Group 18

Time Banking System

SRS

5/1/18

Xin Tang, Vincent Zhang, Joseph Urie,Cletus Andoh

**Table of Contents**

1.0 Introduction………………………………………………………………………………..4

1.1 Goals and Objectives…………………………………………………………...……….....4

1.2 Scope of Solution………………………………………………………………..………...4

2.0 Scope of Solution………………………………………………………………………….5

2.1 Product Context…………………………………………………………………....5

2.2 Functional Requirements…………………………………………………………..5

2.2.1 Expected of the System………………………………………………….5

2.2.2 Not Expected of the System……………………………………………..5

2.3 Non-Functional Requirements…………………………………………………….5

2.3.1 Expected of the System……………………………………………….....5

2.3.2 Not Expected of the System……………………………………………..6

3.0 Detailed Requirements…………………………………………………………….7

3.1 Use Case Diagram…………………………………………………………………7

3.2 Communication Diagram.........………………………………………………......7

4.0 Quality Assurance………………………………………………………………………..8

**1.0 Introduction**

The Time Bank System is a mobile based application where people on this platform can trade minutes as local currency. This system allows people to request various services and also to provide various services to people on this platform. When a user needs to request a service, the request can be seen by all users of the application when they are searching for services in which they can perform. The request includes how many minutes the requester is willing to trade for the service. Upon creating a new account, each user will be given thirty minutes for free to get started.

**The major roles of this system are:**

• The requester who will need a service

• The responder who is available to provide service to the requester

**The major use cases of the system are:**

• User signs up with name, location, description

• Requester posts a service

• Responder browses through all available services request

• Responder views approximate time to a service location

• Requester rates the service provided by a responder

**Example Usage:**

1. A requester creates a post:

* Post Title: “Need Tire Changed”
* Post Location: [Insert Location]
* Post Description: “I need my car tire changed and will pay 30 mins to someone who helps with this”
* Post Amount: 30 minutes

1. System:

* Stores info in database

1. A responder searches for job:

* Searches for a post
* Responder finds post and click button to view description

1. System:

* Get Responder location and Requester’s job location and calculate distance

1. Responder selects the job
2. System sends notification to Requester
3. Responder goes to job location(Location services must be turned on). Responder can press start to start the timer.
4. Once 30 minutes has passed, system updates both the number of services and the hours for the requester and the responder.
5. Requester of the service can leave a rating for the responder.

**2.0 Overall Description**

The main user of the system will be the requester who request the jobs and receive the person that takes the job. Requester has the option to assign the job and remove it. The receiver can take the job. Once the job is completed, the job listing will be remove. Once the receiver finishes a job, the requester pays the receiver with time. After job is completed bothe the requester and receiver can rate each other. The rating will be saved in our firebase database and average all attempts.

**2.1. Product Context**

The system interacts with firebase database

**2.2 Functional Requirements**

**2.2.1 Expected of the system**

1. Job management

1.1 Requester can add and remove jobs

1.2 Receiver can take jobs

1.3 Requester pays receiver

1.4 Both receiver and requester can rate each other

2. Payment

2.1 Requester pays the receiver

2.2 Receiver takes payment from requester

3. Rating system

3.1 Receiver rate Requester

3.2 Requester rates Reciever

4. Search for and accept jobs

4.1 Receiver can search jobs in the search job above

5. Post a new job

5.1 Requester can post a new job

6. Ability to create an account/profile to store information about user

6.1 users are able edit their names,email in their profile

6.2 user are to create a account and a password

**2.2.2 Not Expected of the system**

· Receiver and Requester cannot comment on each other profile

· Requester can’t get their money bad if job is done bad

**2.3 Non-Functional Requirements**

**2.3.1 Expected of the System**

1. **Operating requirement**

1.1 The system should do a real-time update any time.

**2. Performance Requirement**

2.1 The system should be able to return the result of operation for 5 seconds or

2.2 The system should be able to update the database

**2.3.2 Not Expected of the System**

* Allow users to sign up for jobs they can’t afford

**2.4. Operating Environment**

* Android

**2.6. Assumptions and Dependencies**

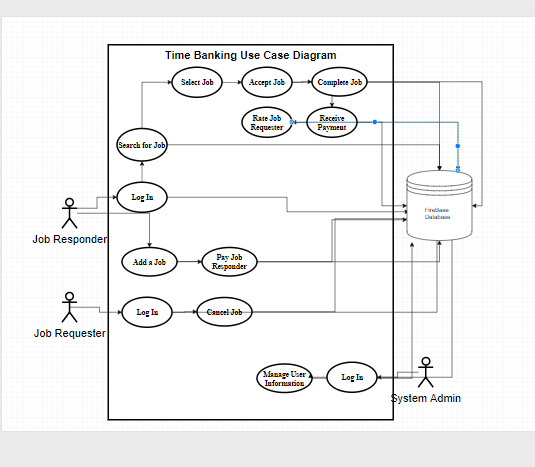
· Users should be able to fully interact with computer system and interface

· Users should be to view job list

**3.0. Detailed Requirements**

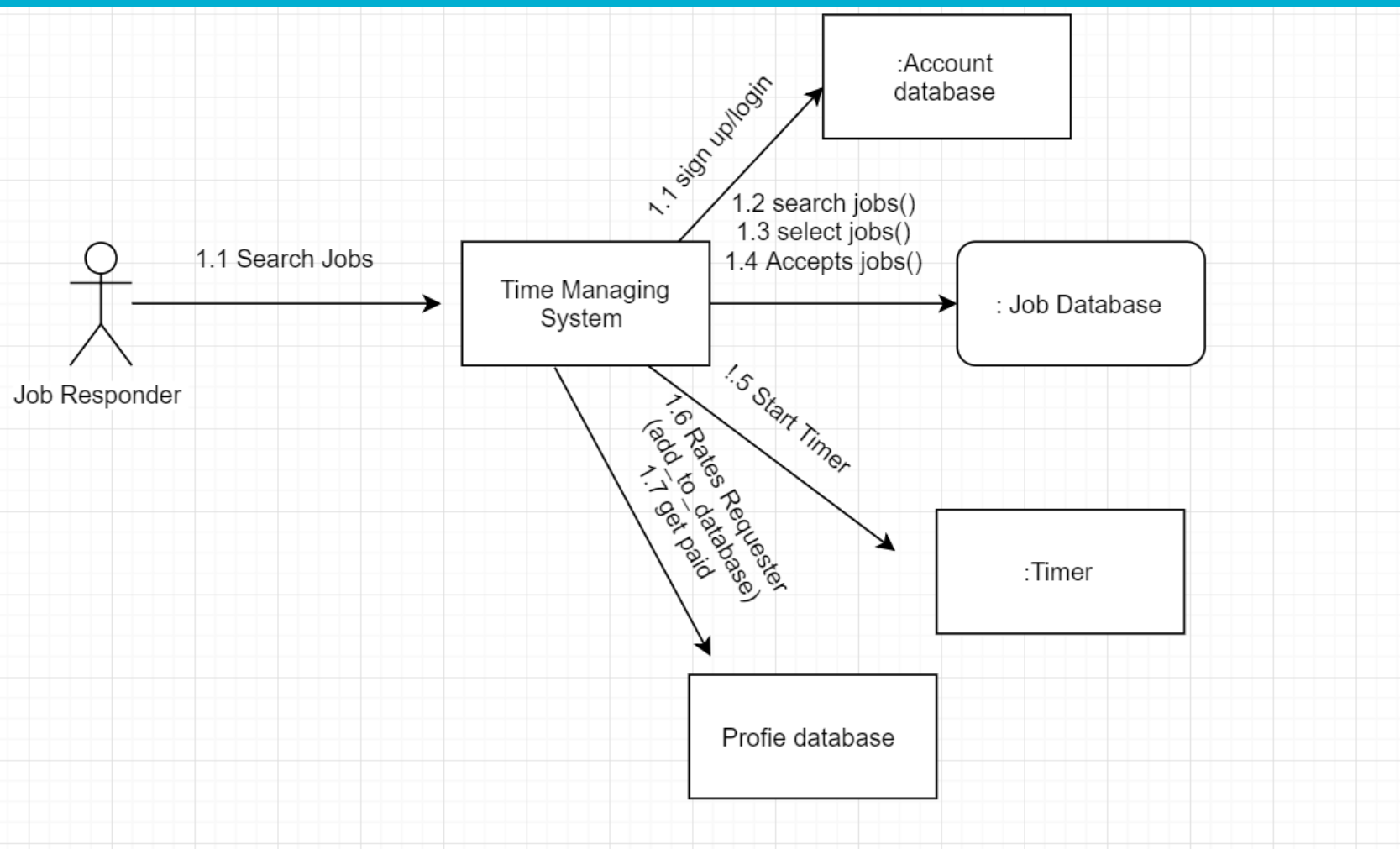
The software gives user an easy and productive way to post and accept jobs. It gives the user an easy and painless process to sign themselves in and immediately post and accepts jobs. The system further provides a place to rate each person. The best part is that the software will be able to run on android platform,the world most convenient mobile system.

**3.1. Use Case Diagram**

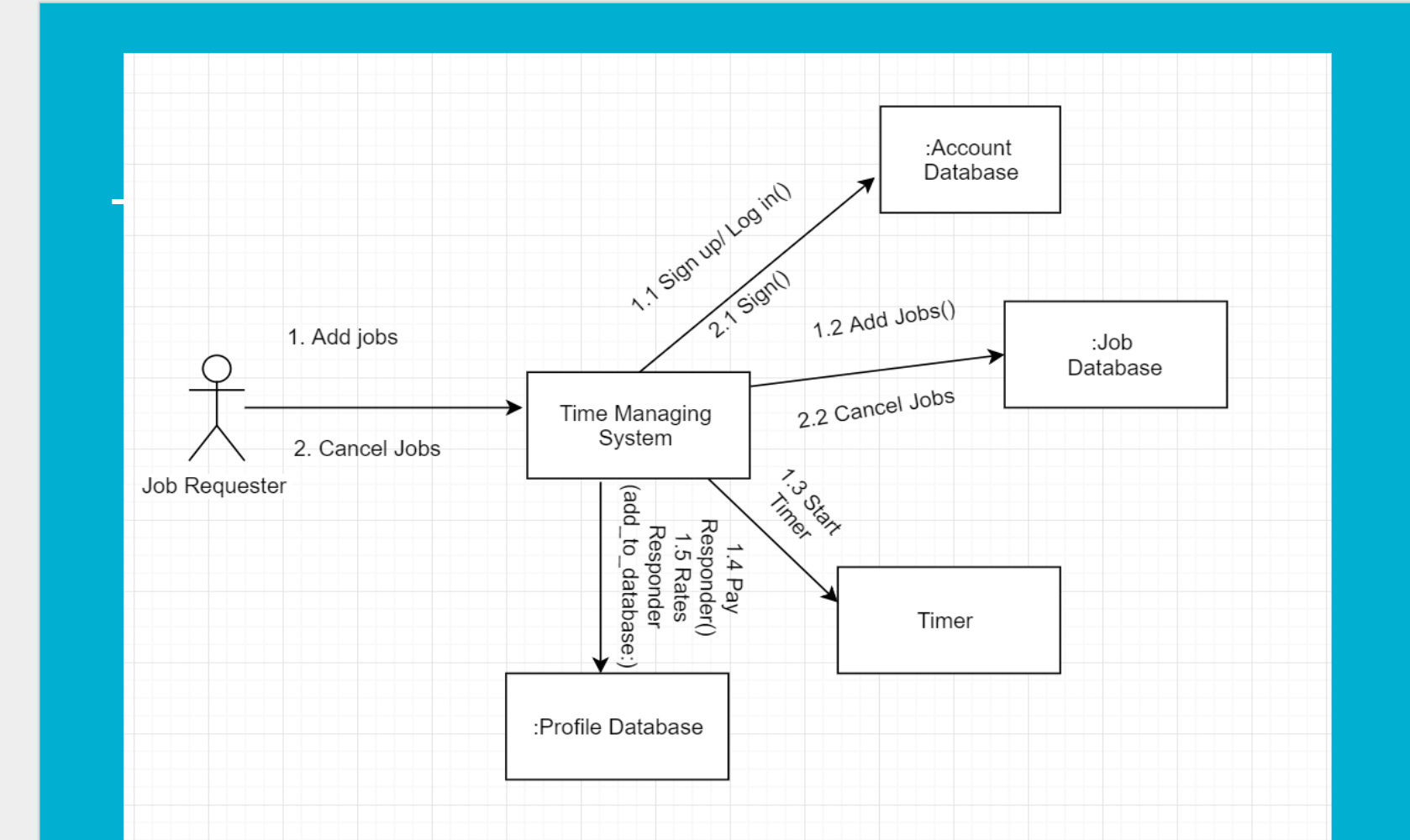


**3.2 Communication Diagram**

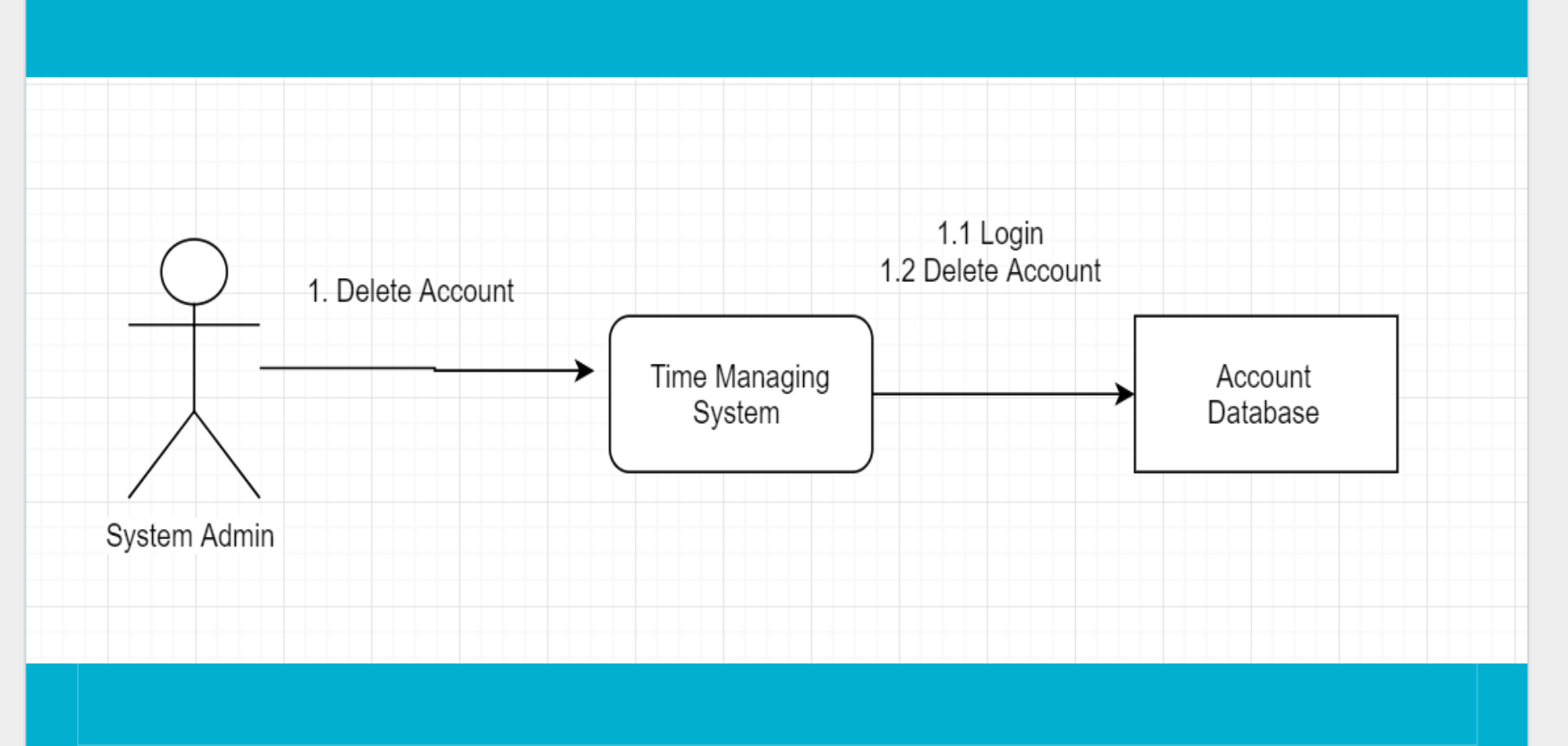
**Job Responder**

****

**Job Requester**



**System Admin**



**4. Quality Assurance**

To test the system, a beta version of the system will be sent out to volunteer prospects and they will send their feedback and report any bugs or crashes to me and the system administration, in which they’ll ensure that all the major components of the system are working properly. Prospective volunteers and managers will test the smoothness and convenience of the inventory managing application, the sign-in/out application, and the scheduling application. To do this, we will:

1. Try to login to the system with correct ID and see if it works successfully.
2. Let 3000 people use the system at the same time, check if the server could sustain the large amount of user at once
3. Test if several responder responds to a job almost at the same time, the job will go to the first user
4. After job is completed, see the change of system of several different individuals.

5. Make sure both the Responder and requester can rate each other’s profiles