Arab Open University- Saudi Arabia



Faculty of Computer Studies

***Tools Rental Platform***

Laith Hussam Shono 101708516

TM471: Final Year Project, 2021  
Supervisor: Dr. Basil Kasasbeh

# Table of Contents

[Table of Contents ii](#_Toc68958751)

[Introduction 1](#_Toc68958752)

[Chapter 2 - Literature Review 2](#_Toc68958753)

[Chapter 3 - Requirements and analysis 4](#_Toc68958754)

[3.1 Domain modelling: 4](#_Toc68958755)

[3.1.1 Business use cases: 4](#_Toc68958756)

[3.1.2 use case diagram: 4](#_Toc68958757)

[3.1.3 Textural description of the use cases: 5](#_Toc68958758)

[3.1.4 Activity diagrams 14](#_Toc68958759)

[3.1.5 Conceptual model 17](#_Toc68958760)

[3.2 Functional Requirements 18](#_Toc68958761)

[3.3 Non-Functional Requirements 29](#_Toc68958799)

[3.3.1 Look and feel requirements 29](#_Toc68958800)

[3.3.2 Usability and humanity requirements 29](#_Toc68958801)

[3.3.3 Performance requirements 29](#_Toc68958802)

[3.3.4 Maintainability and support requirements 30](#_Toc68958803)

[3.3.5 Legal requirements 30](#_Toc68958804)

[3.4 Analysis 31](#_Toc68958805)

[3.4.1 Constraints on objects attributes 31](#_Toc68958806)

[3.4.2 System Operations 33](#_Toc68958807)

[3.4.3 Analysis class diagram 36](#_Toc68958808)

[3.5 Glossary 37](#_Toc68958809)

[3.5.2 Glossary of class attributes 37](#_Toc68958810)

[Table of Figures 39](#_Toc68958811)

[References 40](#_Toc68958812)

# Introduction

The internet helped change the way we live. It allowed information to be available to everyone. Including information on how to do specific jobs, some of which we always hired a professional to do. This resulted in a lot of people start doing these jobs themselves. According to (National Association of REALTORS, 2019), 47% of homeowners do projects on their homes by themselves. However, some of these jobs require specific tools that not everyone may have. Some people may go and buy such tools, just to end up using them once and never again. Which results in a loss of money, and the tools taking space and cluttering up the place.

A solution is to rent the tools instead of buying them. However, this may be harder than it sounds, as it’s not easy to find a place to rent specialized tools, and there are not many rental platforms online. And most of the ones available are businesses-to-consumer websites, which don’t allow people to rent their own tools. The few ones that are peer-to-peer either lack major features or are not available for everyone.

The solution is a peer-to-peer e-commerce rental platform. Allowing the owners to offer their tools for rent, and making a profit out of them. And allowing others to request to rent other people’s tools, and avoid buying a new one. The platforms also provide a unified place where people can look for tools to rent, instead of searching for hours across multiple sites online.

In this report, I will first go through my research on the subject and the methods I found and will be following. Then, I will go through the domain modeling process, identifying the use cases and their textural description. Then I will list the functional and non-functional requirements. And finally, I will list the system constraints and operations. And a glossary of all the important terms will be at the end of chapter 3.

# Chapter 2 - Literature Review

When I chose this subject, I was mostly concerned with people stealing or damaging the tools. So, I searched for methods people use to prevent that. I also searched for current platforms that people use and learnt what works best and what doesn’t.

First, I started searching for platforms and methods of renting tools online. I found most of them were business-to-consumer (B2C), meaning they were businesses offering their own tools for rent/hire. Where you select a tool, check for availability in one of their stores, and pick it up from there. Two of these businesses were HSS Hire and United Rentals.

I found an app that did offer a good peer-to-peer (P2P) tools renting services. The app is Sparetoolz. However, the app has a really small user base, thus there’s almost no tools offered. And that the app is only available on the Apple App Store and planned to come soon to Google Play, With no announced plans for the web. So, with all of these problems, I couldn’t get much from Sparetoolz.

So, I searched for general renting platforms (not specific for tools). I found the most widely used in the app store were Rent4Me, MyRent, and Rent iT. However, MyRent, and Rent iT are only available in the United States. But, Rent4Me is available globally, it’s on the top of the rent apps chart, and it’s available on Android, iOS, and the web. So, I used it the most to compare and research the subject.

While Rent4Me seemed perfect at first. I found some of the methods it uses are limiting the system. Offering something for example, could take a while as you need the administrators to accept the post first, then it can be published and available for others. The same thing for IDs, they must be approved by the administrators before the user is verified. While these methods seem better and more secure at first, it could actually push away the users specially when the user base gets bigger and moderation starts taking longer, and it also delay the user experience. Instead, it’s better to let the users evaluate the posts and users’ IDs themselves, while giving them instructions and tips on how to do so safely.

Another concerning choice with Rent4Me is that unverified users (users who didn’t provide an ID or phone number) can still offer and request stuff, and this a security concern for a lot of users.

In addition, I found multiple bugs and problems with the system, here are some of the most troubling and important to avoid ones:

* phone validation doesn't work. I tried on my personal phone number and I tried an online USA number. In both cases the validation message doesn’t arrive. Therefor I couldn’t get my account fully verified.
* The user interface is confusing to navigate, and it has an old design.
* The search feature is a bit buggy specially when zooming out in the map view.
* Sometimes loading pages become so slow (more than 5 seconds).
* Uploading pictures is buggy where sometimes you have to upload it multiple times to work.
* Lack of feedback after user interactions. For example: when uploading IDs, it won't give a feedback that the image was uploaded and you need to refresh the page to see it

Another method I found lots of people using is renting through social media platforms such as Instagram, Twitter, Facebook…etc. Where they post about their tools and receive requests in the direct-messages. Then paying in-cash on delivery.

Then, I started looking for the best methods of the renting process. I found the best and most used method is for the owner to set an insurance amount/deposit that will be held on the renter’s credit card, and will be returned once the tool is returned in a good condition.

Another method, is the renter and owner providing their IDs so they are verified and in case of any problems they can take the problem to the police or court. This is also helpful on the system side, as it makes banning a user from the system much easier as everyone has only one ID number.

# Chapter 3 - Requirements and analysis

## 3.1 Domain modelling:

### 3.1.1 Business use cases:

1. Offer tool
2. Edit/remove tool post
3. Accept a tool-request
4. Deliver tool
5. Request a Tool
6. Edit/remove tool request
7. Return tool
8. Create an account
9. Log into account
10. Add/edit/remove user review
11. Ban a user

### 3.1.2 use case diagram:



Figure - use case diagram

### 3.1.3 Textural description of the use cases:

**ID:** UC1

**Name:** Offer a tool

**Initiator:** Owner

**Goal:** creating a post for a tool.

**Precondition:**

1. The owner must be logged in and provided his/her ID number.
2. The owner must have provided his/her credit card details.

**Postcondition:** A post will be created where others can request to rent the tool.

**Main success scenario:**

1. The owner requests to create a new post.
2. The system ensures the owner is authorized for posting (provided ID number and credit card).
3. The owner enters the post details.
4. The owner presses the confirm button.
5. The system ensures that all inputs are valid.
6. The system creates a post in the database.
7. The system directs the owner to the post page.

**Extensions:**

* + 1. The owner isn’t authorized to post.
    2. The request will be denied.
    3. The system displays a prompt for the user.
    4. Some fields are not valid.
    5. The request will not be created.
    6. The system will inform the user of the invalid values.
    7. Continue at step 3.

**ID:** UC2 - A

**Name:** Edit tool post.

**Initiator:** Owner.

**Goal:** Editing a post of a tool.

**Precondition:**

1. The post must exist.
2. The owner must be authorized to edit the post.

**Postcondition:** The tool’s post will be changed as the owner specifies.

**Main success scenario:**

1. The owner requests to edit a post.
2. The system ensures the owner is authorized to edit the post.
3. The owner edits the desired post details.
4. The owner confirms the changes.
5. The system ensures that all inputs are valid.
6. The system changes the post details in the database.

**Extensions:**

* + 1. The owner isn’t authorized to edit the post.
    2. The request will be denied.
    3. The system displays a prompt for the owner.
    4. Some fields are not valid.
    5. The request will not be created.
    6. The system will inform the user of the invalid values.
    7. Continue at step 3.

**ID:** UC2 - B

**Name:** Remove tool post

**Initiator:** Owner

**Goal:** Deleting a post of a tool.

**Precondition:**

1. The post must exist.
2. The owner must be authorized to remove the post.

**Postcondition:** The post will be deleted and removed from the database. And users won’t be able to access it anymore.

**Main success scenario:**

1. The owner requests to remove a post.
2. The system ensures the owner is authorized to remove the post.
3. The owner confirms that (s)he wants to delete the post.
4. The system deletes any requests to the post.
5. The system deletes the post from the database.

**Extensions:**

* + 1. The owner isn’t authorized to post.
    2. The request will be denied.
    3. The system displays a prompt for the owner.

**ID:** UC3

**Name:** Accept request

**Initiator:** Owner

**Goal:** Accepting a renter’s request to rent the owner’s tool.

**Precondition:**

1. A request must have been sent from a renter on a post.
2. The owner must be authorized to accept/reject requests for the post

**Postcondition:** The request will be accepted, a chat room will open for the owner and renter, and the tool will be marked as unavailable.

**Main success scenario:**

1. The owner requests to see the requests for the post.
2. The system ensures the owner is authorized to view and accept requests for the post.
3. The owner accepts the renter’s request.
4. The system marks the tool as ‘Unavailable’
5. The system opens a chat room between the renter and owner.
6. The system sends a notification for the renter.

**Extensions:**

* + 1. The owner isn’t authorized.
    2. The request will be denied.
    3. The system displays a prompt for the owner.
    4. The owner rejects the renter’s request.
    5. The system deletes the request.
    6. Continue at step 3.

**ID:** UC4

**Name:** Deliver tool

**Initiator:** Owner

**Goal:** Handing over the tool to the renter, and starting the rent period.

**Precondition:**

1. The owner and renter agreed to meet at a place.

**Postcondition:** the renter will get the tool and the rent starts.

**Main success scenario:**

1. The owner and renter confirm their arrival to the meeting place and that they met.
2. The owner and renter take pictures and/or videos to document the state of the tool before the hand-over.
3. The owner and renter upload the pictures and/or videos to the system.
4. The owner and renter confirm the pictures and/or videos.
5. The system gives the owner the renter’s ID number, and gives the renter the owner’s ID number.
6. The owner and renter ask for each other’s ID and confirm they match the ones in the system.
7. The system deducts the insurance-price and the intended rent from the renter’s credit card.
8. The system pays the owner the rent-price.
9. The owner gives the tool to the renter.
10. The system updates the database and sets the rent start time.

**Extensions:**

* + 1. Either the owner or renter didn’t confirm arrival and meeting
    2. The delivery will not continue.
    3. Either the owner or renter doesn’t agree on the pictures and/or videos.
    4. The owner and renter try again and go back to step 2.
    5. If they didn’t agree: the owner/renter cancels the delivery and tool-request.
    6. The system deletes the tool-request.
    7. The tool is marked as available.
    8. The owner or renter refuse to give their IDs.
    9. The owner or renter cancels the delivery and tool-request.
    10. The system deletes the tool-request.
    11. The tool is marked as available.
    12. The system fails to deduct the money
    13. The operation gets canceled
    14. The system fails to pay the owner
    15. The operation gets canceled

**ID:** UC5

**Name:** Request tool

**Initiator:** Renter

**Goal:** Sending a request to rent a tool on a certain post.

**Precondition:**

1. The renter must be logged in and provided his/her ID number.
2. The renter must have provided his/her credit card details.
3. The renter must not have a previous request for the tool (i.e., renter can only have one request to a tool).

**Postcondition:** a request will be sent to the owner on the tool post, where the owner could decide to accept or reject it.

**Main success scenario:**

1. The renter requests to send a *request* to a post.
2. The system ensures the renter is authorized for requesting (provided a valid credit card details).
3. The system ensures the renter doesn’t have a previous request on the post.
4. The renter enters the request details.
5. The renter confirms the request.
6. The system ensures all inputs are valid.
7. The system updates the database and adds the request.
8. The system sends a notification for the owner.

**Extensions:**

* + 1. The renter isn’t authorized (didn’t provide a valid credit card details).
    2. The request will be denied.
    3. The system displays a prompt for the renter.
    4. The system directs the user to a screen to enter his/her credit card details.
    5. The renter has a previous request on the post.
    6. The request will be denied.
    7. The renter will be redirected to the previous page.
    8. Some fields are not valid.
    9. The tool-request will not be created.
    10. The system will inform the renter of the invalid values.
    11. Continue at step 4.

**ID:** UC6-A

**Name:** Edit a request

**Initiator:** Renter

**Goal:** Editing the details of a request.

**Precondition:** The renter must be authorized to edit the request.

**Postcondition:** The request details will be changed as the renter specifies.

**Main success scenario:**

1. The renter requests to edit a *request*.
2. The system ensures the renter is authorized to edit the request.
3. The renter enters the changes (s)he desires.
4. The renter confirms the changes.
5. The system ensures all inputs are valid.
6. The system updates the request in the database.

**Extensions:**

* + 1. The renter isn’t authorized.
    2. The request will be denied.
    3. The system displays a prompt for the renter.
    4. Some fields are not valid.
    5. The edit will be denied.
    6. The system will inform the renter of the invalid values.
    7. Continue at step 3.

**ID:** UC6-B

**Name:** Remove a request

**Initiator:** Renter

**Goal:** Removing (canceling) a request.

**Precondition:** The renter must be authorized to remove the request.

**Postcondition:** The request will be removed from the database, and becomes unavailable to the owner.

**Main success scenario:**

1. The renter requests to remove a *request*.
2. The system ensures the renter is authorized to remove the request.
3. The renter confirms the decision.
4. The system removes the request from the database.

**Extensions:**

* + 1. The renter isn’t authorized.
    2. The request will be denied.
    3. The system displays a prompt for the renter.

**ID:** UC7

**Name:** Return a tool

**Initiator:** Renter

**Goal:** Handing back the tool to the owner

**Precondition:**

1. The owner and renter agreed to meet at a place.

**Postcondition:** The tool is back with the owner and it’s marked as available.

**Main success scenario:**

1. The owner and renter confirm their arrival to the meeting place.
2. The owner confirms the state of the tool is unchanged.
3. The renter returns the tool.
4. The system calculates the remaining of the insurance money.
5. The system returns the remaining of insurance money to the renter.
6. The system sends the deducted money to the owner
7. The system ends the rent.
8. The system marks the tool as available and updates the database.

**Extensions:**

* + 1. The owner or renter doesn’t confirm arrival and meeting.
    2. If the rent period has ended, the user has 1 day to deliver the tool to the authorities and asks for a paper to confirm the delivery.
    3. The renter uploads the paper.
    4. The admin reviews the paper and confirm it.
    5. The system sends a notification for the owner of the tool position.
    6. If the renter didn’t deliver the tool each day’s rent will be deducted from the insurance money until the renter returns the tool or until the insurance money is all deducted.
    7. The owner claims the tool is damaged.
    8. The renter admits it
    9. The renter and owner agree on a compensation price.
    10. Continue at step 5.
    11. The owner claims the tool is damaged.
    12. The renter refuses the claims.
    13. The owner and renter take pictures and/or videos to show the state of the tool.
    14. The owner and renter upload their pictures and/or videos to the system.
    15. The system creates a disagreement case.
    16. The admin reviews the case and arrives to a decision.
    17. The system sends a notification to the renter and owner informing them of the decision.
    18. If the tool is indeed damaged, continue at step 2.1.3
    19. If it’s not damaged, the owner and renter agree on another meeting.
    20. Continue at step 1 and skip step 2.

**ID:** UC8 - A

**Name:** Create an account using an email address.

**Initiator:** User

**Goal:** Creates an account for the user to log into.

**Precondition:**

1. The user must have an email address.
2. The user must have an ID number.
3. The email address and ID number must not be on the banned-list.

**Postcondition:** an account is created for the user to log into, where (s)he can rent or offer tools.

**Main success scenario:**

1. The user requests to create a new account.
2. The user enters the name, email address, ID number, and the password.
3. The system ensures all inputs are valid.
4. The system ensures the user isn’t on the banned-list.
5. The user enters his/her credit card details.
6. The system creates a new user account with the provided details.

**Extensions:**

* + 1. Some fields are not valid.
    2. The request will be denied.
    3. The system will inform the renter of the invalid values.
    4. Continue at step 2.
    5. The user is on the banned list.
    6. The request will be denied and the account will not be created.
    7. The system will display a prompt for the user.
    8. The system will redirect the user to the homepage.

**ID:** UC8 - B

**Name:** Create an account using Google, Facebook, Microsoft, or Apple account.

**Initiator:** User

**Goal:** Creates an account for the user to log into.

**Precondition:**

1. The user must have a Google, Facebook, Microsoft, or Apple account.
2. The user must have an ID number.
3. The email address and ID number must not be on the banned-list.

**Postcondition:** an account is created for the user to log into, where (s)he can rent or offer tools.

**Main success scenario:**

1. The user requests to create a new account.
2. The user signs in using one of the sign-in buttons for Google, Facebook, Microsoft, and Apple.
3. The user logs in the selected account.
4. The user enters the ID number.
5. The system ensures the user isn’t on the banned-list.
6. The user enters his/her credit card details.
7. The system creates a new user account with the provided details.

**Extensions:**

* + 1. The user is on the banned list.
    2. The request will be denied and the account will not be created.
    3. The system will display a prompt for the user.
    4. The system will redirect the user to the homepage.

**ID:** UC9 - A

**Name:** Log into account using an email address.

**Initiator:** User

**Goal:** Log the user into his/her account where (s)he can offer and/or rent tools.

**Precondition:**

1. The user must have previously created an account.
2. The user account must not be banned.

**Postcondition:** the user is logged into his/her account, and (s)he can start renting or offering tools.

**Main success scenario:**

1. The user requests to log into his/her account.
2. The system asks for the email-address and the password.
3. The user enters the email-address and the password.
4. The system checks if the email-address and the password are correct.
5. The system ensures the user is not banned.
6. The system logs the user into the account.

**Extensions:**

* + 1. the email-address and the password are not correct.
    2. The user will not be logged in.
    3. The system will inform the renter.
    4. Continue at step 2.
    5. The user is on the banned list.
    6. The request will be denied and the user will not be logged in.
    7. The system will display a prompt for the user.
    8. The system will redirect the user to the homepage.

**ID:** UC9 - B

**Name:** Log into account using Google, Facebook, Microsoft, or Apple account.

**Initiator:** User

**Goal:** Log the user into his/her account where (s)he can offer and/or rent tools.

**Precondition:**

1. The user must have previously created an account.
2. The user account must not be banned.

**Postcondition:** the user is logged into his/her account, and (s)he can start renting or offering tools.

**Main success scenario:**

1. The user requests to log into his/her account.
2. The user logs in using one of the sign-in buttons for Google, Facebook, Microsoft, and Apple.
3. The user logs in the selected account.
4. The system ensures the user is not banned.
5. The system logs the user into the account.

**Extensions:**

* + 1. The user is on the banned list.
    2. The request will be denied and the account will not be created.
    3. The system will display a prompt for the user.
    4. The system will redirect the user to the homepage.

**ID:** UC10 - A

**Name:** Add/edit user rating.

**Initiator:** Rating-user

**Goal:** Adds or edit a review of a rated-user.

**Precondition:**

1. The rating-user and rated-user must have a previous transaction to create a new review.
2. The rating-user must be authorized to edit the review.

**Postcondition:** Adds/edits a review to the rating of the rated-user.

**Main success scenario:**

1. The rating-user requests to rate the rated-user.
2. The system ensures the precondition.
3. The rating-user enters the details.
4. The rating-user confirms the changes.
5. The system ensures all inputs are valid.
6. The system creates/edits the review.

**Extensions:**

* + 1. The rating-user and rated-user didn’t have a previous transaction or the rating-user is not authorized to edit the review.
    2. The request will be denied.
    3. The system displays a prompt for the renter.
    4. Some fields are not valid.
    5. The request will be denied and the review won’t be created.
    6. The system will inform the renter of the invalid values.
    7. Continue at step 3.

**ID:** UC10 - B

**Name:** Remove user rating.

**Initiator:** Rating-user

**Goal:** Deletes the rating-user’s review of the rated-user.

**Precondition:** The Rating-user must be authorized to delete the review.

**Postcondition:** The rating-user’s review of the rated-user will be deleted, and rated-user rating will be updated.

**Main success scenario:**

1. The rating-user asks for the review to be deleted.
2. The rating-user confirms his/her choice.
3. The system ensures the precondition.
4. The system deletes the review.
5. The system updates the rated-user rating.

**Extensions:**

* + 1. The rating-user isn’t authorized.
    2. The request will be denied.
    3. The system displays a prompt for the renter.

**ID:** UC11

**Name:** Ban user

**Initiator:** Admin

**Goal:** Permanently ban a user from the platform.

**Precondition:**

**Postcondition:** the user will be permanently banned from the platform, using their email address, and ID number if provided.

**Main success scenario:**

1. The admin requests to ban a user.
2. The admin enters the ban details.
3. The admin confirms his/her decision.
4. The system will add the user email-address and ID number (if provided), to the banned-list.

### 3.1.4 Activity diagrams



Figure - Activity diagram of accepting/rejecting tool-requests



Figure - Activity diagram of delivering a tool



Figure - Activity diagram of returning a tool

### 3.1.5 Conceptual model

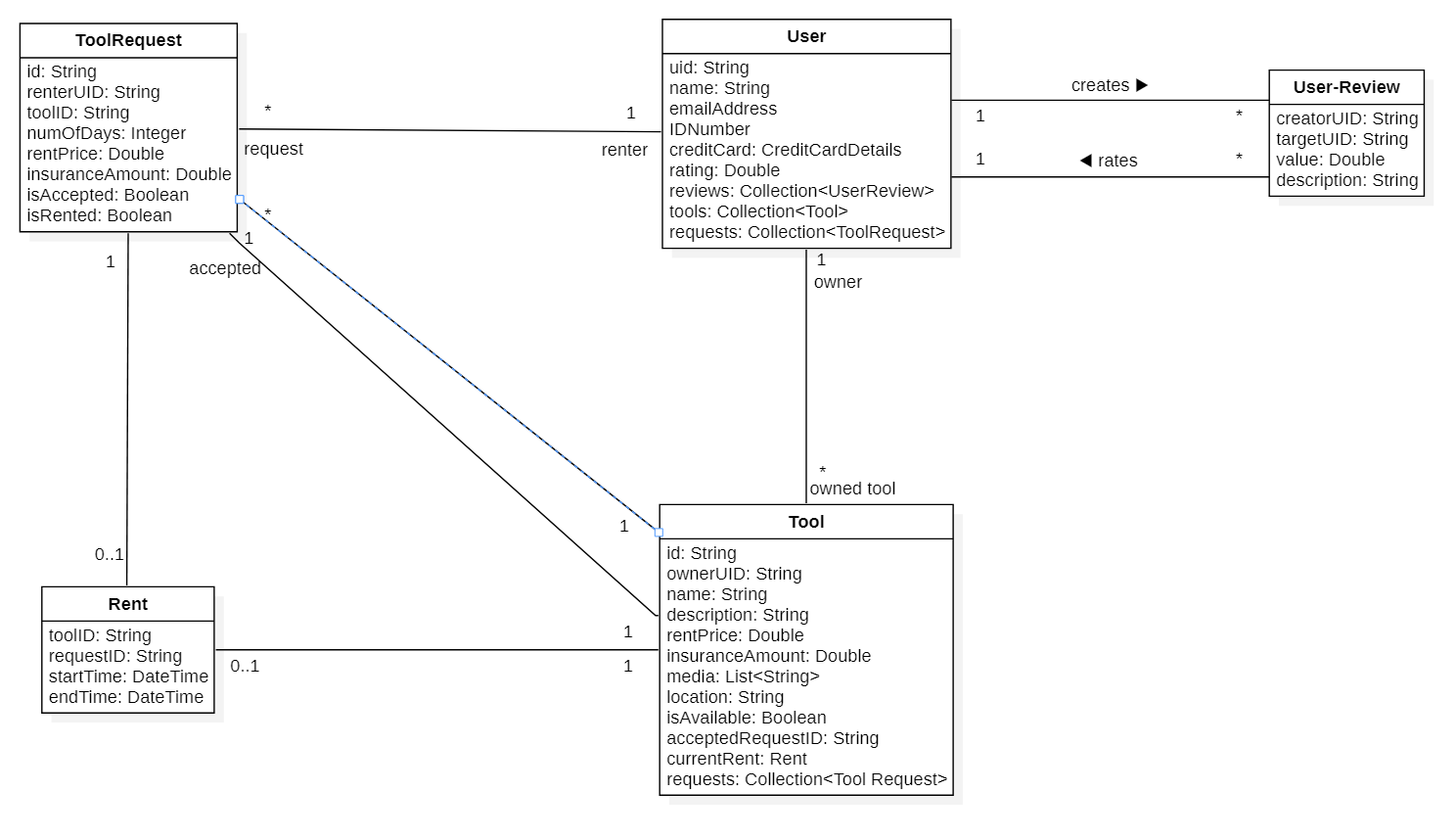


Figure - The Conceptual model

## 3.2 Functional Requirements

### 3.2.1. Functional requirement 1

* **ID:** FR1
* **BUC:** UC1, UC2, UC5, UC6, UC8, UC9, UC10
* **Description:** The system must accept a range of inputs from the user
* **Fit Criterion:** A range of inputs will be accepted
* **Rationale:** To be able to read inputs from the user
* **Dependencies:** None
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.2. Functional requirement 2

* **ID:** FR2
* **BUC:** UC8
* **Description:** The system must allow the user to create an account to log into.
* **Fit Criterion:** The Sign-up screen will be displayed
* **Rationale:** To be able to identify the user and allow the user to access the system functionalities
* **Dependencies:** FR1
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.3. Functional requirement 3

* **ID:** FR3
* **BUC:** UC8-B
* **Description:** The system must allow the user to log-in/sign-up using a Google, Facebook, Microsoft, or Apple account.
* **Fit Criterion:** "Log-in with" buttons will be displayed in the log-in and sign-up screens
* **Rationale:** To add an easier and quicker method of singing in
* **Dependencies:** FR1, FR2
* **Priority:** Medium
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.4. Functional requirement 4

* **ID:** FR4
* **BUC:** UC8
* **Description:** The system must allow the user to enter his/her ID number.
* **Fit Criterion:** a field to enter ID number will be displayed
* **Rationale:** To ensure the user's identity to prevent scams
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.5. Functional requirement 5

* **ID:** FR5
* **BUC:** UC9
* **Description:** The system must allow the user to log into his/her account
* **Fit Criterion:** The log-in screen will be displayed
* **Rationale:** To be able to identify the user and allow the user to access the system functionalities
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.6. Functional requirement 6

* **ID:** FR6
* **BUC:** UC9
* **Description:** The system must be check if the log-in details are correct
* **Fit Criterion:** A wrong log-in details will be rejected
* **Rationale:** To prevent unauthorized access to a user's account
* **Dependencies:** FR2
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.7. Functional requirement 7

* **ID:** FR7
* **BUC:** UC11
* **Description:** The system must allow the admins to ban a user
* **Fit Criterion:** The banned user's email address and ID number will be added to the banned list
* **Rationale:** To prevent the banned user from ever accessing the system's functionalities
* **Dependencies:** FR1, FR2
* **Priority:** Medium
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: Admin requirement

### 3.2.8. Functional requirement 8

* **ID:** FR8
* **BUC:** UC8, UC9
* **Description:** The system must be able to check if the user is banned
* **Fit Criterion:** A user's email or ID that's on the banned list will be detected and his/her request will be denied
* **Rationale:** To prevent a banned user from accessing the system
* **Dependencies:** FR7, FR2
* **Priority:** Medium
* **Support materials:** None
* **History:** Created 4/3/21

Edited 9/3/21

* **Type**: User requirement

### 3.2.9. Functional requirement 9

* **ID:** FR9
* **BUC:** UC1, UC4, UC5, UC8
* **Description:** The system must allow the user to enter a valid credit card detail.
* **Fit Criterion:** fields to enter card details will be displayed
* **Rationale:** To be able to send and receive payments
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21

Edited 9/3/21

* **Type**: User requirement

### 3.2.10. Functional requirement 10

* **ID:** FR10
* **BUC:** UC1, UC4, UC5, UC8
* **Description:** The system must be able to determine if a card is valid or not
* **Fit Criterion:** an invalid card will be denied.
* **Rationale:** To prevent fake cards
* **Dependencies:** FR9
* **Priority:** High
* **Support materials:** None
* **History:** Created 9/3/21
* **Type**: System requirement

### 3.2.11. Functional requirement 11

* **ID:** FR11
* **BUC:** UC1
* **Description:** The system must allow the user (owner) to create a new post
* **Fit Criterion:** a screen to create a new post will be displayed
* **Rationale:** To rent their tools, and for renters to send them tool-requests
* **Dependencies:** FR1, FR2, FR12
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: Owner requirement

### 3.2.12. Functional requirement 12

* **ID:** FR12
* **BUC:** UC1
* **Description:** The system must be able to determine if a user is authorized to create a new post.
* **Fit Criterion:** The user will get a prompt telling them they can’t post if they are not logged in or didn’t provide an ID number and card details.
* **Rationale:** To prevent unauthorized accounts from creating posts and filling the database
* **Dependencies:** FR2, FR4
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: Owner requirement

### 3.2.13. Functional requirement 13

* **ID:** FR13
* **BUC:** UC2
* **Description:** The system must allow the owner to edit and delete their posts
* **Fit Criterion:** a screen to edit the post will be displayed with a button to delete the post
* **Rationale:** To prevent unauthorized users from posting
* **Dependencies:** FR1, FR2, FR11
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21

Edited 9/3/21

* **Type**: Owner requirement

### 3.2.14. Functional requirement 14

* **ID:** FR14
* **BUC:** UC2
* **Description:** The system must be able to determine if a user is authorized to edit or remove a post
* **Fit Criterion:** The user's request to edit or delete will be denied if (s)he is not authorized to
* **Rationale:** To prevent the posts from being edited by unauthorized user
* **Dependencies:** FR2, FR11, FR13
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: Owner requirement

### 3.2.15. Functional requirement 15

* **ID:** FR15
* **BUC:** UC3,
* **Description:** The system must be able to send notifications to a user.
* **Fit Criterion:** The user will receive a notification
* **Rationale:** To alert the user of any important information
* **Dependencies:** -
* **Priority:** Medium
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.16. Functional requirement 16

* **ID:** FR16
* **BUC:** UC3
* **Description:** The system must be able to determine if a user is authorized to view, accept, and deny tool-requests to a certain post.
* **Fit Criterion:** an unauthorized user will not be able to view, accept, or deny a tool-request
* **Rationale:** To either rent the tool to the renter or refuse his request and remove it from the tool-requests list
* **Dependencies:** FR2, FR11, FR21
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: Owner requirement

### 3.2.17. Functional requirement 17

* **ID:** FR17
* **BUC:** UC3
* **Description:** The system must allow the owner to view tool-requests sent to their posts
* **Fit Criterion:** a screen with all the requests will be displayed
* **Rationale:** For the owner to accept it and rent the tool to the renter
* **Dependencies:** FR2, FR11, FR16, FR21
* **Priority:** High
* **Support materials:** None
* **History:** Created 4/3/21

Edited 9/3/21

* **Type**: Owner requirement

### 3.2.18. Functional requirement 18

* **ID:** FR18
* **BUC:** UC3
* **Description:** The system must allow the owner to accept or reject a tool-request
* **Fit Criterion:** the request will be accepted and the tool marked as unavailable. Or the request will be rejected and the tool-request will be deleted.
* **Rationale:** To start renting the tool to the renter. Or reject a renter’s offer
* **Dependencies:** FR2, FR11, FR16, FR17, FR21
* **Priority:** High
* **Support materials:** Figure 2,Figure 1
* **History:** Created 4/3/21

Edited 9/3/21

* **Type**: Owner requirement

### 3.2.19. Functional requirement 19

* **ID:** FR19
* **BUC:** UC3
* **Description:** The system must be able to open a chat room between the owner and renter
* **Fit Criterion:** a chat room will be open where the owner and renter can send and receive messages from each other
* **Rationale:** For the owner and renter to communicate
* **Dependencies:** FR1, FR2
* **Priority:** Low
* **Support materials:** None
* **History:** Created 4/3/21
* **Type**: User requirement

### 3.2.20. Functional requirement 20

* **ID:** FR20
* **BUC:** UC5, UC6
* **Description:** The system must be able to determine if a user is authorized to send, edit, or remove a tool-request to a post.
* **Fit Criterion:** The user's request to add, edit or delete a tool-request will be denied if (s)he is not authorized to
* **Rationale:** To prevent unauthorized access to a tool-request
* **Dependencies:** FR2, FR4
* **Priority:** High
* **Support materials:** None
* **History:** Created 5/3/21
* **Type**: Renter requirement

### 3.2.21. Functional requirement 21

* **ID:** FR21
* **BUC:** UC5
* **Description:** The system must allow the renter to send a tool-request to a post.
* **Fit Criterion:** a tool-request screen will be displayed
* **Rationale:** For the owner to accept it and rent the tool to the renter
* **Dependencies:** FR2, FR11, FR20
* **Priority:** High
* **Support materials:** None
* **History:** Created 5/3/21

Edited 9/3/21

* **Type**: Renter requirement

### 3.2.22. Functional requirement 22

* **ID:** FR22
* **BUC:** UC6
* **Description:** The system must allow the renter to edit or remove his/her tool-request
* **Fit Criterion:** a screen to edit the request will be displayed with a button to delete the tool-request
* **Rationale:** To allow the renter to change his tool-request details or delete the request entirely
* **Dependencies:** FR2, FR20, FR21
* **Priority:** High
* **Support materials:** None
* **History:** Created 5/3/21

Edited 9/3/21

* **Type**: Renter requirement

### 3.2.23. Functional requirement 23

* **ID:** FR23
* **BUC:** UC10
* **Description:** The system must be able to determine if two users had a previous rental-transaction.
* **Fit Criterion:** the system will confirm if two users had a previous transaction or not.
* **Rationale:** To allow them to rate each other
* **Dependencies:** FR2
* **Priority:** Low
* **Support materials:** None
* **History:** Created 5/3/21
* **Type**: User requirement

### 3.2.24. Functional requirement 24

* **ID:** FR24
* **BUC:** UC10A
* **Description:** The system must allow the user to add/edit a review of another user if they had a previous transaction
* **Fit Criterion:** A form to submit the review details will be displayed
* **Rationale:** To rate his/her experience with the other user
* **Dependencies:** FR1, FR2, FR23
* **Priority:** Low
* **Support materials:** None
* **History:** Created 5/3/21
* **Type**: User requirement

### 3.2.25. Functional requirement 25

* **ID:** FR25
* **BUC:** UC10B
* **Description:** The system must be able to determine if a user is authorized to delete a review
* **Fit Criterion:** the delete request will be denied if the user isn't authorized
* **Rationale:** To prevent an unauthorized user from deleting a review
* **Dependencies:** FR2, FR24
* **Priority:** Low
* **Support materials:** None
* **History:** Created 5/3/21
* **Type**: User requirement

### 3.2.26. Functional requirement 26

* **ID:** FR26
* **BUC:** UC10B
* **Description:** The system must allow the user to delete a review.
* **Fit Criterion:** an option (button) to delete the review will be displayed
* **Rationale:** To remove the review from the user's profile
* **Dependencies:** FR2, FR24, FR25
* **Priority:** Low
* **Support materials:** None
* **History:** Created 5/3/21
* **Type**: User requirement

### 3.2.27. Functional requirement 27

* **ID:** FR27
* **BUC:** UC4
* **Description:** The system must allow the user to take pictures and videos.
* **Fit Criterion:** The camera screen will be displayed
* **Rationale:** To take pictures and videos of a tool
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** Figure 3
* **History:** Created 9/3/21
* **Type**: User requirement

### 3.2.28. Functional requirement 28

* **ID:** FR28
* **BUC:** UC4
* **Description:** The system must allow the user to upload pictures and videos.
* **Fit Criterion:** The pictures and videos will be uploaded to the server.
* **Rationale:** to upload pictures and videos of a tool
* **Dependencies:** FR27
* **Priority:** High
* **Support materials:** Figure 3
* **History:** Created 9/3/21
* **Type**: User requirement

### 3.2.29. Functional requirement 29

* **ID:** FR29
* **BUC:** UC4
* **Description:** The system must be able to send and receive payment
* **Fit Criterion:** The money will be added/deducted to/from the system's account balance.
* and vice versa for the user's account.
* **Rationale:** To send money between users and from the system to the users
* **Dependencies:** FR4
* **Priority:** High
* **Support materials:** None
* **History:** Created 9/3/21
* **Type**: System requirement

### 3.2.30. Functional requirement 30

* **ID:** FR30
* **BUC:** UC1,2,3,5, and 6
* **Description:** The system must be able to display a prompt to the user
* **Fit Criterion:** The prompt will be displayed
* **Rationale:** To inform the user of specific information
* **Dependencies:** -
* **Priority:** High
* **Support materials:** None
* **History:** Created 9/3/21
* **Type**: User requirement

### 3.2.31. Functional requirement 31

* **ID:** FR31
* **BUC:** UC7
* **Description:** The system must be able to create a disagreement case. It's created any time the user and owner have a disagreement. And it needs to be reviewed by the admin.
* **Fit Criterion:** The case will be created in the database where it will be available to the admin
* **Rationale:** For the admin to review the problem and arrive to a decision
* **Dependencies:** FR2
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21
* **Type**: System requirement

### 3.2.32. Functional requirement 32

* **ID:** FR32
* **BUC:** UC7
* **Description:** The system must allow the admin to review a disagreement case and arrive to a decision.
* **Fit Criterion:** A screen for the disagreement case will be displayed and it contains a from to fill the decision
* **Rationale:** So, the admin can settle and fix the problem
* **Dependencies:** FR2
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21
* **Type**: Admin requirement

### 3.2.33. Functional requirement 33

* **ID:** FR33
* **BUC:** UC7
* **Description:** The system must allow the owner to claim the tool is damaged.
* **Fit Criterion:** An option or a button will be available to start the claim.
* **Rationale:** To prevent the renter from damaging the tool without paying for it
* **Dependencies:** FR2
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21

Edited 9/3/21

* **Type**: Owner requirement

### 3.2.34. Functional requirement 34

* **ID:** FR34
* **BUC:** UC7
* **Description:** The system must be able to calculate the total money needed from/to a user. Including the remaining of the insurance money.
* **Fit Criterion:** The total will be calculated and displayed
* **Rationale:** To send the appropriate amount of money to each the owner and renter
* **Dependencies:** -
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21
* **Type**: System requirement

### 3.2.35. Functional requirement 35

* **ID:** FR35
* **BUC:** UC4, UC7
* **Description:** The system must allow the user to upload documents.
* **Fit Criterion:** A file upload screen will be displayed
* **Rationale:** To upload any official documents
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21
* **Type**: User requirement

### 3.2.36. Functional requirement 36

* **ID:** FR36
* **BUC:** UC7
* **Description:** The system must allow the renter and owner to agree on a compensation price
* **Fit Criterion:** The agreement price screen will be displayed
* **Rationale:** So, the owner and renter can agree on a middle ground
* **Dependencies:** FR1, FR2
* **Priority:** High
* **Support materials:** Figure 4
* **History:** Created 9/3/21
* **Type**: User requirement

### 3.2.37. Functional requirement 37

* **ID:** FR37
* **BUC:** None
* **Description:** The system must allow the user to search for a tool post.
* **Fit Criterion:** A search bar will be displayed.
* When the user searches, a list of posts that match the search query will be displayed
* **Rationale:** To find a post of a specific tool so the renter can send a tool-request to.
* **Dependencies:** FR1
* **Priority:** High
* **Support materials:** None
* **History:** Created 9/3/21
* **Type**: User requirement

## 3.3 Non-Functional Requirements

### 3.3.1 Look and feel requirements

#### 3.3.1.1 Material design

* **ID:** LFR1
* **Description:** The system must follow the material design guidelines.
* **History:** Created 4/4/2021
* **Type:** Look and feel

#### 3.3.1.2 Colors

* **ID:** LFR2
* **Description:** The system design colors must rely mainly on the two primary colors of the brand, while following LFR1.
* **History:** Created 4/4/2021
* **Type:** Look and feel requirement

### 3.3.2 Usability and humanity requirements

#### 3.3.2.1 Ease of use

* **ID:** UR1
* **Description:** The system must be easy to use and learn by different users.
* **History:** Created 4/4/2021
* **Type:** Usability and humanity requirement

#### 3.3.2.2 Localization

* **ID:** UR2
* **Description:** The system must be available in Arabic and English
* **History:** Created 4/4/2021
* **Type:** Usability and humanity requirement

### 3.3.3 Performance requirements

#### 3.3.3.1 Load handling

* **ID:** PR1
* **Description:** The system must handle up to 500 users simultaneously
* **History:** Created 4/4/2021
* **Type:** Performance requirements

#### 3.3.3.2 Inputs respond time

* **ID:** PR2
* **Description:** The system must respond to most inputs within 2 seconds
* **History:** Created 4/4/2021
* **Type:** Performance requirements

#### 3.3.3.3 Complex respond time

* **ID:** PR3
* **Description:** The system must respond to complex requests within 20 seconds
* **History:** Created 4/4/2021
* **Type:** Performance requirements

#### 3.3.3.4 Availability

* **ID:** PR4
* **Description:** The system must be available at all time.
* **History:** Created 4/4/2021
* **Type:** Performance requirements

### 3.3.4 Maintainability and support requirements

#### 3.3.4.1 Future Expansions

* **ID:** MR1
* **Description:** The system must be able to cope with expansions to global availability.
* **History:** Created 9/4/2021
* **Type:** Maintainability and support requirement

#### 3.3.4.2 Extra localization plans

* **ID:** MR2
* **Description:** The system must be able to add support for different languages.
* **History:** Created 4/4/2021
* **Type:** Maintainability and support requirement

### 3.3.5 Legal requirements

#### 3.3.5.1 Law abiding system

* **ID:** LR1
* **Description:** The system must operate in accordance with the Saudi Arabia’s laws.
* **History:** Created 4/4/2021
* **Type:** Legal requirement

## 3.4 Analysis

### 3.4.1 Constraints on objects attributes

#### 3.4.1.1 Constraints on User objects:

* **Context** User **inv:**

self.uid == self.ownedTool.ownerUID

* **Context** User **inv:**

self.uid == self.request.renterID

* **Context** User **inv:**

self.uid <> Self.request.tool.ownerUID

* **Context** User **inv:**

self.rating >= 0 **AND** self.rating <= 5

#### 3.4.1.2 Constraints on ToolsRequest objects:

* **Context** ToolsRequest **inv:**

Self.isAccepted == true **implies** self.id == self.tool.acceptedRequestID

* **Context** ToolsRequest **inv:**

self.renterUID <> self.tool.ownerUID

* **Context** ToolsRequest **inv:**

self.numOfDays > 0

#### 3.4.1.3 Constraints on Rent objects:

* **Context** Rent **inv**:
* self.toolID == self.request.toolID
* **Context** Rent **inv**:
* self.tool == self.request.tool

* **Context** Rent **inv**:
* self.endTime **after** self.startTime

#### 3.4.1.4 Constraints on UserReview objects:

* **Context** UserReview **inv**:
* self.creatorUID <> self.targetUID
* **Context** UserReview **inv**:
* self.value >= 0 **AND**  self.value <= 5

#### 3.4.1.5 Constraints on Tool objects:

* **Context** Tool **inv**:

self.requests **include** self.acceptedRequest

* **Context** Tool **inv:**

self.owner.tools **include** self

### 3.4.2 System Operations

Below are the main system operations associated with the system classes

#### 3.4.2.1 User.createPost()

**Context** User:: createPost(name: String, description: String, rentPrice: Double, insuranceAmount: Double, media: List<String>, location: String)

**Pre:**

* name.length > 0
* rentPrice > 0
* insuranceAmount > 0
* location.length > 0

**Post:**

* a new Tool object will be created.
* the user will be linked with the Tool object.
* if the operation succeeded the Tool object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.2 Tool.edit()

**Context** Tool::edit(name: String, description: String, rentPrice: Double, insuranceAmount: Double, media: List<String>, location: String)

**Pre**:

* name.length > 0
* rentPrice > 0
* insuranceAmount > 0
* location.length > 0

**Post**:

* the Tool object will be edited.
* if the operation succeeded the Tool object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.3 Tool.acceptRequest()

**Context** Tool::acceptRequest(toolRequestID)

**Pre**:

* Tool.requests must include toolRequestID.
* Tool.isAvailable == true

**Post**:

* Tool.isAvailable will become false
* Tool.acceptedRequestID will become toolRequest.id
* A chat room will be created
* The Tool owner will be linked with the chat room
* The ToolRequest sender (renter) will be linked with the chat room
* The ToolRequest sender (renter) will be notified
* if an error occurred a string indicating the error will be returned.

#### 3.4.2.4 Tool.startRent()

**Context** Tool :: startRent()

**Pre**:

* The tool must have an accepted tool-request (i.e., Tool.isAvailable == false)

**Post**:

* A new Rent object, rent, will be created as follows:
  + rent.toolID will become tool.ID
  + rent.requestID will become tool.acceptedRequestID
  + rent.startTime will become DateTime.now()
* tool.currentRent will become *rent*
* tool.acceptedToolRequest.isRented will become true
* The Rent object will be linked with the Tool
* The Rent object will be linked with the ToolRequest
* if the operation succeeded the Rent object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.5 Tool.addRequest()

**Context** Tool:: addRequest(renterUID: String, toolID: String, numOfDays: Integer, rentPrice: Double, insuranceAmount: Double)

**Pre**:

* the renter must not have a previous request on the tool.
* renterUID is a valid user ID
* toolID is a valid tool ID
* toolID doesn’t reference a tool where tool.ownerID == renterID (i.e., a user can’t request his/her own tool)
* numOfDays > 0
* rentPrice > 0
* insuranceAmount > 0

**Post**:

* a new ToolRequest object will be created.
* The ToolRequest and tool will be linked
* The object will be linked with the renter
* The object will be linked with the Tool object
* if the operation succeeded the ToolRequest object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.6 ToolRequest.edit()

**Context** ToolRequest:: edit(numOfDays: Integer, rentPrice: Double, insuranceAmount: Double)

Pre:

* numOfDays > 0
* rentPrice > 0
* insuranceAmount > 0

**Post**:

* the ToolRequest object will be edited.
* if the operation succeeded the ToolRequest object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.7 Tool.endRent()

**Context** Tool:: endRent()

**Pre**:

* currentRent <> null (i.e., there must be a rent in progress)

**Post**:

* tool.currentRent will be deleted and become null.
* tool.isAvailable will become true
* tool.acceptedToolRequest will be deleted.
* tool.acceptedRequestID will become null
* if an error occurred a string indicating the error will be returned.

#### 3.4.2.8 User.addReview()

**Context** User:: addReview(value: Double, description: String)

**Pre**:

* user.hadAPreviousRentWith() == true
* value >= 0 AND value <= 5 (i.e., value must be between 0 and 5)

**Post**:

* a new Review object, review, will be created
* review will be linked to user
* review will be linked to target
* if the operation succeeded the Review object’s document-reference will be returned.
* otherwise (if an error occurred) a string indicating the error will be returned.

#### 3.4.2.9 User.ban()

**Context** User:: ban()

**Pre**:

* initiator.isAdmin == true (i.e., the initiator must be an administrator)

**Post**:

* user.IDNumber will be added to the banned list
* user.emailAddress will be added to the banned list
* any tool-requests from the user will be delete

### 3.4.3 Analysis class diagram



Figure - Analysis class diagram

## 3.5 Glossary

|  |  |  |
| --- | --- | --- |
| **Term** | **Category** | **Definition** |
| Owner | Role | The user that is offering his tool for rent. |
| Renter | Role | The user that is requesting to rent an owner’s tool. |
| Admin | Role | The system administrator. (S)He has access to all functions and operations. |
| Post | Term | An online page for a *Tool*, where users can find it and send tool-requests on it. |
| User | Concept | A person that is using the system, includes *Owner, Renter,* and *Admin.* |
| Tool | Concept | A tool that a *User* has offered for rent. |
| ToolRequest | Concept | A request from the renter to rent the owner’s tool. Where the owner can reject or accept it. |
| Rent | Concept | A renting process that is created after delivering a *Tool* to the *Renter*. |
| UserReview | Concept | A review/rating from a *User* to another *User.* |
| Ban a *User* | Operation | Prevent a *User* from accessing the system functionalities and operations |

### 3.5.2 Glossary of class attributes

|  |  |  |
| --- | --- | --- |
| **Term** | **Category** | **Definition** |
| uid | Attribute of *User* | User ID: a unique identifier for a *User.* |
| name | Attribute of *User* | The name of a *User*. |
| emailAddress | Attribute of *User* | The email address of a *User.* |
| IDNumber | Attribute of *User* | The ID number of a *User.* |
| creditCard | Attribute of *User* | Credit card details of a *User.* |
| rating | Attribute of *User* | A value that rates a *User* out of 5, it's the average of all *User.reviews* |
| reviews | Attribute of *User* | A collection of *UserReview* objectsthat rates a *User.* |
| tools | Attribute of *User* | A collection of *Tool* objects that belong to the *User*. |
| requests | Attribute of *User* | A collection of *ToolRequest*. |
| id | Attribute of *Tool* | A unique identifier for a *Tool*. |
| ownerUID | Attribute of *Tool* | A unique identifier for the *User* that owns a *Tool* (i.e., the *Owner.uid*). |
| name | Attribute of *Tool* | The name of a *Tool*. |
| description | Attribute of *Tool* | A description of a *Tool.* |
| rentPrice | Attribute of *Tool* | A *Tool* rent price for one day. |
| insuranceAmount | Attribute of *Tool* | An insurance amount of a *Tool* that is held on the *Renter*'s credit card until the *Tool* is returned in a good condition. |
| media | Attribute of *Tool* | A List of URLs of pictures and/or videos of a *Tool.* |
| location | Attribute of *Tool* | The country and city of a *Tool.* |
| isAvailable | Attribute of *Tool* | An indicator if a *Tool* is available for rent or not. If not, no *ToolRequest* can be accepted. |
| acceptedRequestID | Attribute of *Tool* | A unique identifier for the accepted *ToolRequest* on a *Tool.* |
| currentRent | Attribute of *Tool* | A *Rent* object that is created after delivering the *Tool* to the *Renter*. |
| requests | Attribute of *Tool* | A collection of *ToolRequest*s that are sent to a *Tool.* |
| id | Attribute of *ToolRequest* | A unique identifier for a *ToolRequest*. |
| renterUID | Attribute of *ToolRequest* | A unique identifier for the *User* that sent a *ToolRequest* (i.e., the *Renter.uid*). |
| toolID | Attribute of *ToolRequest* | A unique identifier for the *Tool* a *ToolRequest* is sent to. |
| numOfDays | Attribute of *ToolRequest* | The number of days a *Renter* want to rent the tool for. |
| rentPrice | Attribute of *ToolRequest* | A *Tool* rent price for one day. |
| insuranceAmount | Attribute of *ToolRequest* | An insurance amount of a *Tool* that is held on the *Renter*'s credit card until the *Tool* is returned in a good condition. |
| isAccepted | Attribute of *ToolRequest* | An indicator if a *ToolRequset* is accepted or not. |
| isRented | Attribute of *ToolRequest* | An indicator if a *ToolRequset* is accepted and the *Tool* is delivered and a *Rent* has started. |
| toolID | Attribute of *Rent* | A unique identifier for a *Tool* linked to a *Rent*. |
| requestID | Attribute of *Rent* | A unique identifier for a *ToolRequest* linked to a *Rent.* |
| startTime | Attribute of *Rent* | the start date and time of a *Rent.* (i.e., the date and time of delivering the *Tool*). |
| endTime | Attribute of *Rent* | the end date and time of a *Rent.* (i.e., the date and time of returning the *Tool*). |
| creatorUID | Attribute of *UserReviews* | a unique identifier for a *User* that is creating a *UserReview*. |
| targetUID | Attribute of *UserReviews* | a unique identifier for a *User* that a *UserReview* is rating/targeting. |
| value | Attribute of *UserReviews* | A value that rates a *User* out of 5, it's the average of all *User.reviews*. |
| description | Attribute of *UserReviews* | A description of a *UserReview*. |

# Table of Figures

[Figure 1 - use case diagram 4](#_Toc69322480)

[Figure 2 - Activity diagram of accepting/rejecting tool-requests 14](#_Toc69322481)

[Figure 3 - Activity diagram of delivering a tool 15](#_Toc69322482)

[Figure 4 - Activity diagram of returning a tool 16](#_Toc69322483)

[Figure 5 - The Conceptual model 17](#_Toc69322484)

[Figure 6 - Analysis class diagram 36](#_Toc69322485)

# References

HSS, 2021. *FAQ - HSS Hire.* [Online]   
Available at: https://www.hss.com/hire/faqs  
[Accessed 5 April 2021].

National Association of REALTORS, 2019. *Remodeling Impact Report: D.I.Y.,* s.l.: National Association of REALTORS.

Sparetoolz, 2021. *Sparetoolz FAQ.* [Online]   
Available at: https://sparetoolz.com/pages/faq  
[Accessed 5 April 2021].

United Rentals, 2021. *Equipment Rentals | United Rentals.* [Online]   
Available at: https://www.unitedrentals.com/marketplace/equipment#/  
[Accessed 5 April 2021].