therefore, we hope to be able to implement items 1 to 20 and also number 27 that requests a mandatory GUI. And we put items 21 to 26 in our wish list.