Project Topic:

"E-commerce website to sell secondhand product; Bidding system for AIT."

Software Engineering

Final Software Requirements Specification

(SRS) Document

Shubhangini Gontia

Suyogya Ratna Tamrakar

Younten Tshering

https://github.com/shubhanginigon/Bid-Buy-Sell-Project

27/04/2021

Table of Contents

1. Introduction

In the recent years, the electronic marketplace has been an exponentially grown in usability, size and worth. It is expected that this trend will exaggerate in the upcoming years. Because of the rapidly growing internet environment, the customer can conveniently obtain the products that he/she purchases from the traditional market by online systems. Online platform is a main component of the electronic marketplace that makes use of electronic commerce mechanisms. The idea of our project is like any e-commerce platform in which buying and selling of goods are done. But the core addition will be the system of bidding before buying anything. The products added will be subjected to a certain time frame where users most probably buyers will be allowed to bid on the products and are able to buy the product once the bid time closes or when there are no other bidders. In case of AIT, this platform can be used to sell the used items. More than one person could be interested in the same product, but with this system, there will be a sense of competition as one person bids over another user's bid at the same time and the product value will also increase.

1.1 Scope

We will be using MVC with Client-Server architecture. System backup and recovery will not be handled as there are no long transactions that go in our system. However, we will be storing some instant information in the cache, so that with little interruptions like network connectivity issues users will be able to restore their session data. The application will be using MySQL as a data store. For developing, we are using Java Spring Boot for backend and HTML & CSS for frontend.

In modern software systems, it is not uncommon to have hundreds or thousands of users independently and concurrently interacting with the resources. We generally want to avoid situations when changes made by one client are overridden by another one without even knowing. The main architectural challenges that will be addressed is handling multiple concurrent transactions, real-time information updates and handling sessions.

1.2 Overview

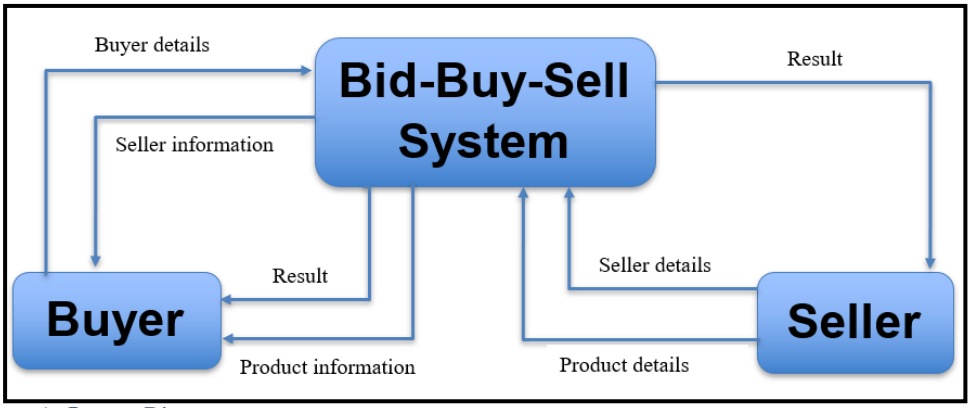

Figure 1. Context Diagram

2. Functional Requirement

2.1 User Module

- Users should be able to register into the system.
- Users should be able to login to the system.
- Users can update their profile details.
- Admin users can see and update user's information.
- Admin users can see the data dashboard.

2.2 Product Module

- Registered users can add new products with detailed description.
- Registered users can update or delete their product details until the bidding time starts.
- Admin users can update or delete product information.
- Sellers can set the bid start time and bid close time for the product.

2.3 Bidding Module (For registered users)

- Users can bid on products only during the valid bidding period.
- Users will be allowed to raise the bid amount only by certain predefined percentages.
- Users will be able to see the number of bidders active on the product.

2.4 Payment Module

- Sellers can determine mode of payment while adding product details.
- Winning users can select the payment method from the options available.

Note:

- ❖ Product, Users, and bidding details provided to the system shall be stored in the MySQL Database.
- ❖ The information shall be accessible via queries and JPA (Java Persistence API).
- ❖ The data stored should be able to be manipulated through forms or interface.

3. Interface Requirements

3.1 User Interfaces

The user interface for this project is the interface provided by HTML and CSS using Spring Boot. Interface includes forms and dashboard for the users to query and organize data to suit their needs. Forms and dashboard both have builders that let the user specify which fields they want to use and which constraints they want to define.

3.2 Hardware Interfaces

The project uses the MySQL. Access hardware is managed by the operating system and MySQL.

3.3 Communications Interfaces

If we decide to implement an Ad Hoc network for a shared database, the operating system will handle those connections.

4. Non-functional Requirements

4.1 Performance

The system shall be designed with high performance level to handle concurrent or multiple bidding and multiple bids of different product at same time. The output is to show who won and is bidding for the product in this system and we need to focus on concurrency, response time and block time.

This system should be available most of the time (mostly in bidding period) and it consists of transactions of lots of users concurrently.

4.2 Security

The system shall be designed with a level of security appropriate for the sensitivity of information enclosed in the database. More interaction is needed with client about the instability of the information. Since there is no obvious information that is of a high security level such as credit card information, the only requirements that could be implemented are encrypting the database and/or making the database password-protected, by user's request.

4.3 Modifiability

System should be easy to modify since data will not remain the same all the time and input sources may change with newer bidders/buyers and sellers. For users with different privileges/roles, the system should modify and make changes accordingly.

4.4 Reliability

Reliability is one of the key attributes of the system. Back-ups will be made regularly so that restoration with minimal data loss is possible in the event of unforeseen events. The system will also be thoroughly tested by all team members to ensure reliability.

4.5 Interoperability

For the website we just import and integrate various information with pictures into the system from the seller.

4.6 Portability

The system shall be designed in a way that allow it to be run on multiple computers with different browsers.

4.7 Scalability

With data, the storage size will increase but can be manage with time. This app can be made vertically scalable when there are issues of memory storage.

4.8 Reusability

The system should be designed in a way that allows the database to be re-used regularly for the various bidding that the organization shall hold.

4.9 Serviceability

The maintenance of the system should be able to be sufficiently performed by any person with a basic understanding of bidding system.

5. Architectural Challenges of our system

5.1 Handling multiple concurrent transactions

There will be many users active on the system and they will be able to bid on the products. But multiple users may place bids on the same product at the same time.

5.2 Updating the product price in real time for ongoing bidding processes.

If the product is being bided by someone, there will be multiple users who will be on the same page waiting or trying to bid. If one bid is updated, other users must instantly see the updated bid amounts on their systems. This must be nearly real-time.

6. Architectural Design

In modern software systems, it is not uncommon to have hundreds or thousands of users independently and concurrently interacting with the resources. We generally want to avoid situations when changes made by one client are overridden by another one without even knowing.

To prevent our data integrity from being violated we often use locking mechanisms provided by database engine, or even use abstractions provided by tools like JPA.

Therefore, in our system optimistic locking will let every client read and write data with the restriction that just before committing the transaction we need to check whether a particular record has not been

modified by someone. This is usually done by adding the current version or last modification timestamp attribute.

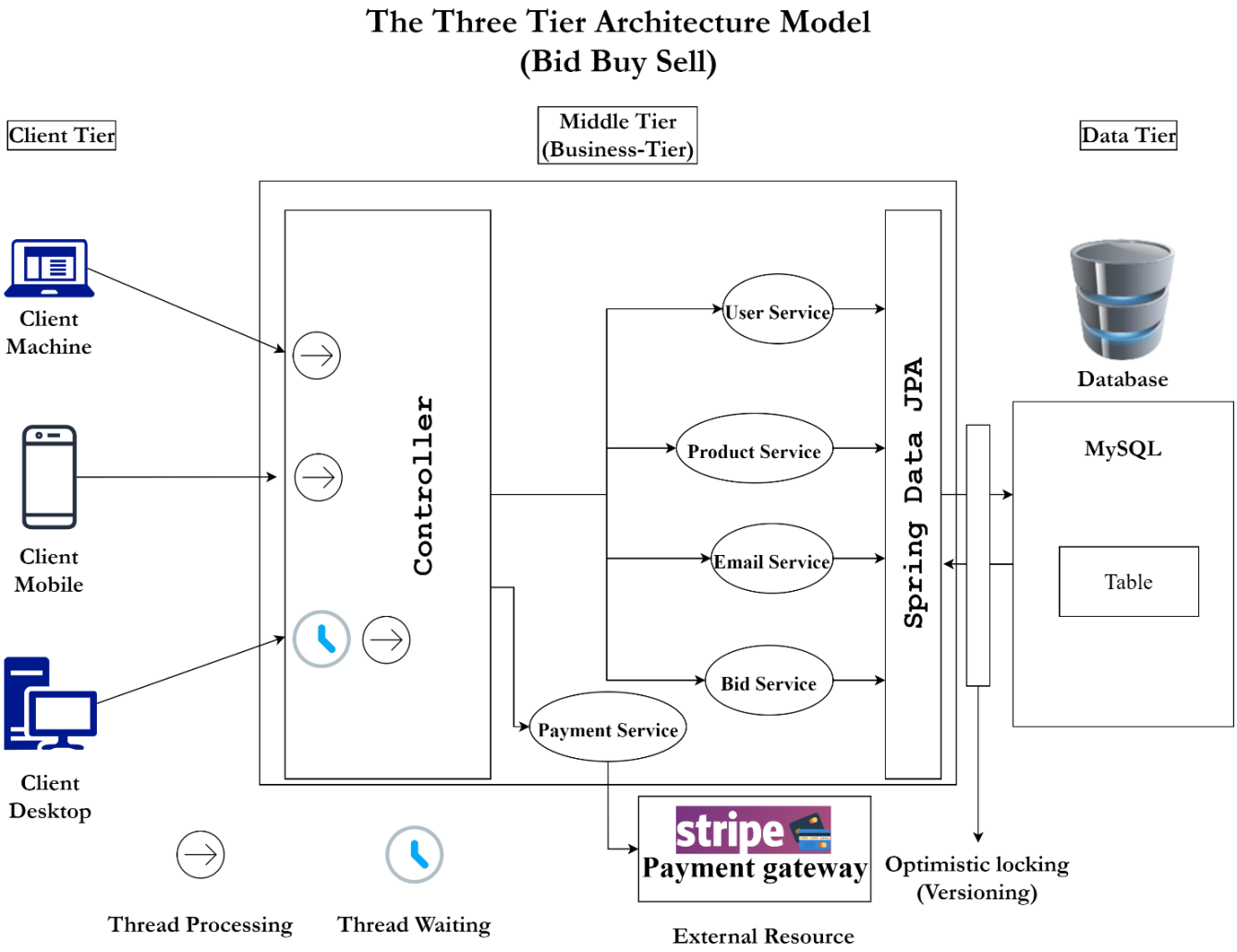

Figure 2. Architectural Design

In Layer Architectural Style, application consists of 3 hierarchically ordered subsystems as follows:

1. user interface,
2. middleware and
3. database system

The middleware subsystem services data requests between the user interface and the database subsystem.

Three Layer architectural style is used for the development of our web app where:

- the Web Browser implements the user interface.
- the Web Server serves requests from the web browser.
- the Database manages and provides access to the persistent data.

7. Quality Attribute Analysis

Quality attributes	Priority Rating (L, M, H)	Priority Justification
Performance	H	The output is to show who won and is bidding for the product in this system and we need to focus on concurrency and response time. This system should be available most of the time and it consists of transactions of lots of users concurrently. ➢ The application should respond to a user within 2 seconds. ➢ The application should be able to handle 20 transactions per second in the peak load time. ➢ The application will be available with the uptime of 95% between 6.00 am to 12.00 am.
Security	M	Since the site is buying and selling product, the payment or transactions with payment gateways might be incorporated so storing all the credit card information of the users, hence requiring high level of security mechanisms. ➢ All the transaction data between client and server must be encrypted. ➢ Since the site is buying and selling product, the payment or transactions with payment gateways will be incorporated, hence requiring high level of security mechanisms. ➢ The system should be able to restore backward data of 24 hours (maximum 3 Days) within 2 hours as a recovery function.
Usability	M	System should be easy to modify since data will not remain the same all the time and input sources may change with newer buyers and sellers. For users with different privileges/roles, the system should modify and make changes accordingly. ➢ The web app must support latest Web browsers for any OS. ➢ The web app should be responsive. ➢ The web app should be easy to operate by users with a certain navigation menu or option. No need for a user manual.

Interoperability	L	For the website we just import and integrate various information with pictures into the system from the seller.
Scalability	L	With data, the storage size will increase but can manage before time. This app can be made vertically scalable when there is lack of memory storage.

8. System Design

This section presents a list of the fundamental sequence diagrams, use cases and class diagram that satisfy the system's requirements. The purpose is to provide an alternative, "structural" view of the requirements stated above and how they might be satisfied in the system.

Use Case Model

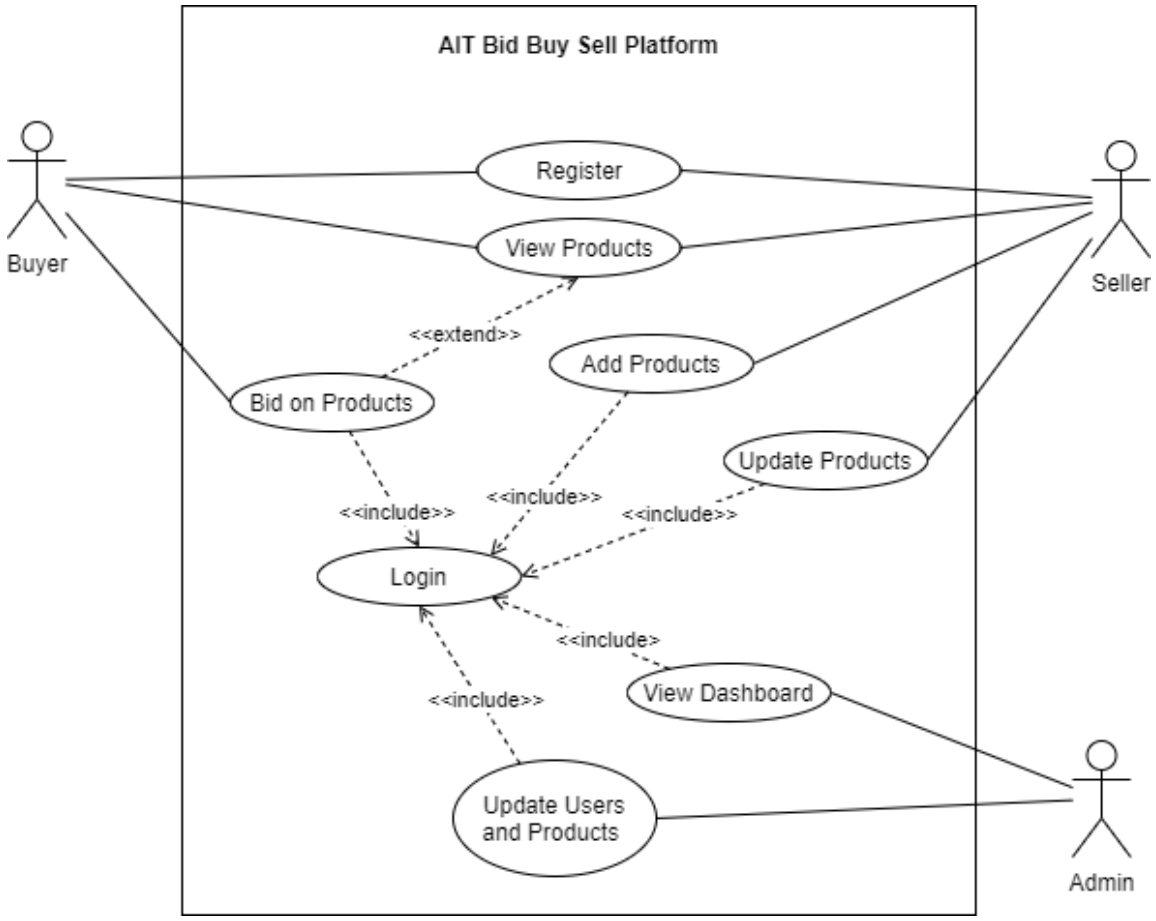

Figure 3. Use Case Diagram

Main actor are buyer and seller of the system where they get registered into system and accordingly add product to sell or bid on product.

Above use case diagram covers the overview of our system and major use case such as **Bid on Product** will be explained in detail and shown in sequence diagram (figure 5).

8.2 System Sequence Diagram

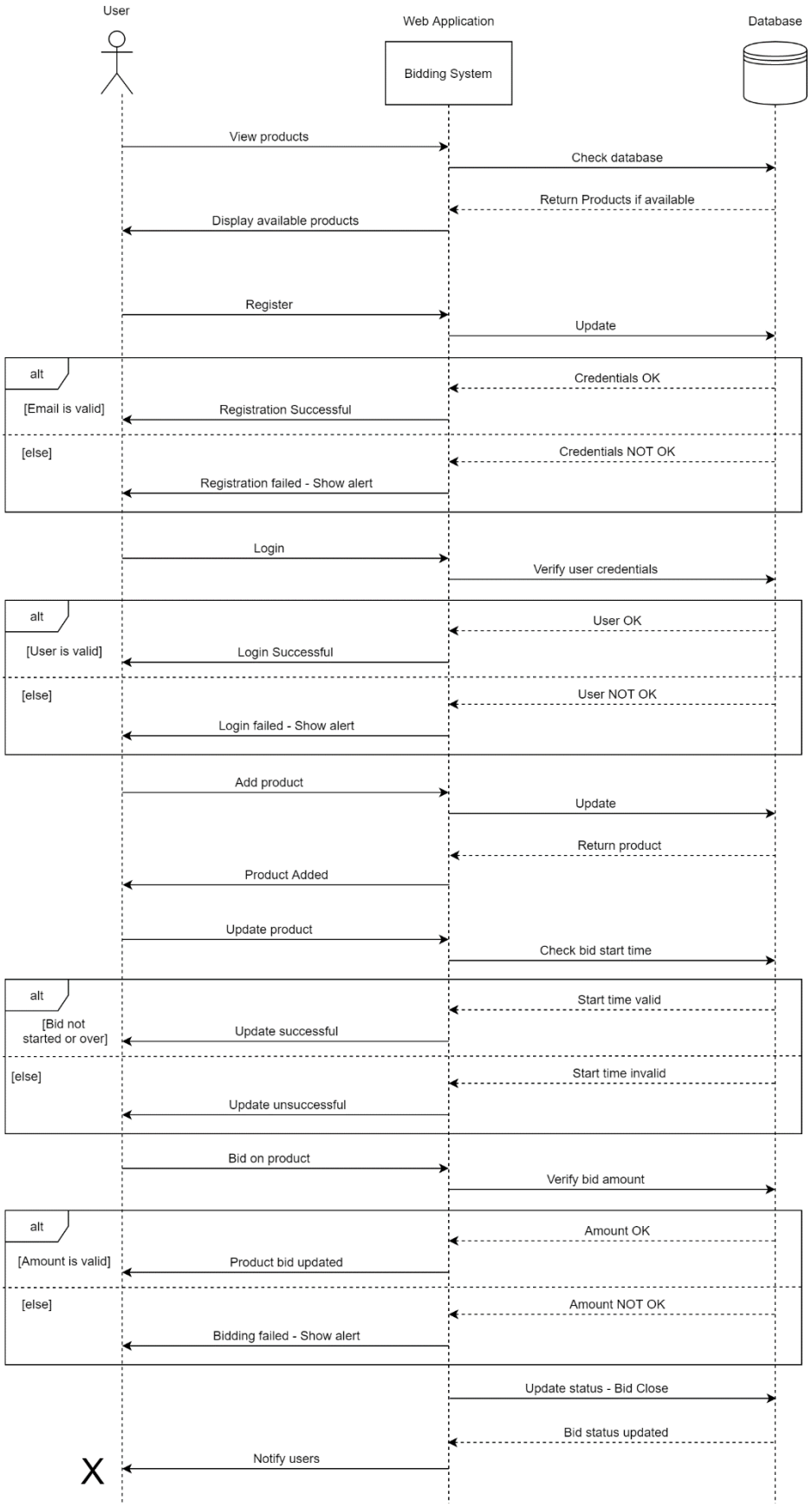

Figure 4. System Sequence Diagram

8.3 Sequence Diagram for Use Case: “Bid on product”

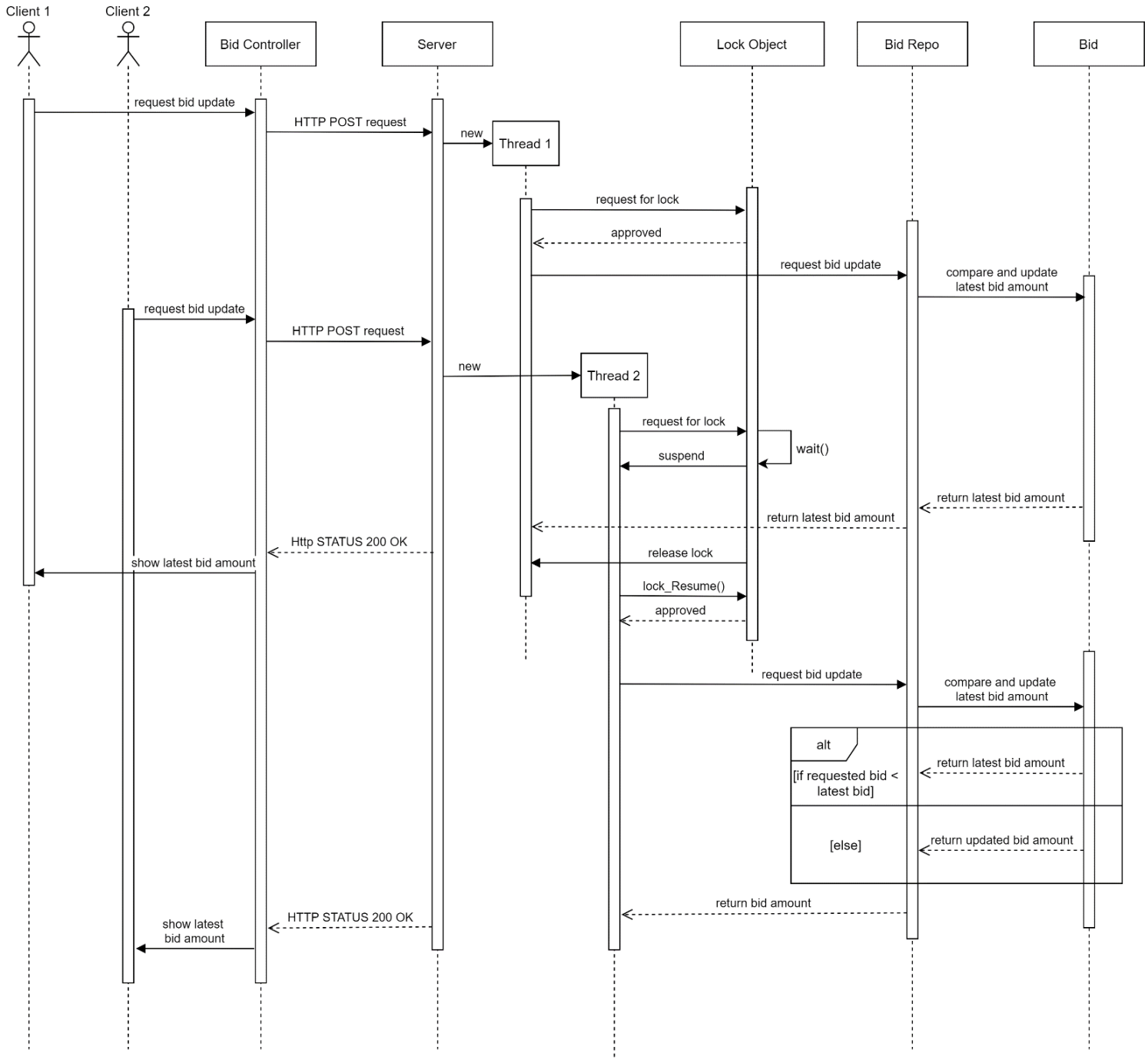

Figure 5. Sequence Diagram of Bid on product

Above sequence diagram is for the use case 'Bid on product'. Here we have shown two actors as client 1 and client 2 and bidController, server, bidRepo and bid (Entity) as the object for the sequence diagram.

As shown in the sequence, when client 1 and client 2 both requests to update the bid amount simultaneously the timestamp taken here will be till nanosecond to diminish the possibility of clicking the bid at the same time. Once, the request is sent to the server, it will use locking to lock object and

further proceed to update the bid entity. When the request from the second actor client 2 is sent to the server, say after 1 nanosecond, it will create another thread for the request and try to request the lock object, but since the thread 1 is already using it thread 2 will be sent to wait state until the lock is released by thread 1. After that, thread 2 will proceed to update the bid, and bid will be updated if the value is more than the latest bid amount.

8.4 System level Class Diagram

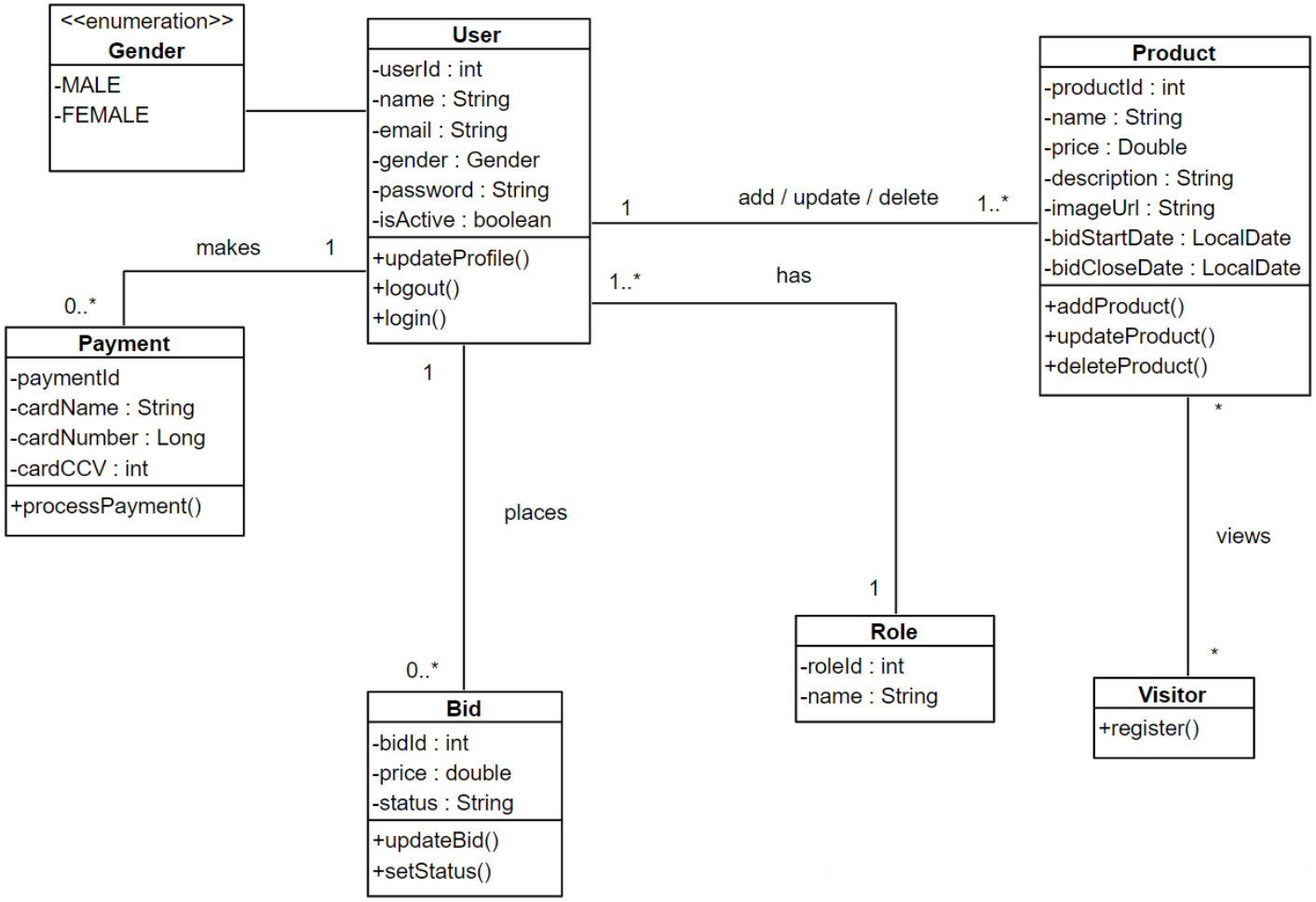

Figure 6. System level Class diagram

8.5 Codebase Class Diagram

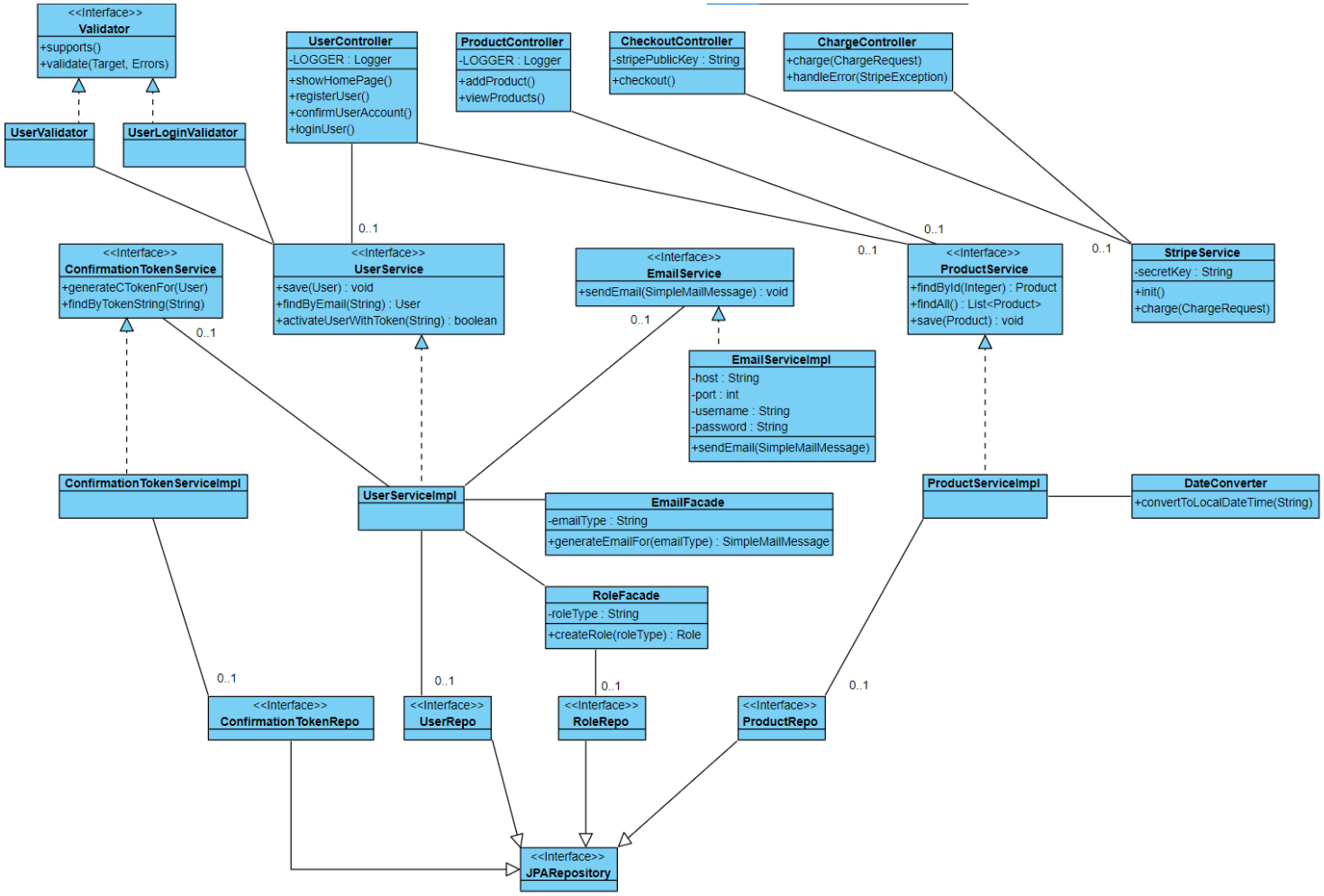

Figure 7. Detailed Class Diagram

9. Risk and mitigation plan

Risk management is the identification assessment, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities.

The main objective of Risk Management is to minimize, monitor and control the probability of infortune events that may occur while doing the project. It is also to assure that uncertainty does not deviate the endeavor from goals thus maximizing the realization of opportunity.

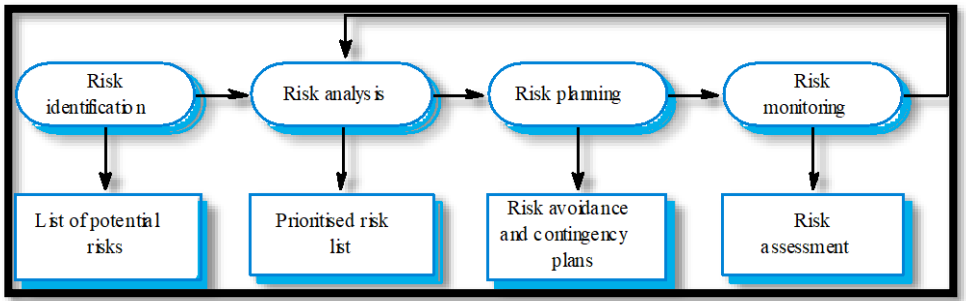

Figure 8. Risk Management process

9.1 Different risks that can occur during the progress of the project

Table 1. Risk Identification

Risk	*Description*	*Example*
Technology	The risk that can come from uncertainty in technologies that we used during our project.	System crash
People	During the progress of the project, the potential problem that might occur due to the group members' contribution.	Members are ill during critical times in the project.
Tools	The risks that occur when the CASE tools which support the project do not perform as expected.	Code is inefficient.
Requirement	While doing the project, if any changes occur in our mind to either add or reduce some functionality on our project then it will lead to requirement risk.	Deletion of one of the functionalities.
Estimation	The risk that are likely to occur due to time of completion and size of the project.	Underestimate time and size

9.2 Risk Analysis

Table 2. Risk Analysis

Risk	**Probability**	**Effect**
System crashes	High	Serious
No internet access	Low	Tolerable
Members not able to come on time due to sickness or other reasons	Moderate	Serious
CASE tool which supports the project do not perform as expected	Moderate	Tolerable
Changes to requirement which require major work design are proposed	Moderate	Serious
Time required to develop the software is underestimated.	High	Serious
The size of the software is underestimated.	Moderate	Tolerable

9.3 Risk Planning

Consider each risk and develop a strategy to manage that risk by:

1. Avoidance strategies

Taking actions that will avoid the risk altogether.

2. Minimisation strategies

Try to either reduce the impact of the risk or reduce the probability of risk arising.

3. Contingency plans

If the risk arises, contingency plans are plans to deal with that risk.

Table 3. Risk Planning

Risk	**Strategies**
System crashes (avoidance)	Constant systems scan and backup the files.
Members unavailable due to some reasons (minimize)	By mitigating the work in between and making an extra effort to meet the deadlines.
CASE tool which supports the project do not perform as expected (avoidance)	We could look for plugins to work with the current CASE tools and if it does not work then, we can use other CASE tools to complete our project on time.
Changes to requirements which require major work. (contingency)	Maximizing information hiding in the design by reviewing the report.
Time required to develop the software is underestimated (contingency)	Create a short-term plan to complete the spill over tasks and give our extra effort
The size of the software is underestimated. (minimization)	Produce simple and effective products so that we can briefly describe the functionality.

9.4 Risk Monitoring

Risk Management is an iterative process:

- Assess each identified risk regularly to decide whether or not it is becoming less or more probable.
- Also assess whether the effects of the risk have changed.
- Each key risk should be discussed at group meetings.

10. Planned Schedule

Table 4. Work Break Down

Activities	**Remarks**
1. Planning	
1.1. Project topic, problem statement and System scope	Done
1.2. Assignment of members, their roles and scheduling	Done
1.3. Setting up project git repo	Done
1.4. Quality attribute analysis	Done
1.5. Background study on architecture and design patterns	Done
2. Project requirements	
2.1. Determining and analysis of system requirements	Done
2.2. Initial UI Mockups design	Done

2.3. Preliminary Use case diagram and sequence diagram	Done
2.4. One series of mockups for one use case.	Done
2.5. Risk and mitigation plan	Done
2.6. Software requirements document.	Done
3. Architectural Design	Done
3.1. Design analysis	Done
3.2. Final Use-case diagrams, class-diagrams, sequence diagrams, deployment diagrams	Done
3.3. Literature review (RELATED WORK)	Done
3.4. Draft Scientific report	Done
4. Implementation	
4.1. Code design patterns	Done
4.2. Home page, login form, registration form	Done
4.3. Email functionality with authentication and authorization	Done
4.4. Product model	Done
4.5. Bidding model	Done
5. Testing	
5.1. Unit testing	Done
5.2. Integration testing	
5.3. Performance testing	Done
5.4. Stress testing	Done
5.5. Acceptance testing	
5.6. Testing Documentation	Done
5.7. Update Scientific report	Done
6. Documentation	
6.1. Presentation slide	Done
6.2. Final Scientific report	Done
6.3. All updated Requirement, Design, Testing documents	Done