**CS3354 Software Engineering**

**Final Project Deliverable 2**

Nibiru

Justin Holloway

Jin Chen

Duy Truong

Kristofer Sanchez

Paul Thang

Hongyun Du

Tze Yang Chen

Table of Contents

[Final Project Description 3](#_Toc6576339)

[Title 3](#_Toc6576340)

[Group Members 3](#_Toc6576341)

[Purpose 3](#_Toc6576342)

[Description 4](#_Toc6576343)

[Tasks (Each member assigned at least 3 different tasks) 4](#_Toc6576344)

[Previous Draft and Feedback 6](#_Toc6576345)

[GitHub Repository 6](#_Toc6576346)

[Delegation of Tasks 7](#_Toc6576347)

[Project Deliverable Document 7](#_Toc6576348)

[GitHub Creation/Initial Commits 7](#_Toc6576349)

[Software Process Model 7](#_Toc6576350)

[Software Functional & Non-Functional Requirements 7](#_Toc6576351)

[Use Case Diagrams 7](#_Toc6576352)

[Sequence Diagram 7](#_Toc6576353)

[Class Diagram 7](#_Toc6576354)

[Architectural Design 7](#_Toc6576355)

[Software Process Model 7](#_Toc6576356)

[Software Functional and Non-Functional Requirements 7](#_Toc6576357)

[Use Case Diagram 9](#_Toc6576358)

[Sequence Diagram 10](#_Toc6576359)

[Creating an Account 10](#_Toc6576360)

[Message Other User 10](#_Toc6576361)

[Search Other Accounts for Items 11](#_Toc6576362)

[View Other Users 12](#_Toc6576363)

[Accept Transaction Offer 13](#_Toc6576364)

[Customize Profile 14](#_Toc6576365)

[Make an Offer On an Item 15](#_Toc6576366)

[Rate a Transaction 16](#_Toc6576367)

[Class Diagram 17](#_Toc6576368)

[Architectural Design 18](#_Toc6576369)

[Project Scheduling 18](#_Toc6576370)

[Cost, Effort and Pricing Estimation 19](#_Toc6576371)

[Estimated cost of hardware products (such as servers, etc.) 21](#_Toc6576372)

[Estimated cost of software products (such as licensed software, etc.) 21](#_Toc6576373)

[Estimated cost of personnel (number of people to code the end product, training cost after installation) 21](#_Toc6576374)

[Software Testing Plan for Nibiru 22](#_Toc6576375)

[Unit Testing 22](#_Toc6576376)

[Validation Testing 22](#_Toc6576377)

[Testing Input / Output 23](#_Toc6576378)

[Testing Encryption/Decryption 23](#_Toc6576379)

[Testing Object (UserNode) 23](#_Toc6576380)

[Results and Analysis 23](#_Toc6576381)

# Final Project Description

### Title

**Nibiru:** Musician’s equipment community and exchange application

### Group Members

Justin Holloway

Jin Chen

Duy Truong

Kristofer Sanchez

Paul Thang

Hongyun Du

Tze Yang Chen

### Purpose

This project will serve the purpose of designing a PC application intended for musicians looking to share and exchange equipment with users of similar genres, specs, experience, etc. The main feature of the software will be the ability to create a profile containing information about current equipment within a users setup. With this data, other user’s will be able to view this profile and view the instruments and other components that may be of interest to them. This is different from other services that are currently available. Our goal is to combined multiple available services into one friendly location/application. Service such as eBay and Craigslist provide online personal retail, but what they do not do is ensure authentic equipment is being sold, it is essentially an unverified open market which is great but not for a customer who is looking for a specific sound or functionality from an instrument. Additionally, the profile owner may post demos of their work using the specified equipment they have listed, this function is not currently available on any online sales platform. From there, they can make the viewing users an offer on the equipment if they are interested in purchasing it. If no sale is offered, then this stage can be replaced by showing purchase options for the equipment at local retailers instead. Another component of the application will be the ability for businesses to post shows that they will be hosting. Local users will be able to see this posting and offer to perform in that event. An equipment rental feature will also be included for both business accounts and users.

### Description

Our motivation is to change the relationship between the current instrument market and people. The central theme of most social software on the market today is the interaction between people. Our hope is to create a product that can use personal items to be the theme, thus driving the communication between those interested in this product. It is suitable for People with specific needs, or a group with a particular hobby for an item. For example, musicians have a hobby for musical instruments. Where to get more intuitive and professional information is crucial and not currently available all in one location. Traditional musical instrument sales are a purely commercial transaction between the buyer and the seller. These transactions usually take place in a brick and mortar store which is great, but not for those who do not have access to these stores. The goal of this software is to change this structure, joining the interaction between the buyer and the seller. This application will let the transaction itself become an interaction with the music and discuss the topic of the instrument itself.

Every day, hundreds of music albums are released globally, people are going to buy those albums and enjoy the music. We will add a feature to make a connection between the music and the instruments the musician used in making the music. The software will allow professional-level buyers to have a more informative sharing and shopping experience. For example, a musician participated in the production of a specific song. The musician then posted the instrument and connected it to the music, those who have heard the music and were impressed by the sound of the device/instrument are more likely to be a potential purchaser. This kind of shopping information will have a more accurate match value than the traditional text description. The software allows sellers to sell their instruments while also allowing people to pay attention to the music. For buyers, buying a guitar with a music album that improves their sound is also a new experience.

### Tasks (Each member assigned at least 3 different tasks)

1. Function Management

Tze-Yang Chen and Hongyun Du

1. Organization and workflow

Justin Holloway

1. Proof reading

Justin Holloway

1. UI / UX

Tze-Yang Chen and Jin Chen

1. Professor / TA communication

Kristofer Sanchez and Paul Thang

1. (Optional) implementation

All members

1. Presentation construction

All Members

1. Profits and cost

Justin Holloway, Jin Chen, Duy Truong, and Paul Thang

1. User communication

Hongyun Du

1. Logistics

Justin Holloway and Jin Chen

1. Functional / non-functional requirements

Kristofer Sanchez and Hongyun Du

1. Technical details

Duy Truong

1. File management (providing demo video files, demo mp3 files, etc. and backup files in cloud driver)

Tze-Yang Chen, Jin Chen, and Paul Thang

=============================Below is deliverable 2==============================

1. Project Scheduling <include MS project>

Tze-Yang Chen Hongyun Du

1. Cost, Effort and Pricing Estimation <include hardware, software, and personnel>

Duy Truong, and Paul Thang

1. Test plan

Justin Holloway

1. Comparison of similar designs

Jin Chen

1. Implementation of the code

Hongyun Du

1. Migration of old file and all documenting

Kristofer Sanchez and Tze-Yang Chen

1. PowerPoint and conclusion

All members

============================Above is deliverable 2===============================

### Previous Draft and Feedback

Figure 1:Feedback of Proposal page 2

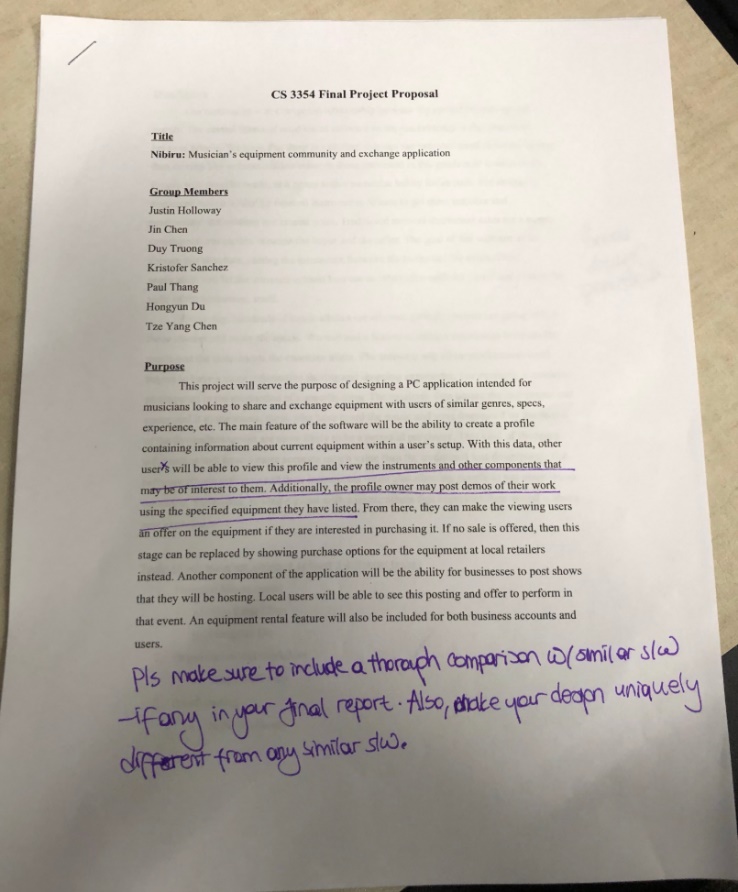


Figure 2:Feedback of Proposal page1

We address the feedback left by searching and analyzing what is currently available on the market. Currently we have sales services provided by eBay and Craigslist, and the likes. What these services do not provide is the ability to get first hand experience sales representatives, or access to subject matter experts. While they serve as a platform to sale music equipment they do not give an experience tailored to musicians. Nor is there the ability to find equipment you want based on sound, I can not go to any of these services and search for equipment based on a specific sound I am looking for in my music. The second aspect of our service is to provide opportunity for musicians and venues to find talent. This service is provided by MusicianCasting.com and can be done by websites like Monster and Indeed, but our goal is to bring the social aspect and sales aspect together in one location so there is no need to use multiple services.

# GitHub Repository

The following link is to the GitHub repository for Nibiru

https://github.com/KSanchez89/3354-Nibiru

# Delegation of Tasks

## Project Deliverable Document

Kristofer Sanchez

## GitHub Creation/Initial Commits

Kristofer Sanchez and Justin Holloway

## Software Process Model

Tze Yang Chen

## Software Functional & Non-Functional Requirements

Completed as team

## Use Case Diagrams

Justin Holloway and Paul Thang

## Sequence Diagram

Hongyun Du and Jin Chen

## Class Diagram

Duy Truong

## Architectural Design

Tze Yang Chen

# Software Process Model

The software process model we will use for Nibiru is an Evolutionary Model more specifically Prototyping. The reasons are as follows. First, we are making a complex software. To implement different functions, our team often needs to spend time together to discuss and research whether these functions can meet the needs of users. Secondly, our time is limited, we need to design and build a software framework in a short period. We need to quickly categorize the functionality of the software through rapid design. In addition, we need to gradually study and discuss specific software features, as well as to maximize the functionality of the software and the needs of users who actually use it. Finally, in this design environment, we need to use a lot of iterative processes in the development and design process to gradually implement more complex functions on the basic functions.

# Software Functional and Non-Functional Requirements

The following table lists the functional and non-functional requirements for Nibiru.

|  |  |
| --- | --- |
| **Non-Functional** | **Functional** |
| Performance Requirements:  Uploading speed of at least 1Mbps | Users must be able to direct message with other users. |
| Space requirements:  Website will have a backup of user accounts and transactions. | User may post comments to sellers product pages. |
| Accounting requirements:  Website will not retain/store user financial account information e.g. credit card numbers and bank information. | Seller may upload product pictures and descriptions. |
| Safety/security requirements:  User account passwords will be required to be 8 to 23 alphanumeric characters, as well as have one upper case, lower case, and symbol. | Software will give the seller the option to select music genre they belong to. |
| Usability requirements:  When an offer is accepted by a seller the buyer must fulfill payment for the product within 3 business days of offer acceptance. | Seller may upload a music demo of 10 second for users to listen to. |
| Environmental requirements:  We will only provide paperless billing statements as our green initiative. | Buyers can make an offer to the seller. |
| Operational requirements:  Each user must have an account to make transactions. Users without an account will be able to view a demonstration of provided services. | The website will schedule/track/update shipping information. |
| Development requirements:  Service prototype will be completed in two months. |  |
| Legislative requirements:  Content will be in compliant with all Recording Industry Association of America standards, including parental advisory warnings for explicit lyrics. |  |
| Ethical requirements:  Users can not post fake deals or imitation products, for high value products sellers must provide certificate of authentication. |  |
| Regulatory requirements:  Services will be provided over Transmission Control Protocol (TCP) as well as HTTPS |  |
| Efficiency requirements:  Service will ensure reliable data transfer (RDT) via TCP. |  |

# Use Case Diagram

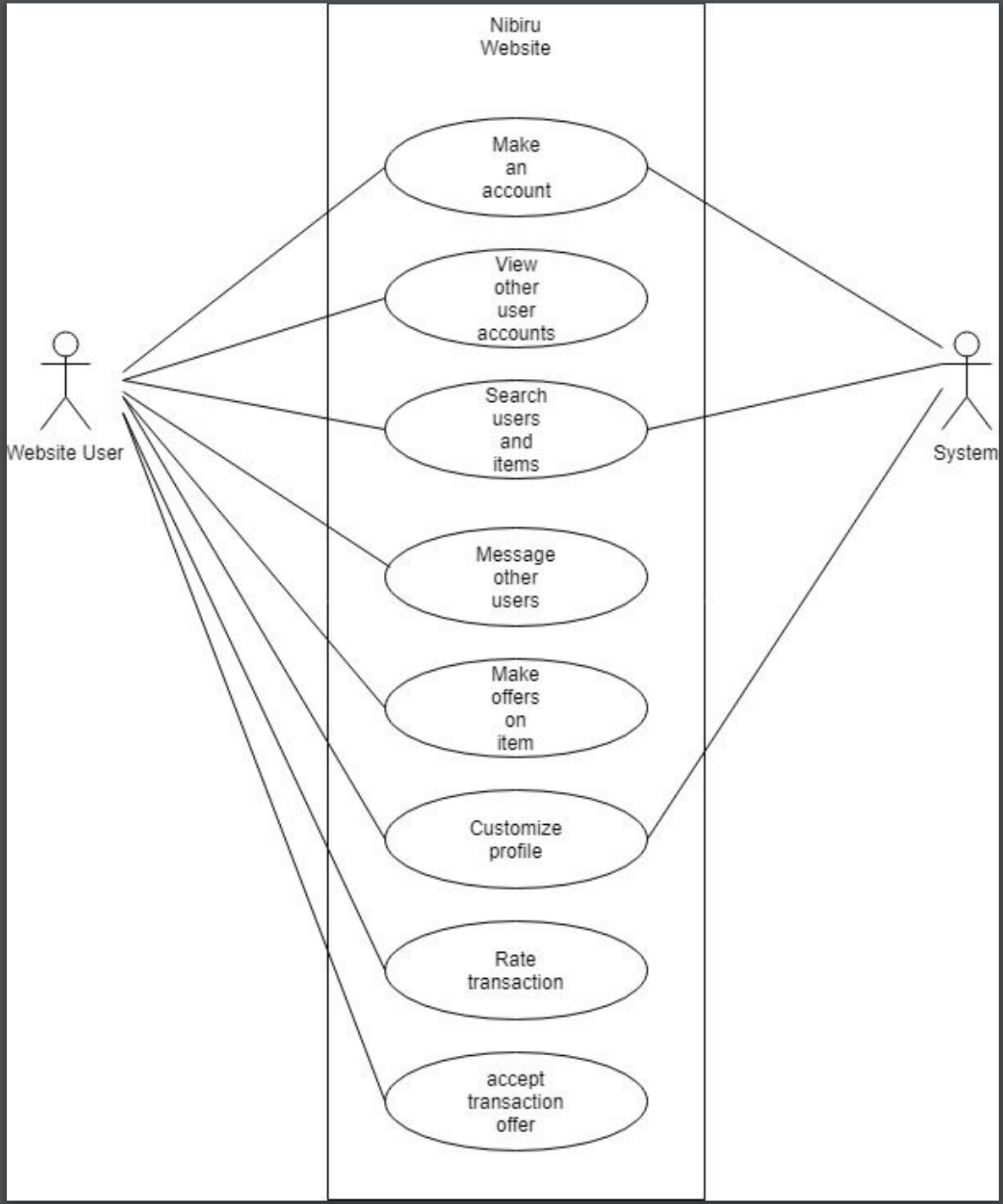


Figure 3:Use Case Diagram

# Sequence Diagram

## Creating an Account

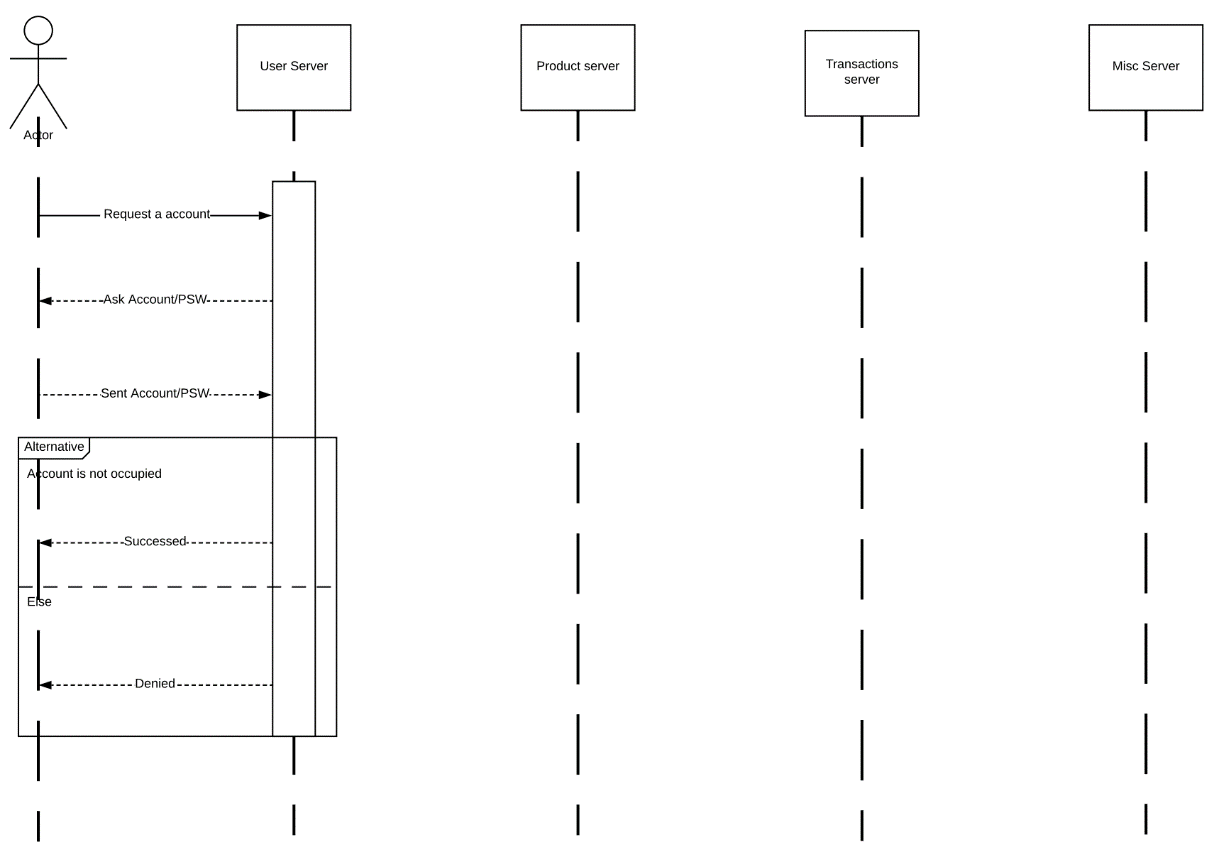


Figure 4:Seq.Diagram of Creating an Account

## Message Other User

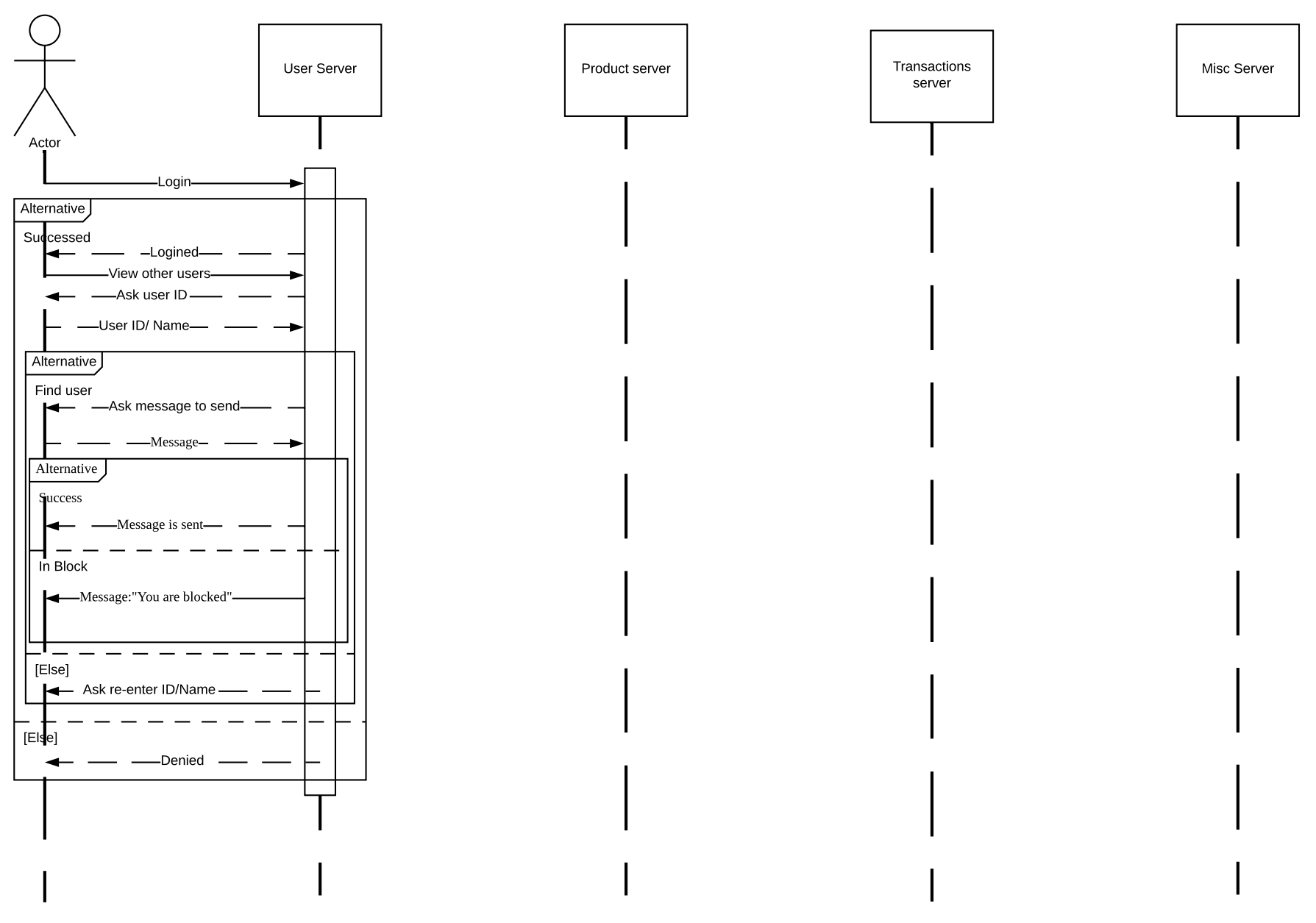


Figure 5:Seq. Diagram of Message Other User

## Search Other Accounts for Items

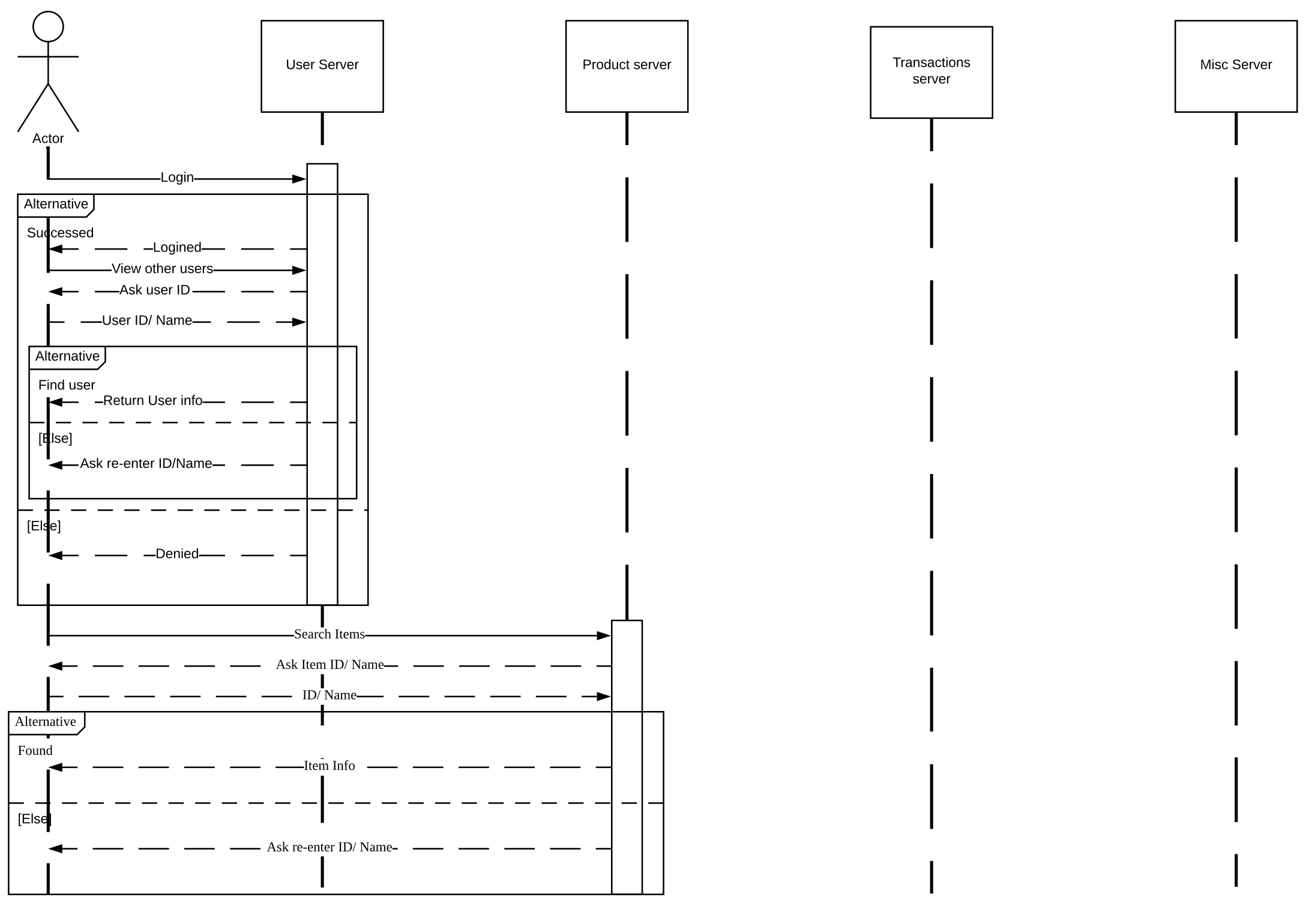


Figure 6: Seq. Diagram of Search Other Accounts for Items

## View Other Users

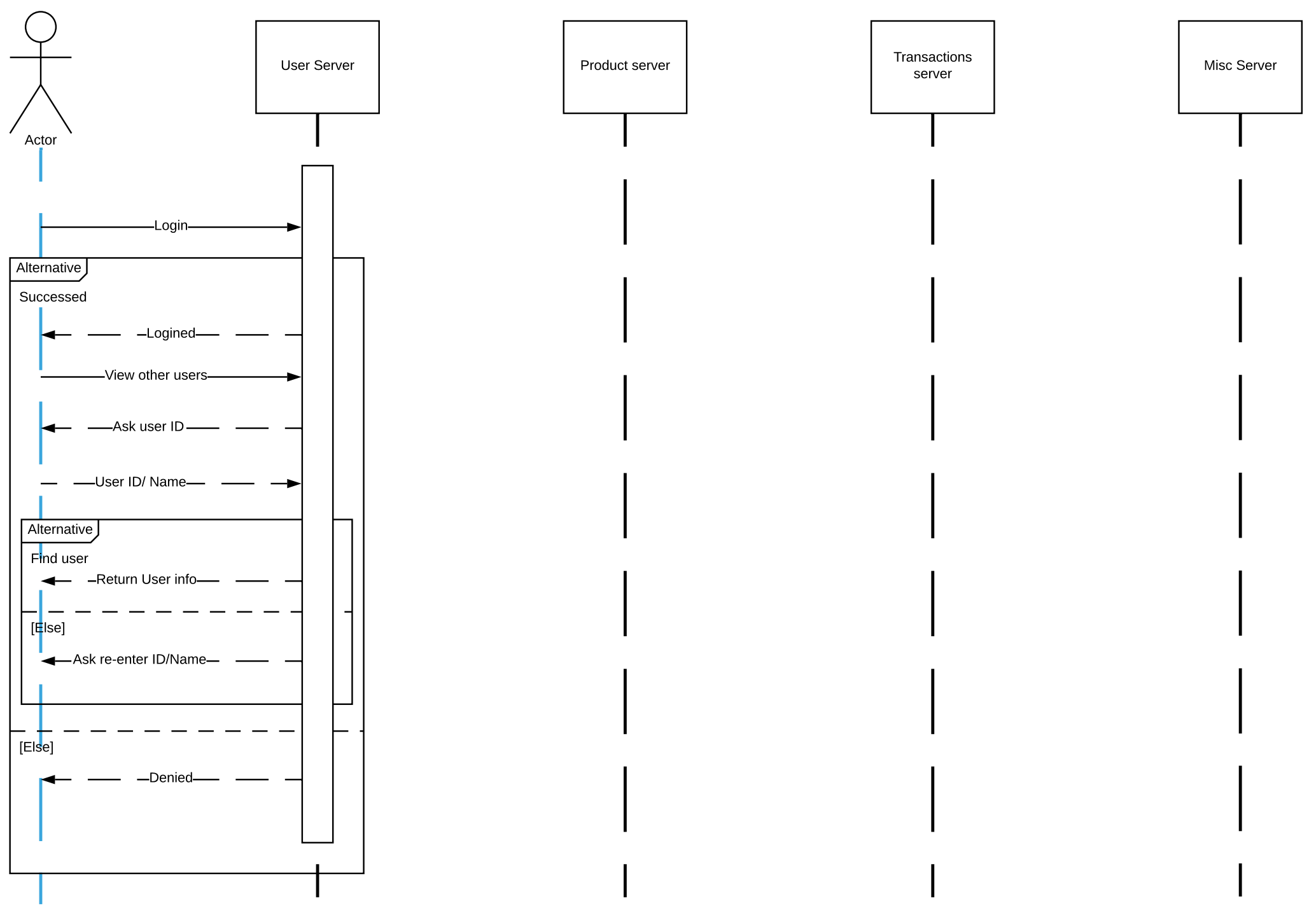


Figure 7: Seq. Diagram of View Other Users

## Accept Transaction Offer

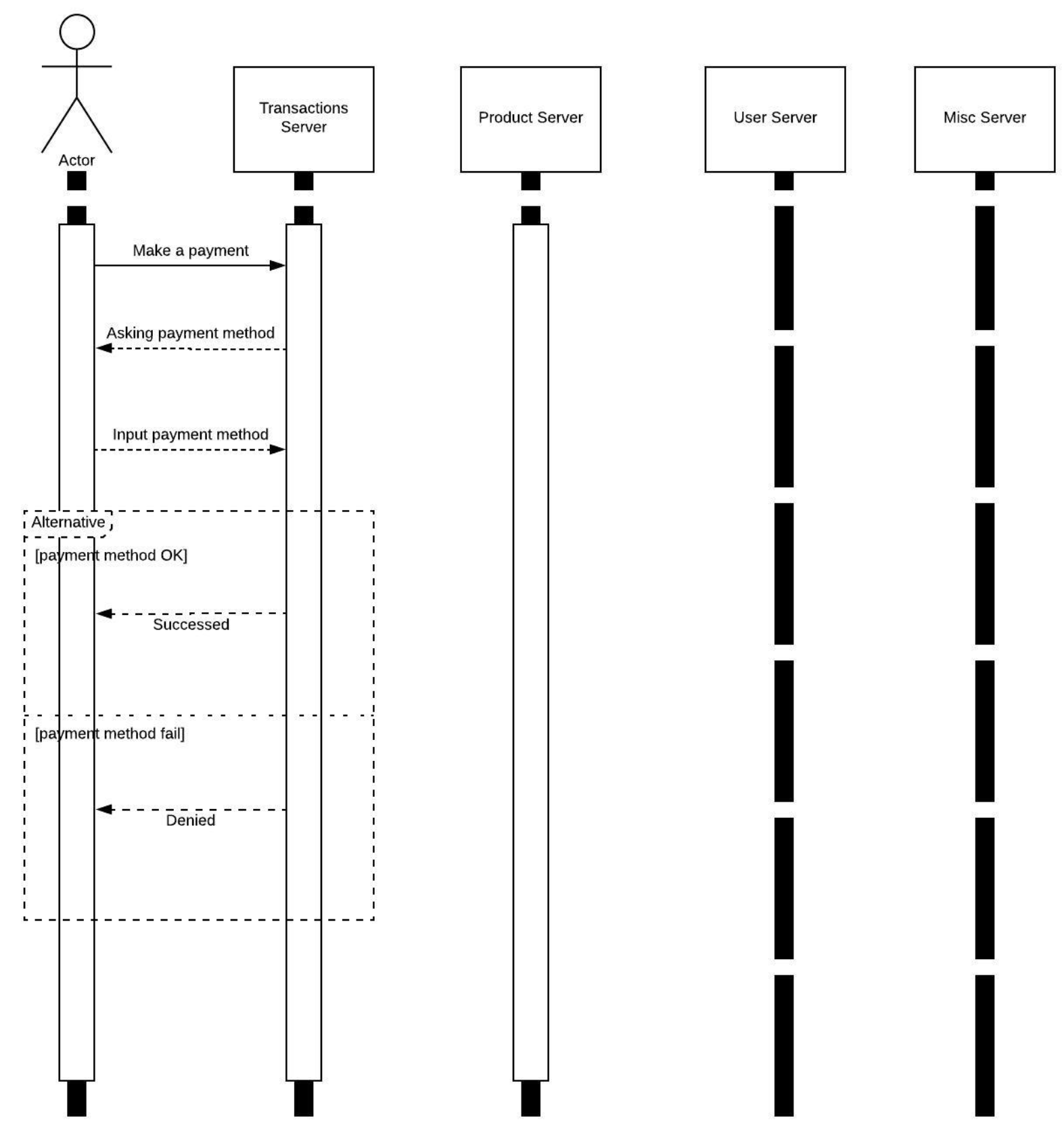


Figure 8: Accept Transaction Offer

## Customize Profile

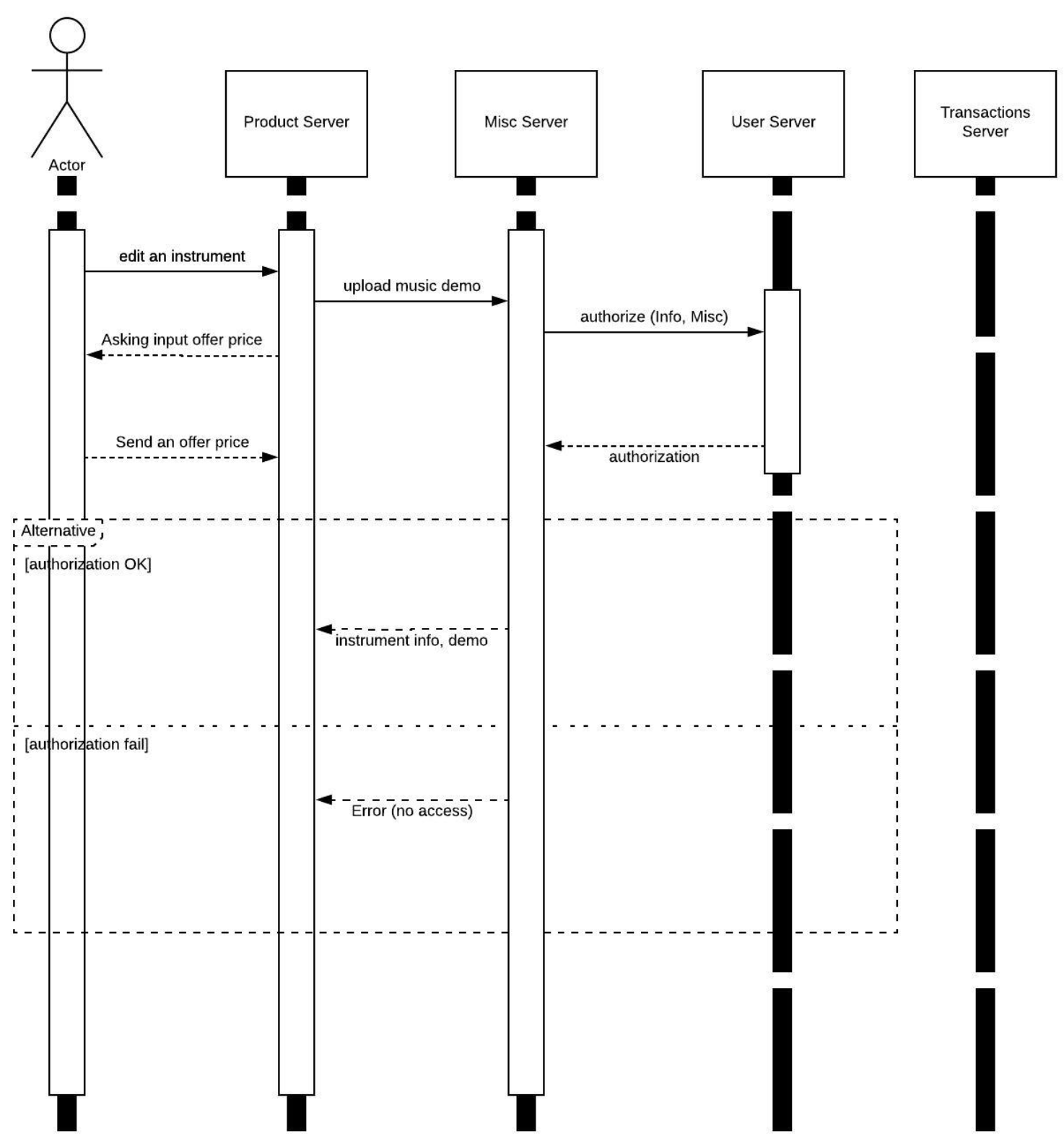


Figure 9: Customize Profile

## Make an Offer On an Item

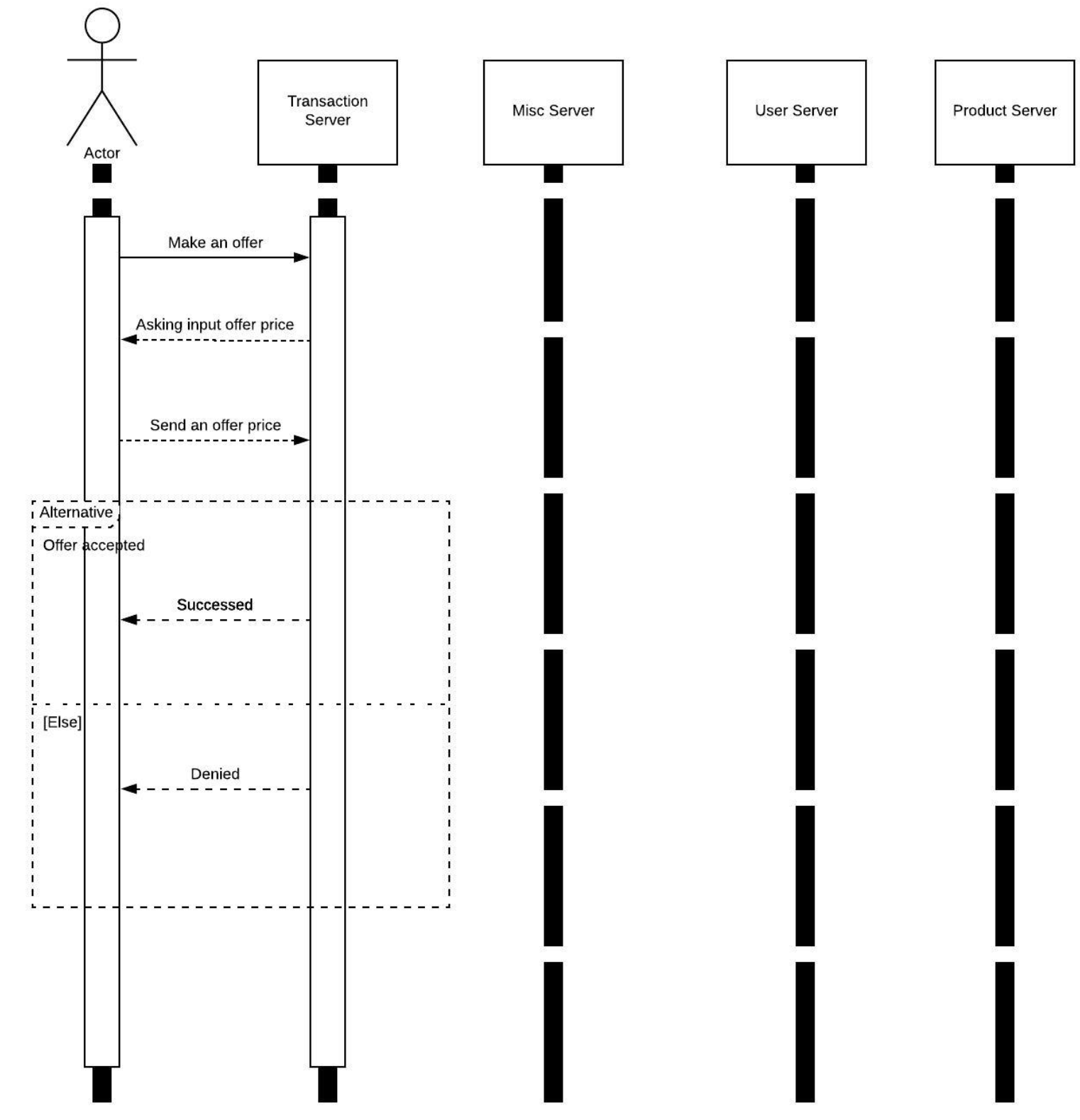


Figure 10: Make an Offer On an Item

## Rate a Transaction

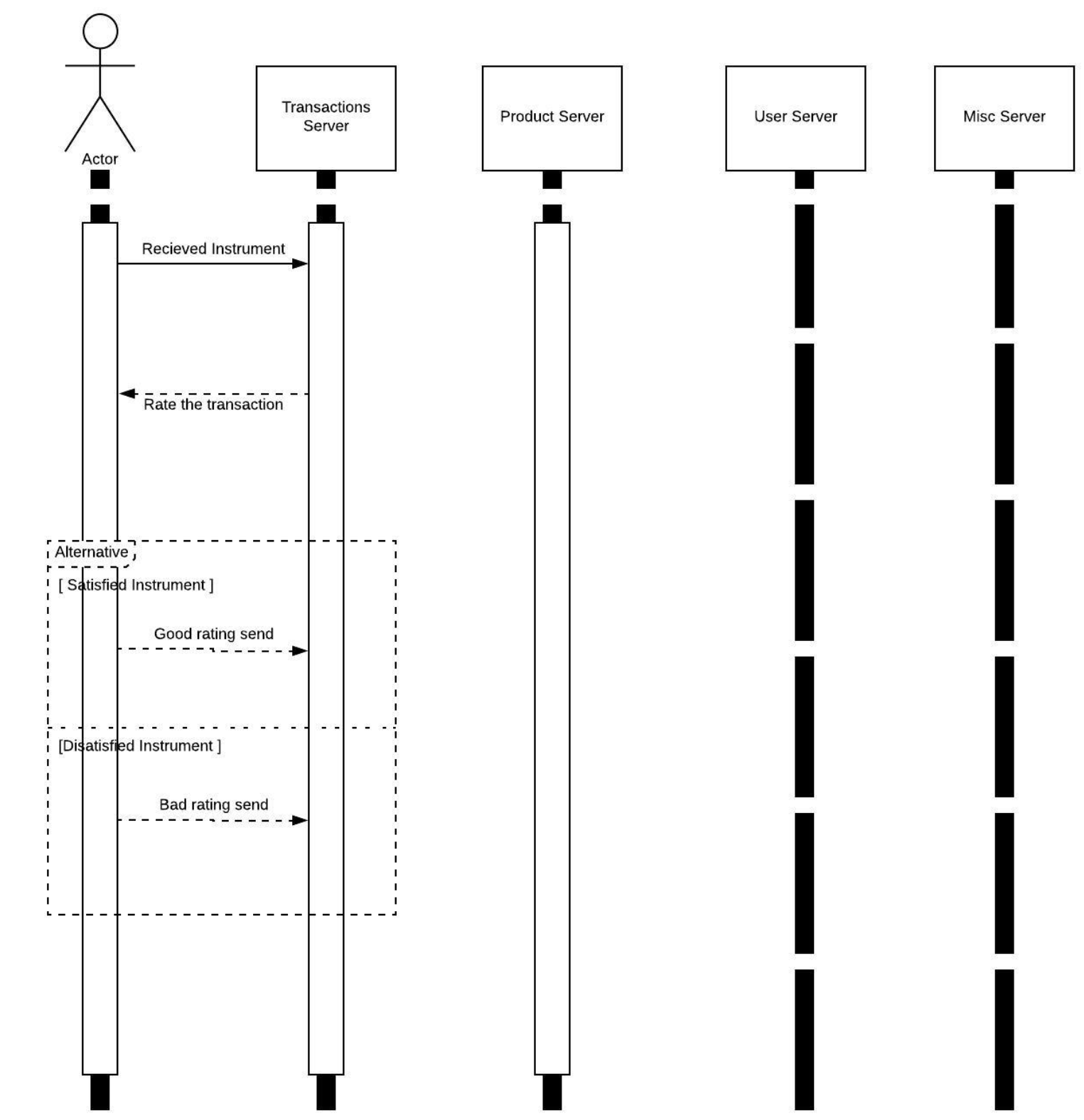


Figure 11: Rate a Transaction

# Class Diagram

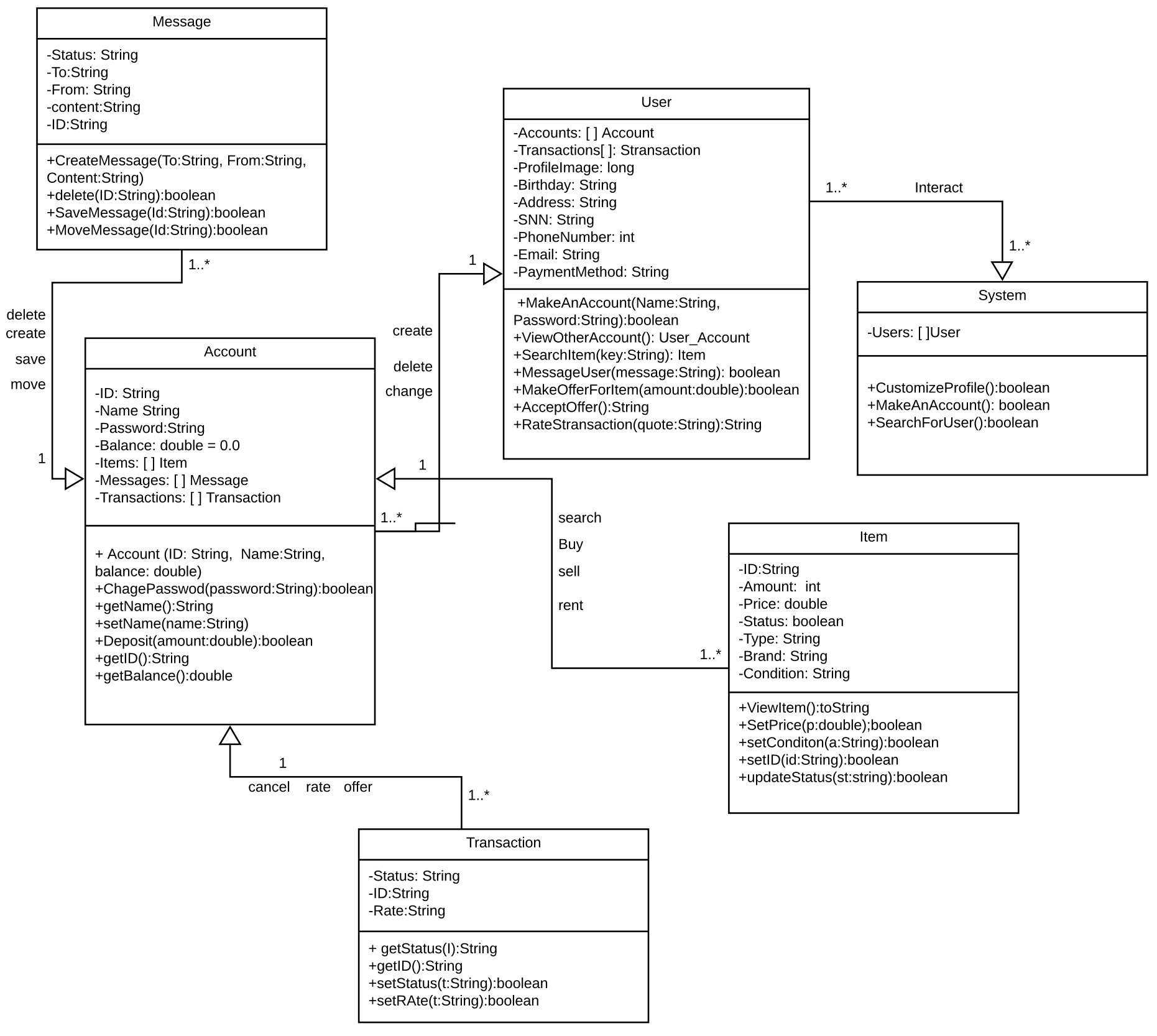


Figure 12: Class Diagram

# Architectural Design

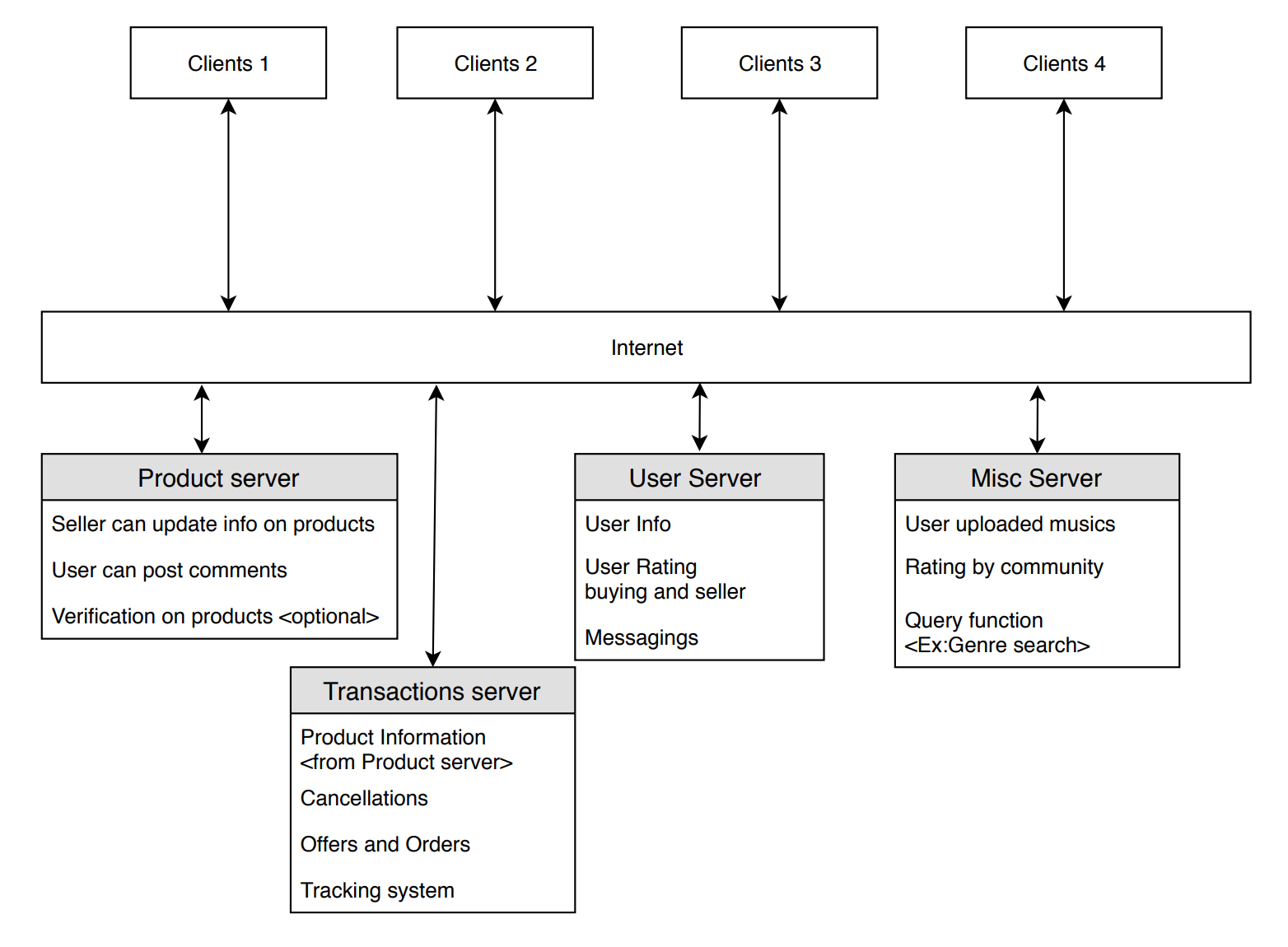


Figure 13: Architectural Design

=============================Below is deliverable 2==============================

# Project Scheduling

According to the cost estimation, we need 10 people, and 20 months to compete, we also agreed on the task path and MS Project results. Below is the screenshot of MS project assign results and critical path results. I have used multiple website that provides salary for each resource. [3] [4] [5]

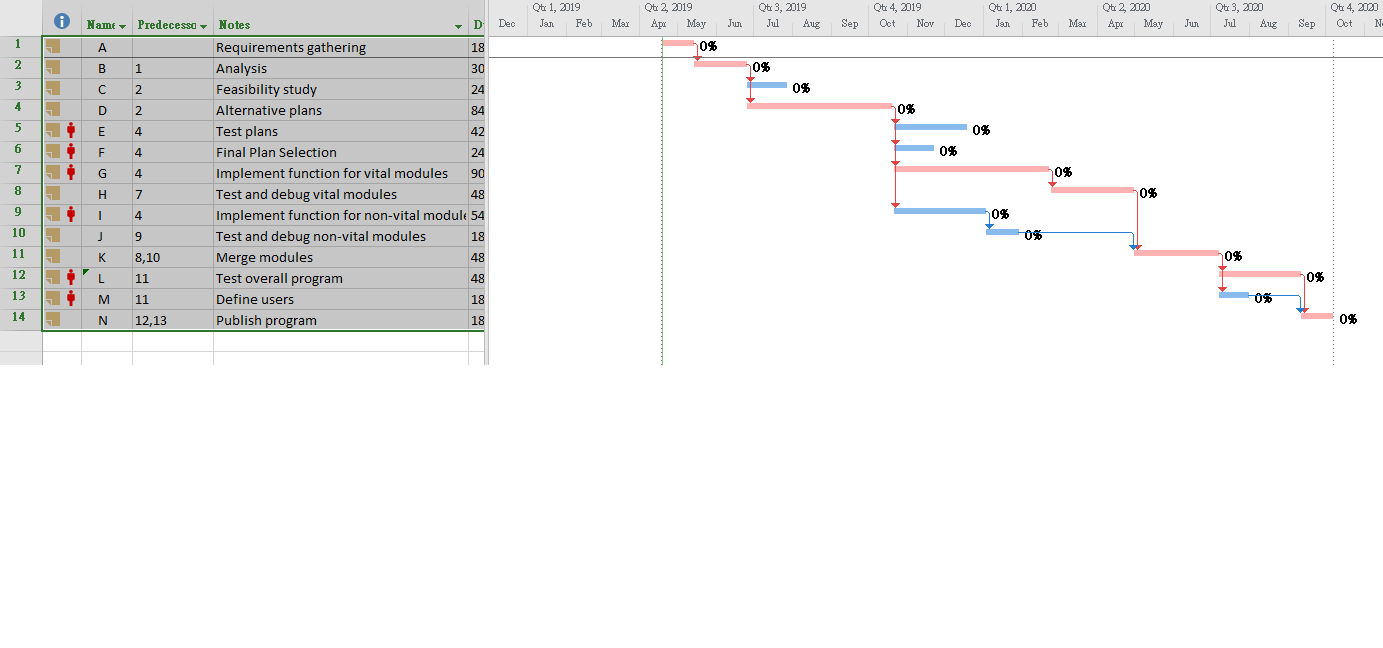


Figure 14: Critical Path

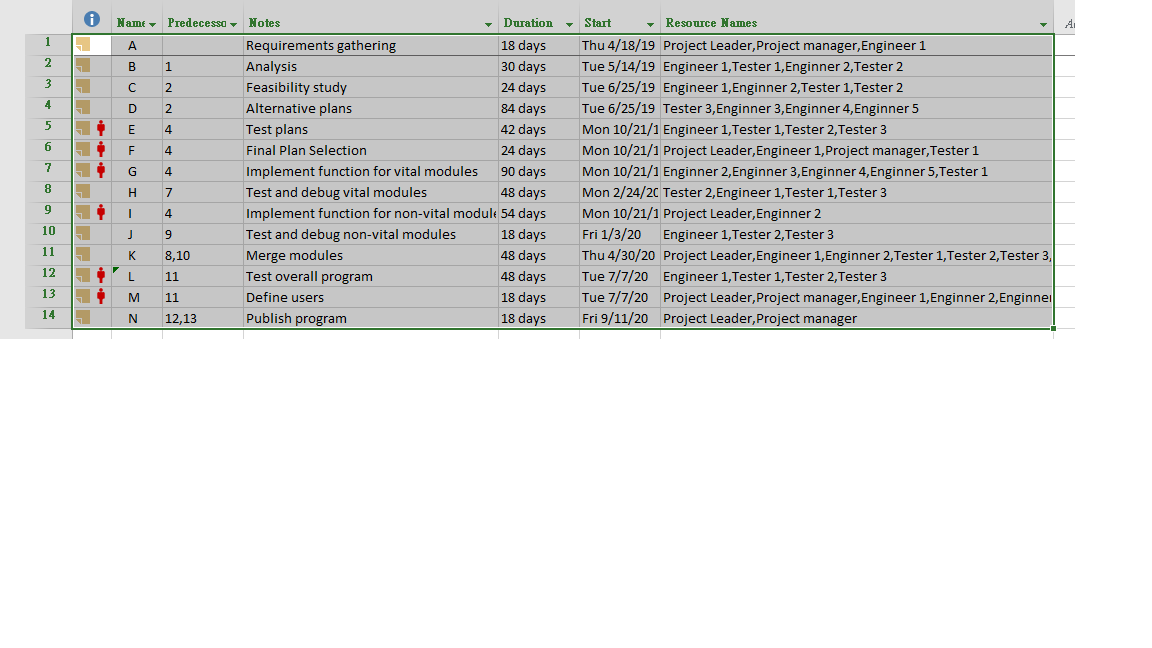


Figure 15: table of all tasks for Nibiru Project

# Cost, Effort and Pricing Estimation

On Wednesday, April 10th, our team decided to use Post-Architecture Model to implement the estimation technique to calculate the estimated cost and in turn the price for our project. We decided that the process control software will have **40 KSLOC** for the fact that we used Java as our implementation language.

Estimated Size for the project = 40 KSLOC

Required Software Reliability = 1.26 very high

Product Complexity = 1.34 very high

Execution Time Constraint = 1.29 very high

Programmer Capability = 0.76 very high

Platform Experience = 0.85 very high

Multisite Development = 0.86 very high

Normal cost drivers: Everything else.

**The implementation is as follows:**

Compute **the constant b** as follows:

b = \_\_\_\_\_\_\_\_?

b = 0.91 + 0.01 \* (SF 1 + · · · + SF 5)

= 0.91 + 0.01 \* (3.72 + 3.04 + 4.24 + 3.29 +4.68)

= 0.91 + 0.01 \* 18.97

b = 1.0997

Compute **the effort** as follows:

a = 2.94 (constant)

b = 1.0997

size = 40 KSLOC

PM = \_\_\_\_\_\_\_\_?

PM = a\*sizeb \*EM. (size =40)

PM = 2.94 \* (40)1.0997 \* 1.26 \* 1.34 \* 1.29 \* 0.76 \* 0.85 \* 0.86

PM = 205.555 person-month

The development time **(TDEV)** is:

b = 1.0997

c = 3.67

d = 0.28

PM = 205.555

TDEV = \_\_\_\_\_\_\_\_?

TDEV = c\*PMd+0.2\*(b-0.91)

= 3.67 \* (205.555)0.28+ 0.2\*(1.0997-0.91)

= 3.67 \* (205.555)0.28+ 0.03794

= 3.67 \* (205.555)0.31794

TDEV = 20 months

The number of developers **(N)** is:

N = \_\_\_\_\_\_\_\_?

N =

=

N = 10.28 persons

## Estimated cost of hardware products (such as servers, etc.)

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Count** | **Estimated Cost (US dollar)** |
| Server PC | 1 | 8000 |
| Engineer Laptops | 15 | 15,000 |
| Telephone | 3 | 1000 |
| Printers | 2 | 500 |
| Other components | unknown | 2000 |

## Estimated cost of software products (such as licensed software, etc.)

|  |  |  |
| --- | --- | --- |
| **Software** | **Count** | **Estimated cost (US dollar** |
| Window Server client access licenses | 2 | 2000 |
| Microsoft Exchange server | 1 | 700 |
| Microsoft SharePoint server | 1 | 900 |
| Microsoft SQL server (database service) | 1 | 1500 |

## Estimated cost of personnel (number of people to code the end product, training cost after installation)

|  |  |
| --- | --- |
| **Category** | **Estimated cost (US dollar)** |
| Recruiting expenses | 1000 -1200 |
| Basic salary | 30,000 – 50,000 |
| Benefits | 4000 - 6000 |
| Training cost | 1500 -2000 |
| Employment taxes | 5000- 8000 |

# Software Testing Plan for Nibiru

## Unit Testing

The project will utilize Top-down integration testing. The logic behind this decision is that the main control unit of the site can be used as a platform for the other components that will also need to be tested and evaluated. Testing will start with the subordinate “stubs” being in place of the other modules below the control. Over time we will replace these with the actual components of the software. For our control, we will be using the Account class as out test driver. The User, Message, Item, and Transaction modules will be taking the place of the stubs under the control. System will also be tested as a utility component. Using this unit testing strategy is ideal because as new features and modules are added to the software, more tests can be conducted as the project timespan continues. As each set of components completes their testing, other modules will take their place.

## Validation Testing

Being that the project is web based, validation testing will be focused primarily on beta testing. A test release will be made available after internal testing at the developer site. This release will only include the main features of the main site and all of its utilities. As features are added and changes are made, newer releases of the beta build will be distributed after waves of unit testing until the consumer ready release.

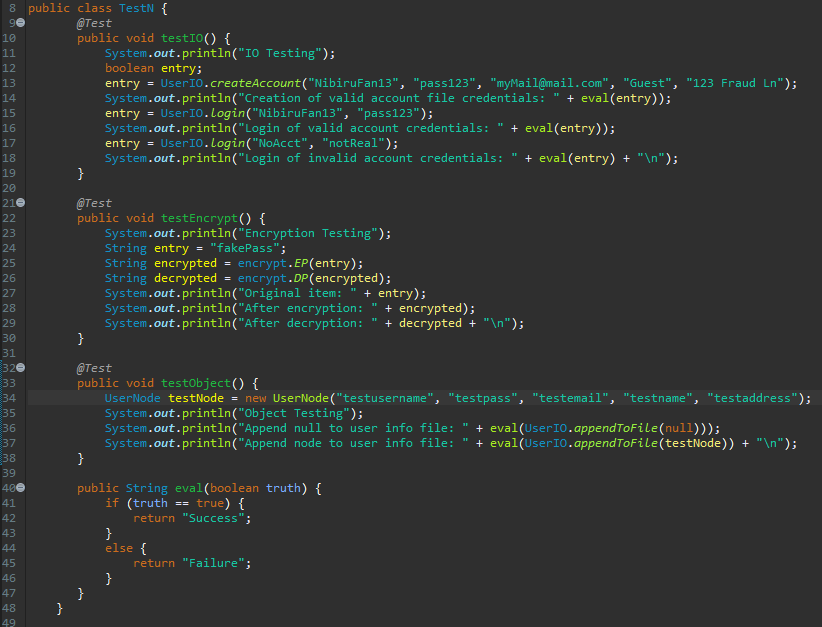
JUnit Test Cases

Figure 16: Code snapshot

## Testing Input / Output

Testing method testIO() will be used to determine if the user can make and account given all the credentials necessary to build the account. This method will also test if the user can login with both valid and invalid credentials. The outputs will display if the operations were successful. It is ideal to have the account creation be successful when there is no existing file and to be a failure when the file already exists. Valid credential login should always be successful and invalid login should always fail.

## Testing Encryption/Decryption

Method testEncrypt() will isolate the encryption and decryption utilities to test if a given string will encrypt after calling the encrypt function. Next, the return of the encrypt will be used as a parameter in the decrypt function. Both the returns of the two methods will be printed as well as the original string. The decrypt should match the original while the encrypt should be unique.

## Testing Object (UserNode)

Method testObject() will determine if the creation and appending of a UserNode is successful. The append method will be tested will null and a new instance of the node that is constructed with test values for the arguments. The output will show if the test fails or succeeds. Obviously, the null test case should always fail and the valid node should succeed.

## Results and Analysis

The results of the test cases show that the methods are rejecting the improper parameters and accepting the proper ones. In figure 1, we see that null nodes cannot be created or appended to the file and that valid accounts can be made and logged into. Encryption and decryption are working properly given valid input as well.

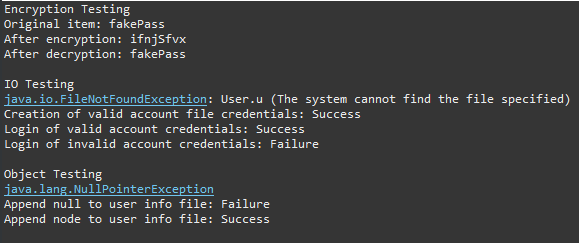


Figure 17: First run of tests (creates new user file)

After the first run of these cases you will notice that the user file has now been created. Running the test again shows that the creation of a valid account is now a failure. This is because the sample is not designed to have more than one account file. The method is meant to return false if this file is already in existence so the failure of this case is favorable.

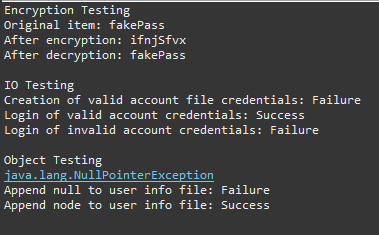
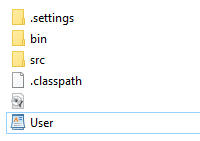
This evaluation the methods and their parameters will be the basis for testing the components of existing modules and those to come later. Using this strategy in the top-down integration method, we can test the components in cycles as they are added to the site. As this is representing a website that will be adding features throughout its lifespan, this will be ideal for growth and evolution.

Figure 18: Run of test cases with user file already created

Figure 19: User file is now created

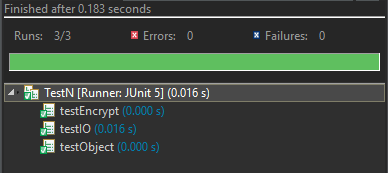


Figure 20: Test case success status

# Comparison of similar designs

The functions provided by our software services are mainly two parts on individual users. The first is musical instrument social and musical instrument trading. At this point, our functional design is similar to the website reverb.com. Reverb.com is a dedicated platform for the sale of online instruments. Because this site only concentrates on instrument sales, it makes it different from the large comprehensive online shopping site such as Ebay. The positioning of the website from a public to a small fixed group can more accurately grasp the needs of musicians, while providing more professional music information sharing (1). For example, excellent private instrument collectors and small but reputational musical instrument stores will have received more market in Reverb. At this point, the concept and function of our website is the same as reverb.

The second feature we offer to individual users is special. Musicians display and promote professional musical instrument information through the sharing of musical works. Because of this feature, we can distinguish it from reverb. We use the artist's work to link to these used instruments. In fact, we have increased the social attributes of music knowledge, and at the same time let interested people increase the chances of buying their favorite instruments.

Our website provides two-way service to business users. Let the musician find the venue that suits his performance. This is achieved by providing free venue information and prices on the website through the venue. This feature is similar to the musiciancasting.com website. It is also a platform for two-way service for musicians and venues. But they are more powerful. On the musician-casting platform, musicians can rent instruments, and professional tour managers can manage and organize their artists' performance plans more effectively. Compared to their profession, our marketing strategy is to let users enjoy the function of booking a place for free. Because there are very few companies that offer such special services in the market, and whether the services on the market can be successfully satisfied with e-commerce services is still unknown. Under the premise of unknown risks, there is a free strategy that is better than the charging strategy. It is uncertain whether the specific needs of music can be met before payment. Because there is not enough market feedback information for analysis and comparison. In the current market environment, musicians' demand for small-scale services is faced with many risks. For example, many large-scale music interconnection companies, such as Spotify, are gradually entering the music service market (2).

# Implementation of the code

In the attached Project file, we have User, User Input/output, and encryot class coded (TestN attached is for testing user mentioned above).

Encryot class is for password envyotion, which the method is changing actual value of the passwords before saving, decryption code is also included. \

UserNode is used as declaring “User” class that will be used in out Project.

UserIO is for the purpose of function regarding on UserNode. In other words, functions like “login”, “create an new account” is coded in here.

**Note: All code is written in Java.**

# Conclusion

Our members did contribute evenly, and most of the project phases and deliverables were completed before the due dates. There are no major changes done to the project. Therefore, judging by the progress and team members interactions, we are expected that Nibiru Project is doable, and should be delivered on time based on the estimation we have in this report. If this project were to be completed in the future, we expect to complete User function first, which is the “social” part of our project, before the transaction part.

# Reference

1. K. Pearsall, “Effects Guide: What is Reverb?,” *Fender Guitars*, 05-Feb-2019. [Online]. Available: https://www.fender.com/articles/tech-talk/pedal-board-primer-reverb/. [Accessed: 19-Apr-2019].

2. M. Errico, “Touring Can't Save Musicians in the Age of Spotify,” *The New York Times*, 25-Jan-2016. [Online]. Available: https://www.nytimes.com/2016/01/25/magazine/touring-cant-save-musicians-in-the-age-of-spotify.html. [Accessed: 19-Apr-2019].

3. “Q: How Much Do Software Tester Jobs Pay per Hour in 2019?,” ZipRecruiter. [Online]. Available: https://www.ziprecruiter.com/Salaries/How-Much-Does-a-Software-Tester-Make-an-Hour. [Accessed: 19-Apr-2019].

4. Salary.com, “Hourly wage for Project Manager I,” Salary.com. [Online]. Available: https://www1.salary.com/Project-Manager-I-hourly-wages.html. [Accessed: 19-Apr-2019].

5. “Q: How Much Do Entry Level Programmer Jobs Pay per Hour in 2019?,” ZipRecruiter. [Online]. Available: https://www.ziprecruiter.com/Salaries/How-Much-Does-an-Entry-Level-Programmer-Make-an-Hour. [Accessed: 19-Apr-2019].