METROPOLITAN STATE UNIVERSITY

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

FINAL YEAR PROJECT REPORT

TOOL RENTAL SYSTEM

Samara Garrett

Derrick Harper

Hana Biadgilgn

Andrew Ortiz

Shannon Fisher

April 2020

"I hereby declare that I have read this project report and in my opinion this report is sufficient in terms of scope and quality for the award of the degree of "Bachelor of Science in Computer Science"

Signature :

Supervisor :

Date : 4/29/2020

FINAL YEAR PROJECT REPORT

TOOL RENTAL SYSTEM

Samara Garrett
Derrick Harper
Hana Biadgilgn
Andrew Ortiz
Shannon Fisher

A report submitted in partial fulfillment of the requirements for the award of the Bachelor Of Science In "Computer Science"

College of Science,
Computer Science and Cybersecurity

April 2020

DECLARATION

I declare that this thesis entitled “Tool Rental System” is the result of my/our own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Name : Samara Garrett
Derrick Harper
Hana Biadgilgn
Andrew Ortiz
Shannon Fisher

Date : 4/29/2020

ACKNOWLEDGEMENT

We would like to thank the Metropolitan State University staff and administration team, and the Computer Science department chair, faculty, and advisors for all their support towards helping us achieve our higher education goals. We would also like to take this opportunity to thank our family, friends, and employers for their continued support, understanding, and flexibility during this challenging time. Plenty of sacrifices through countless hours of hard work were made to reach our goals, and to those most dear to us, it has not gone unappreciated. Thank you!

ABSTRACT

Our goal in this project was to implement a system that will maintain a database of product and customer information for a tool rental shop. Our system will allow an administrative user to not only log in and rent out tools but also view and maintain records regarding said tools and their corresponding rentals. Our team ultimately set out to construct a web application that will serve as a mechanism with which a shop can keep all of their data online as opposed to a messy paper filing method.

Over the course of this project our research has taken us not only to certain hardware stores but also to the websites of companies that already have a similar mechanism implemented. Our purpose in doing that was to see ways we might improve on said mechanisms. Our prototype has evolved from what was presented in iteration 1 to what we now present as our final project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

One thing that is not desired when it comes to home improvement is buying tools that is expected to be used only once or twice. For example, if you want to install flooring in one of the rooms in your house you would need to purchase a specific and specialty tools. These tools could cost the home owner a great deal of money. Unless otherwise, one is constantly installing flooring, the tools would end up consuming space and sitting in a garage or apartment once finished with the job. In order to alleviate this problem, we set out to create a website that can be used by any small businesses and allowing home owners to rent tools.

Having the ability to rent tools, rather than buy them outright, would be a cost-effective way to do projects for the home owner with added benefit of space saving. This also give the flexibility of setting pricing and managing customers to the small business, enabling it to offer lower price than big department rental stores. This creates an opportunity for the small business to attract more customers while saving money for the tool renters.

2.0 VISION AND BUSINESS CASE AND SCOPE

2.1 VISION

To create a website that enables anyone who needs simple home improvement tools to be able to rent them, and helps a business rent those tools out to customers effectively.

This tool renting website will allow people to look for available tools and reserve those tools from the convenience of their home. It will also help the tool rental business to keep track of rentals.

2.2 BUSINESS CASE

Converting an on-paper tool-rental system to a web application simplifies the process of record keeping and rental. It can also cut down on previous record keeping labor costs and storage space.

For home improvement projects homeowners may not want to buy a tool that they will only use once, so this business fills a niche to provide tools for single or occasional use.

This can also help the homeowner save money and storage space as well as help the environment as less tools bought means less tools being produced.

2.2.1 Cost Analysis

In order to properly vet the tools and solutions that are required to run and implement our Tool Rental System, we had to dive deep into the nuances of our system and how it could be implemented. There are hundreds, if not thousands, of embedded solutions that could predetermine configuration and UI for our web app. The first solution that we explored was a solution presented by Microsoft Azure. Azure seemed like a viable option to explore based the fact that it can deploy a full stack implementation of our application concept. By “full stack” we mean, a solution that could power, host, and construct the main components of our app from top level UI all the way down to programming language and database design. The specific solution that would be pertinent to our app would include use of tools: Ruby, PostgreSQL, and Azure hosted web-

domains. The only problem, the pricing structure in which Azure maintains, did not fit the practicableness of our project. While Azure was the first juncture in our thought process to consider, another feasible option presented itself, IBM Watson. Watson is another solution that also offers “fully baked” solutions that can be implemented in a top-down fashion with an all encapsulating infrastructure. The only difference that must be noted between the two solutions is that the instantiation of database storage, within the Watson atmosphere, is dynamic - and perfect for scaling our app based on frequency of use and scope of project.

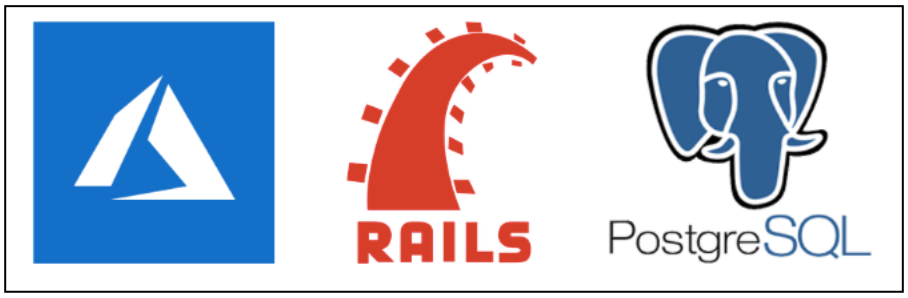


Figure 2.1: Azure All In One Solution for Ruby on Rails

Figure 2.2: Solution Vetted by Azure Standards

Description	1 month	12 months	24 months	36 months
1TB or 1TB increment Standard Performance Tier	$100	$1,080	$2,160	$3,240
1TB or 1TB increment, Premium Performance Tier	$200	$2,160	$4,320	$6,480
1TB or 1TB increment, Extreme Performance Tier	$300	$3,240	$6,480	$8,720

Figure 2.3: Microsoft Azure Pricing Structure

VCORE	MEMORY	PAY AS YOU GO	ONE YEAR RESERVED (% SAVINGS)
2	10	~$127.896/month	~$77.672/month (~39%)
4	20	~$255.792/month	~$155.344/month (~39%)
8	40	~$511.584/month	~$310.688/month (~39%)
16	80	~$1,023.168/month	~$621.376/month (~39%)
32	160	~$2,046.336/month	~$1,242.752/month (~39%)
64	320	~$4,092.672/month	~$2,485.504/month (~39%)

Figure 2.4: IBM Watson Pricing Structure

Figure 2.5: Website Hosting Standard Pricing

Recurring Monthly	Cost Associated	Monthly Total
Salary	5 x 5000	$25,000
Hosting/SiteGround	$3.95	
		$25,004

Start Costs	Cost Associated	Total
Hardware/Housing	$99-$749	$399

Total Evaluation of Initial Sum to Instantiate & Develop	$25,402.95

Table 2.6: Cost Analysis Table

2.3 SCOPE

A three-tier web application that will allow users to view, rent, and return tools.

3.0 FEASIBILITY STUDIES

3.1 PROJECT DESCRIPTION

The Tool Rental System is a multi-page web application that has the elasticity to be used in large scale operations as well as small-midsized organizations. Users, Client and Admin, have the ability to log in to Tool Rental System and perform certain actions. Depending who you are, there are different permissions ranging in functionality. Clients can rent tools, view tools, among other consumer actions. Admin has all of the functionality of client with added authorizations like viewing previous rentals and viewing currently rented tools.

3.2 TECHNOLOGY

Technology in this day drives success. Upon analysis of other preexisting web tool rental software models, the Tool Rental System is scalable enough for any sized organization to utilize the platform with fluidity. The Tool Rental System uses modern coding languages and frameworks; specifically, ASP.NET. It has implemented an AWS RDS database that is fully embedded, not seen by user, and proves the differentiation of possible business fulfillments with companies ranging in size.

3.3 PRODUCT MARKETPLACE

As referenced, our marketplace will focus solely on the commercial software side of things. We want to target all ranges of business who offer a rental solution on one's tool inventory. As commercial software distributors, working with shareholders and key executives to optimize use cases and functionality per organization's internal makeup, is crucial.

4.0 METHODOLOGY

Agile development is a tool that uses iterative approach and it is mainly based on a team work, whose main objective is to deliver application with complete and functional components. It uses a sprint approach. Agile approach involves project initiation, sprint planning, demos, if the project isn't completed by the time intended or there are changes that need to take place, agile makes it possible to reprioritize and be included in future cycle. This application will utilize the agile software development methodology. Focus will be on the team members performing the work and how they work together. This will allow for robust collaboration between self-organizing cross functional teams using the appropriate practices for the context.

5.0 SYSTEM DESIGN

5.1 SYSTEM

5.1.1 USE CASE DIAGRAM

Use cases describe the expected behavior of a system, and a use case diagram describes the relationships between use cases, actors (users of the system), and the system itself. The following use case diagram details the functional requirements of the Tool Rental System.

The Tool Rental System has three actors, New Client, Old Client, and Admin, of which New Client and Old Client are subclasses of the superclass Client. Both Admins and Old Clients can manage their accounts when logged in. Many use cases include a confirmation screen to make it clear to the actor that the change successfully went through.

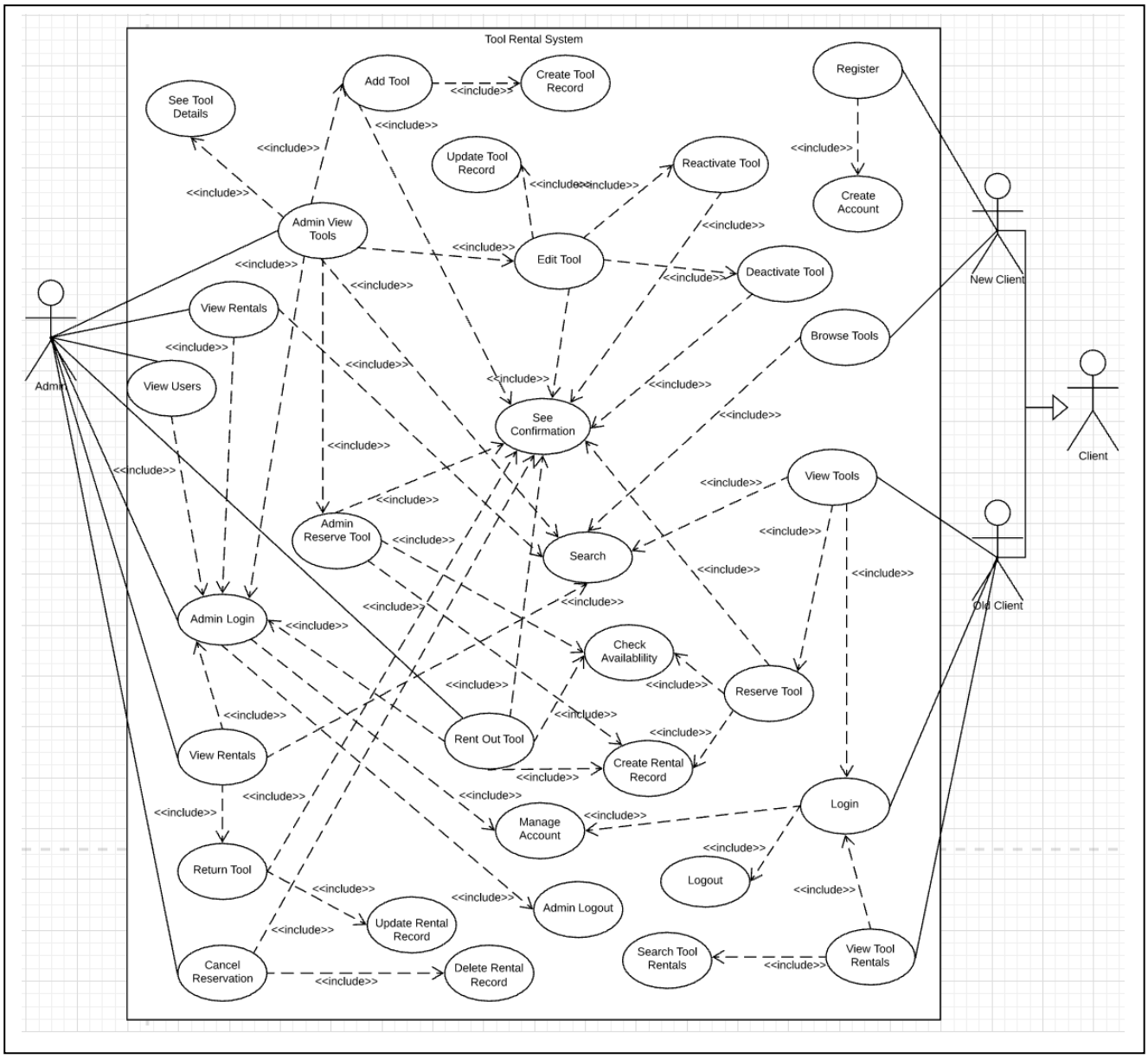


Figure 5.1: Use Case Diagram

5.1.1.1 SYSTEM CLIENT USER

New Clients can browse tools, which includes searching, and then register which turns them from a New Client (no account) to an Old Client (has an account).

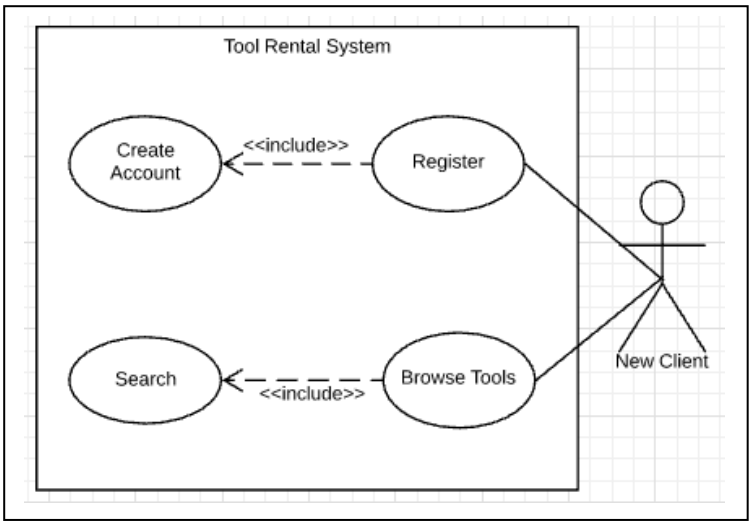


Figure 5.2: Use Case Diagram for New Client

Old Clients can login (and logout) and then view tools, which includes searching, and reserve tools from those available, which creates a rental record. Old Clients can also view and search their tool rentals, which includes current rentals, past rentals, and current reservations.

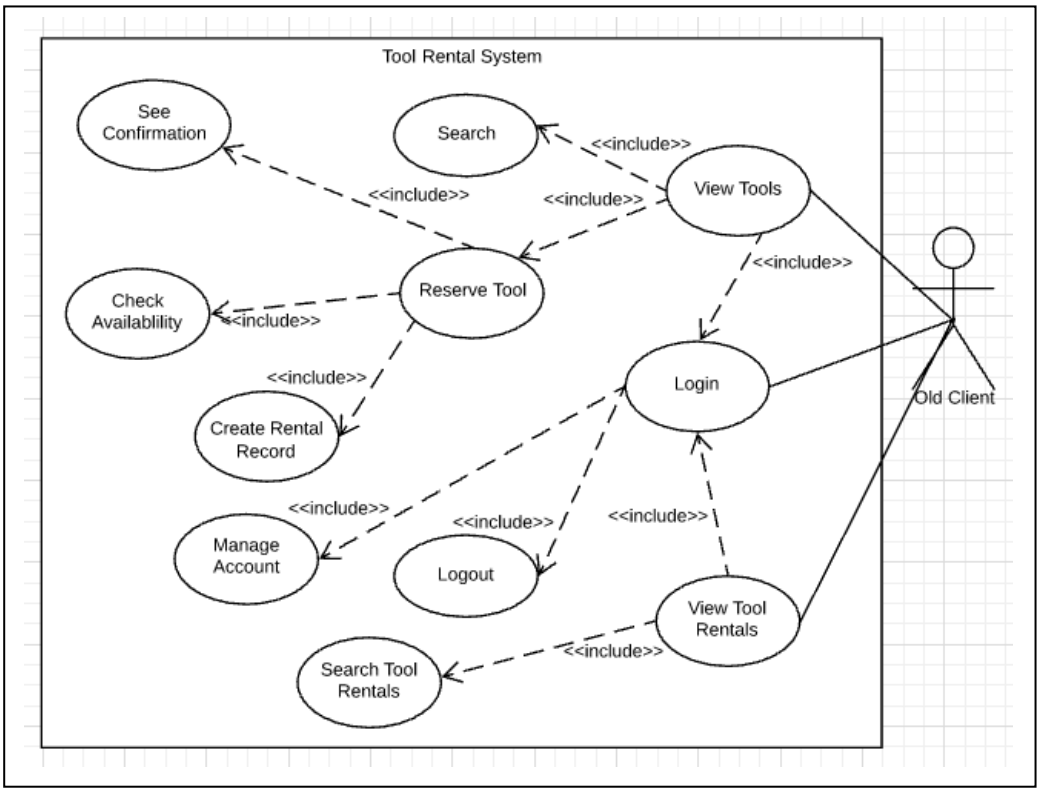


Figure 5.3: Use Case Diagram for Old Client

5.1.1.2 SYSTEM ADMIN USER

Admins have access to a much broader range of functionality when logged in. They can view tools, which allows them to see more of the tools' parameters than Clients can see, and then view each tool in more detail. They can add tools, which creates a tool record, edit tools, which updates tool records, and deactivate and reactivate tools. Deactivated tools cannot be rented out or reserved and are not visible to Clients through browse or view. Admins can reserve and rent out available tools to users, which creates a rental record. They can view rentals and return them, which updates the rental record. They can also cancel reservations, which deletes that reservation's rental record.

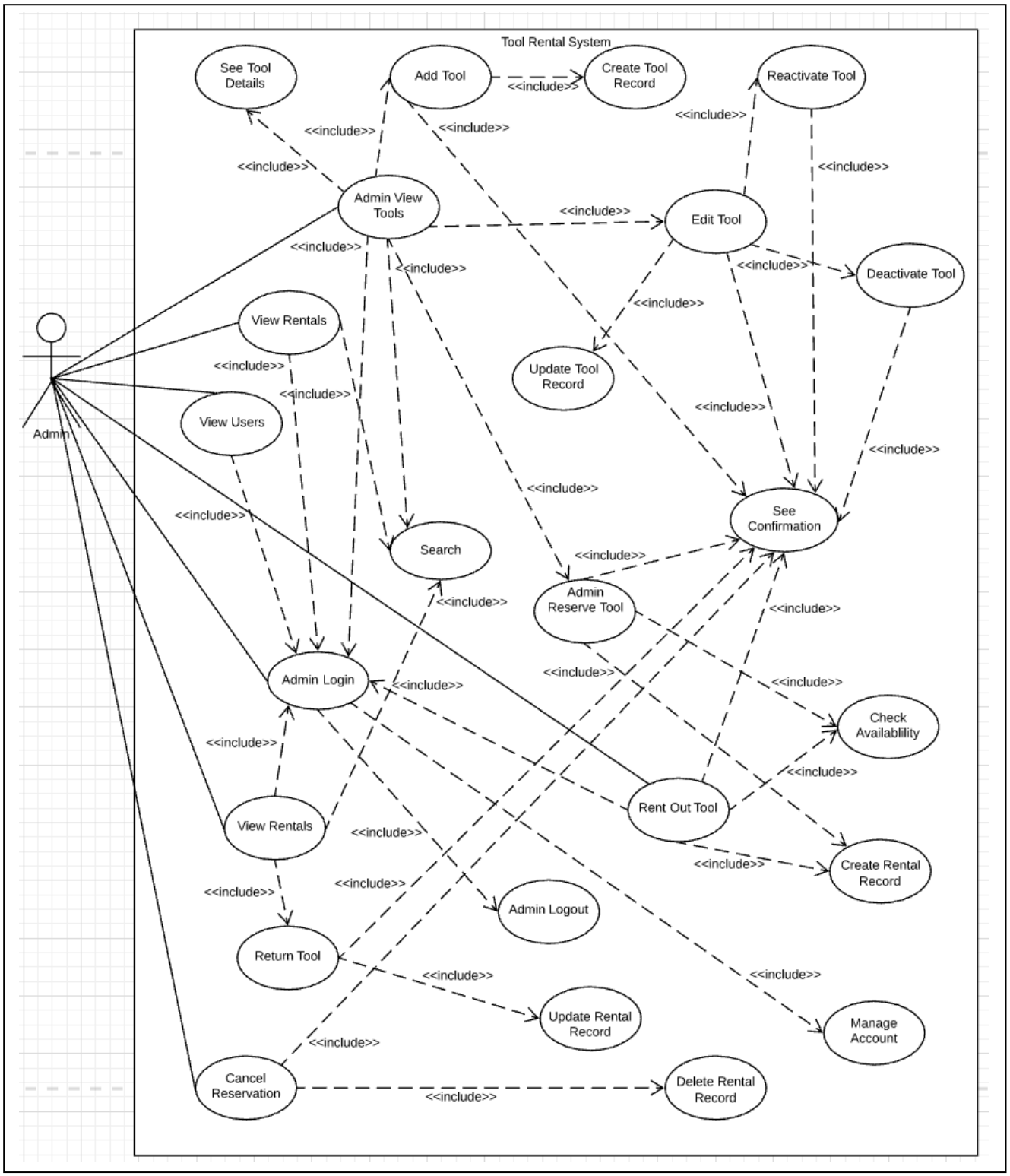


Figure 5.4: Use Case Diagram for Admin

5.1.2 SEQUENCE DIAGRAM

5.1.2.1 SYSTEM CLIENT USER

Our new client diagram documents a restricted set of functionalities that are available to users who have not yet registered in our system. These users may view a list of our available tools and of course register to our system and create an account so they can be privy to the old user luxuries.

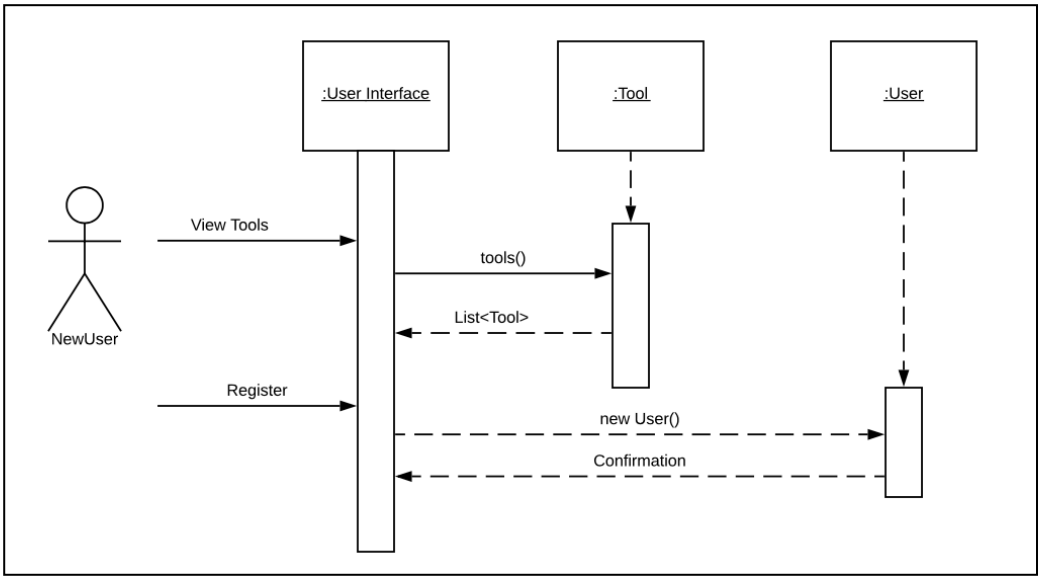


Figure 5.5: Sequence Diagram for New Client

Our client/old user diagram shows the sequence of events that make up the restricted capabilities of a client/old user on our system. These users are also able to log in and out of the system as well as view a list of tools that are available to be rented. Once they find their desired tool they are able to reserve said tool for pick up. And if a user needs a record of their previous rentals there is a mechanism implement to allow for that and it is also documented on this diagram.

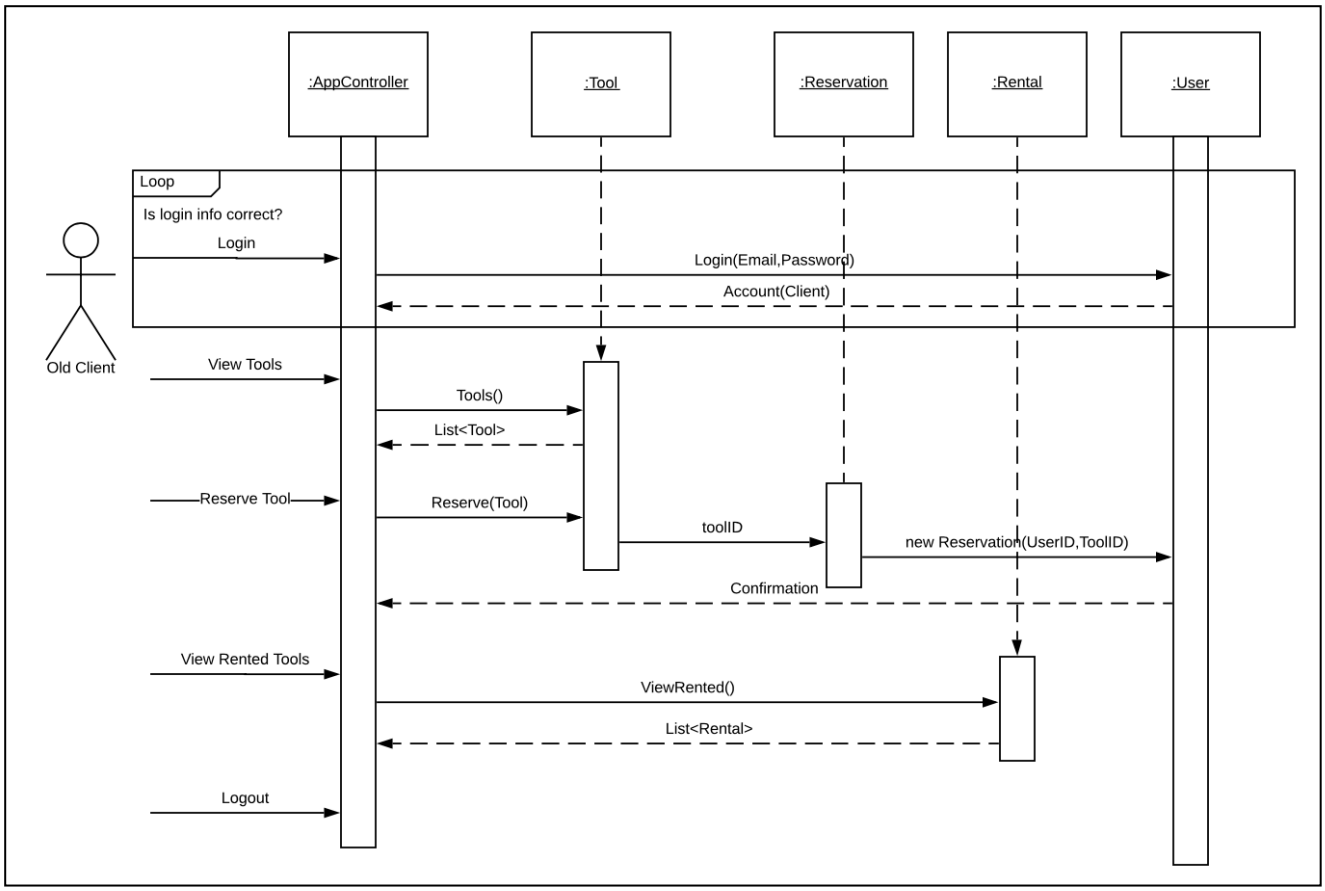


Figure 5.6: Sequence Diagram for Old Client

5.1.2.2 SYSTEM ADMIN USER

Our sequence diagram for our admin user documents the sequence of events that are able to occur in the event that an admin user logs into our system. He can first login. This process causes the system to have an interaction with the user account that corresponds to that particular admin. Provided he has given the correct login information he will then be able to access implementations of our use cases that are restricted to administrative access. One is able to add tools to the system, edit them, remove them from service by deactivating them, and return them to service by reactivating them. This diagram also documents the processes that take place when our administrative user wants to rent out, return, or manage reservations of the tools in the system. The admin's capability to view full sets of the tools and users are also present in this diagram.

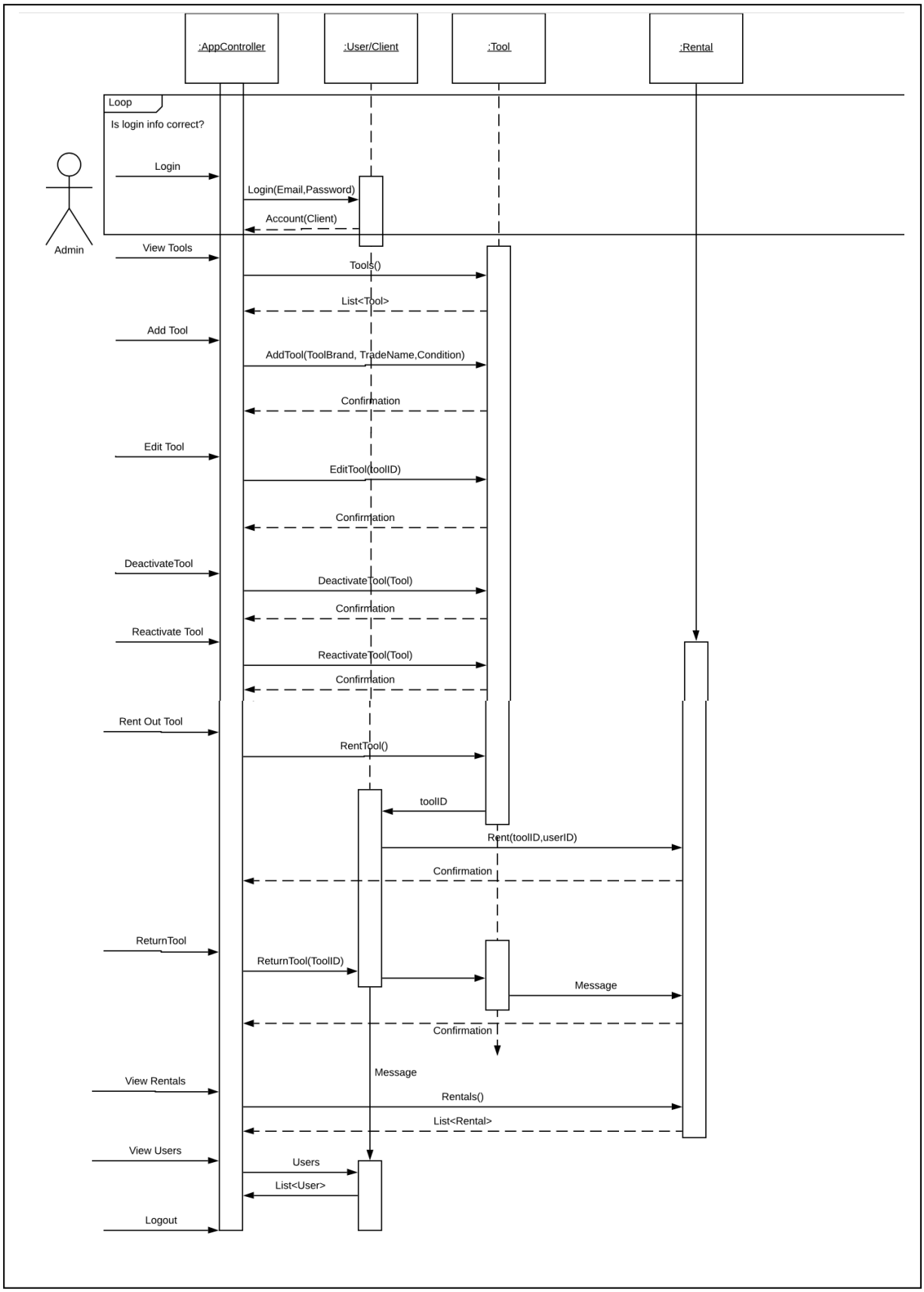


Figure 5.7: Sequence Diagram for Admin

5.1.3 INTERFACE DESIGN

5.1.3.1 SYSTEM NEW CLIENT USER

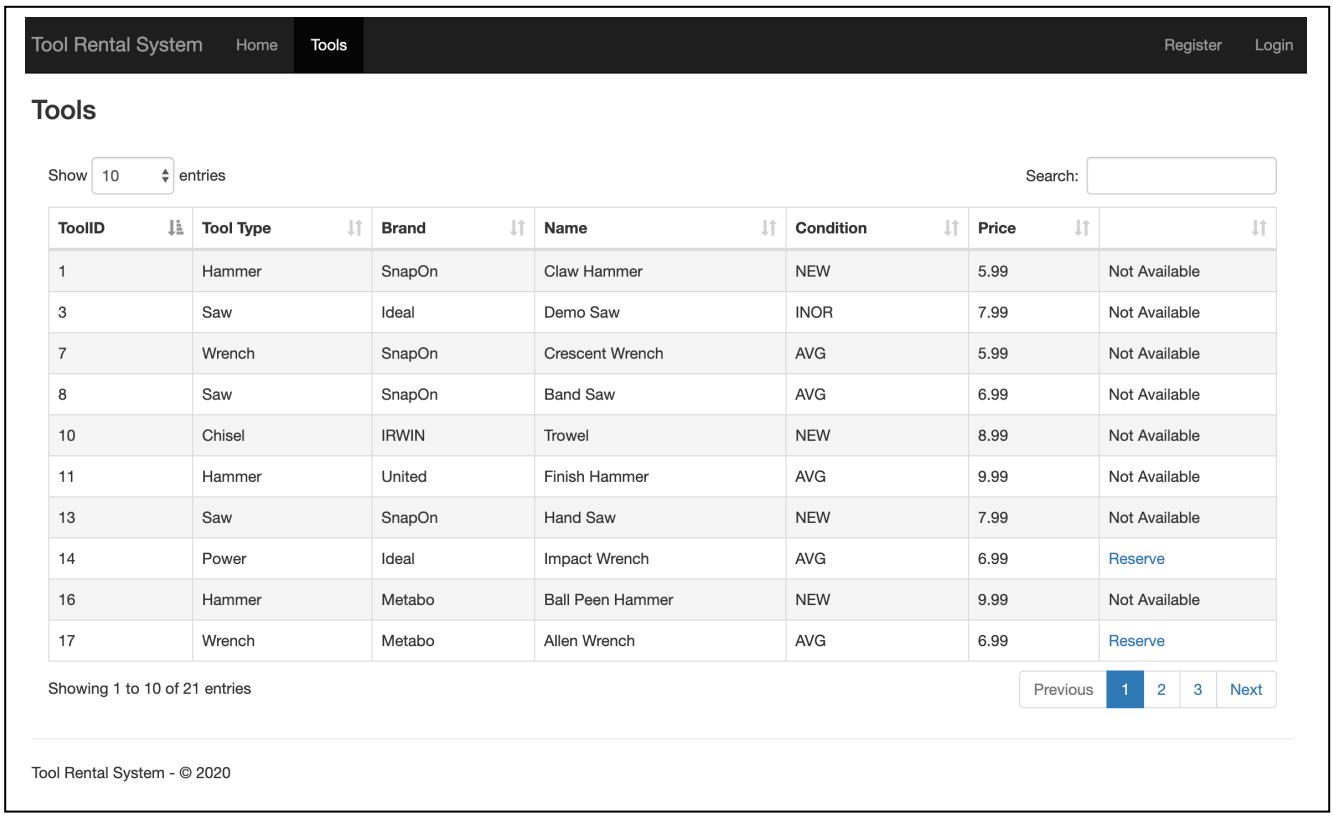


Figure 5.9: Interface design for browsing tools page for new client

5.1.3.2 SYSTEM OLD CLIENT USER

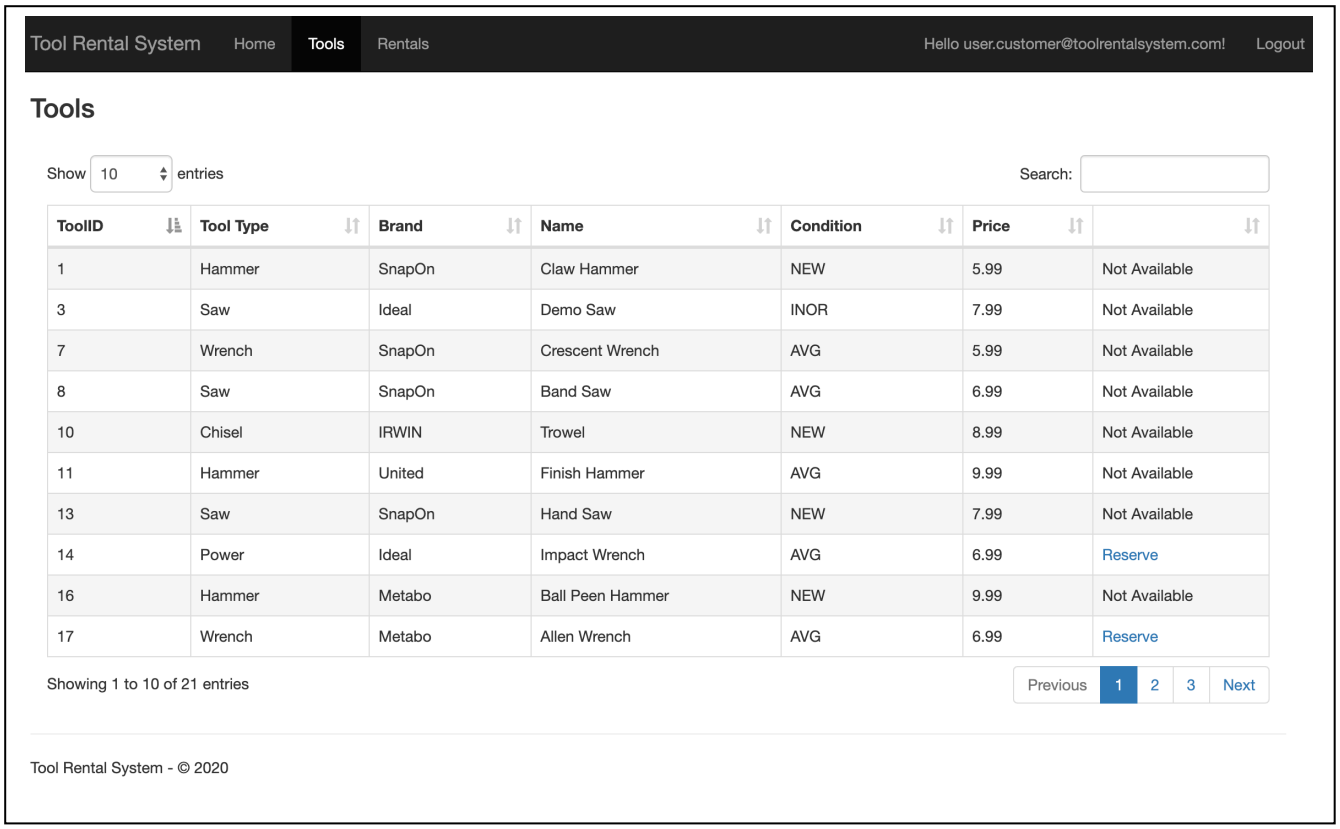


Figure 5.10: Interface design for view tools page for old client

5.1.3.3 SYSTEM ADMIN USER

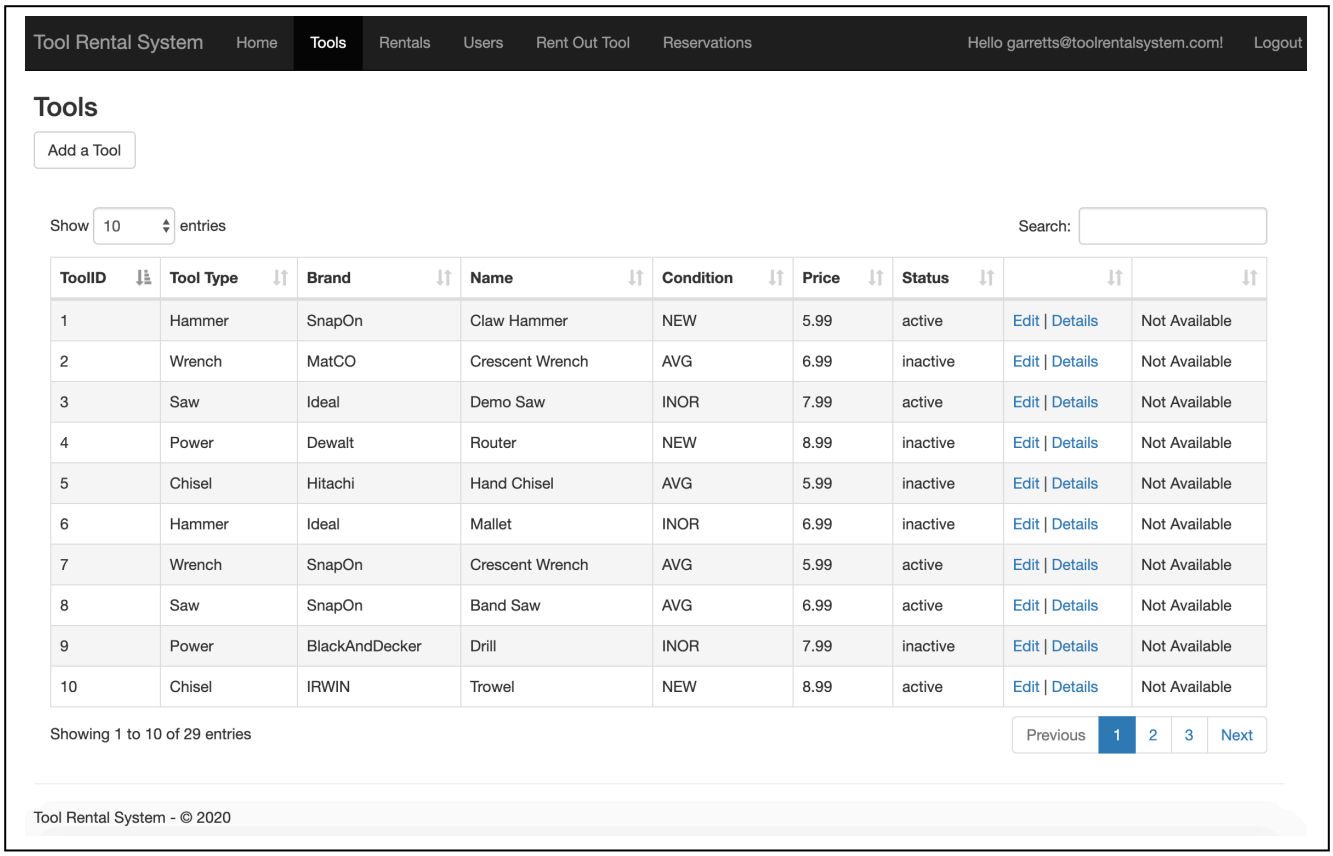


Figure 5.11: Interface design for view tools page for admin

5.1.4 ACTIVITY DIAGRAM

An activity diagram presents a series of actions that used in business modeling. The basic purpose of activity diagrams is to capture the dynamic aspects of the system. It can also describe the pre- and post-condition for Use case diagram and it is used to show message flow one activity to another. The following activity diagrams explain the processes of login and Registration action.

5.1.4.1 USER LOGIN ACTIVITY DIAGRAM

User can login into the website using their user name and password. The user name and password must be valid otherwise the system will prompt to login again.

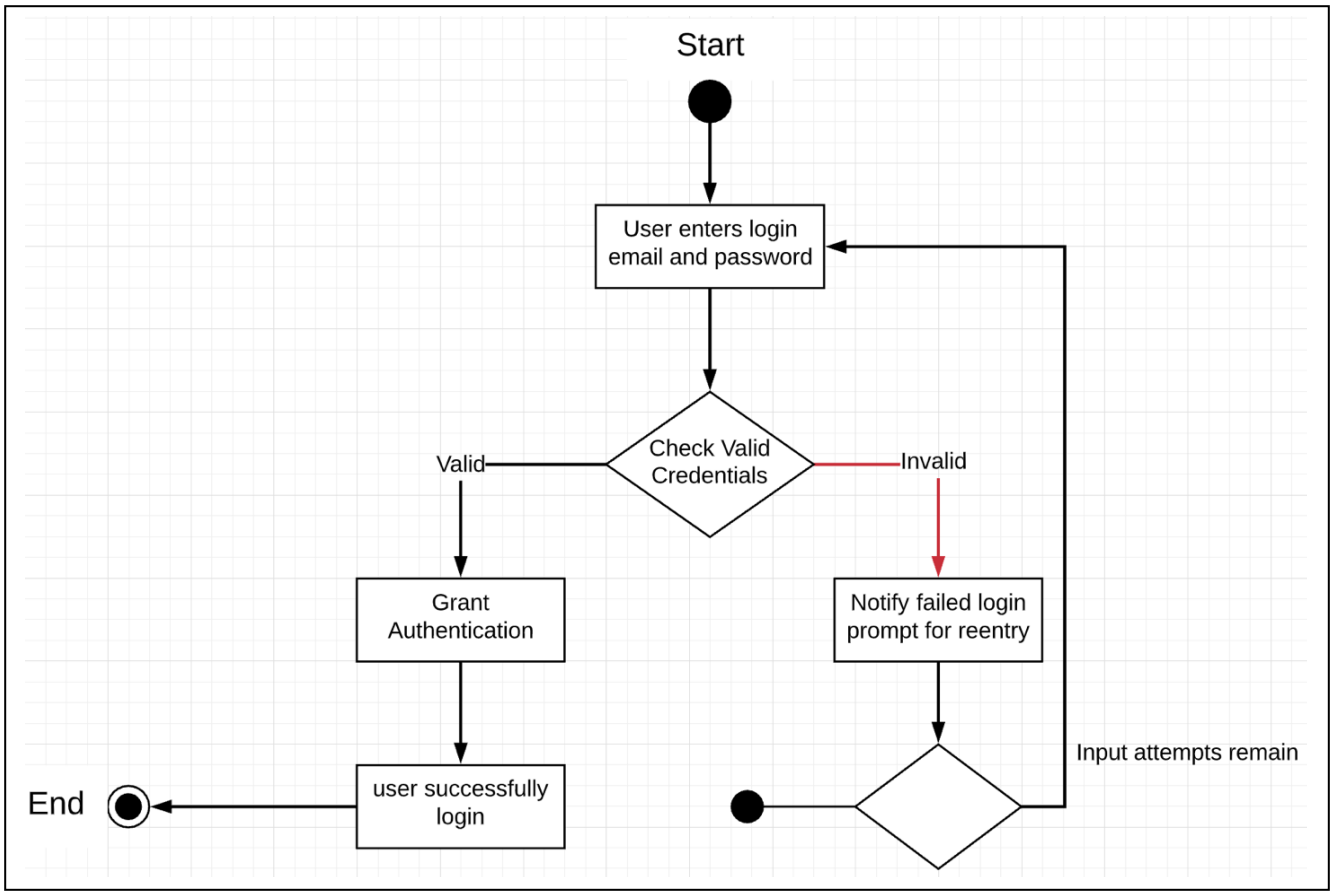


Figure 5.12: Activity Diagram for user login

5.1.4.2 CLIENT REGISTRATION ACTIVITY DIAGRAM

To become a user, a New Client must fill out the registration form and submit the form. The New Client must provide valid personal information to be able to accepted by the system.

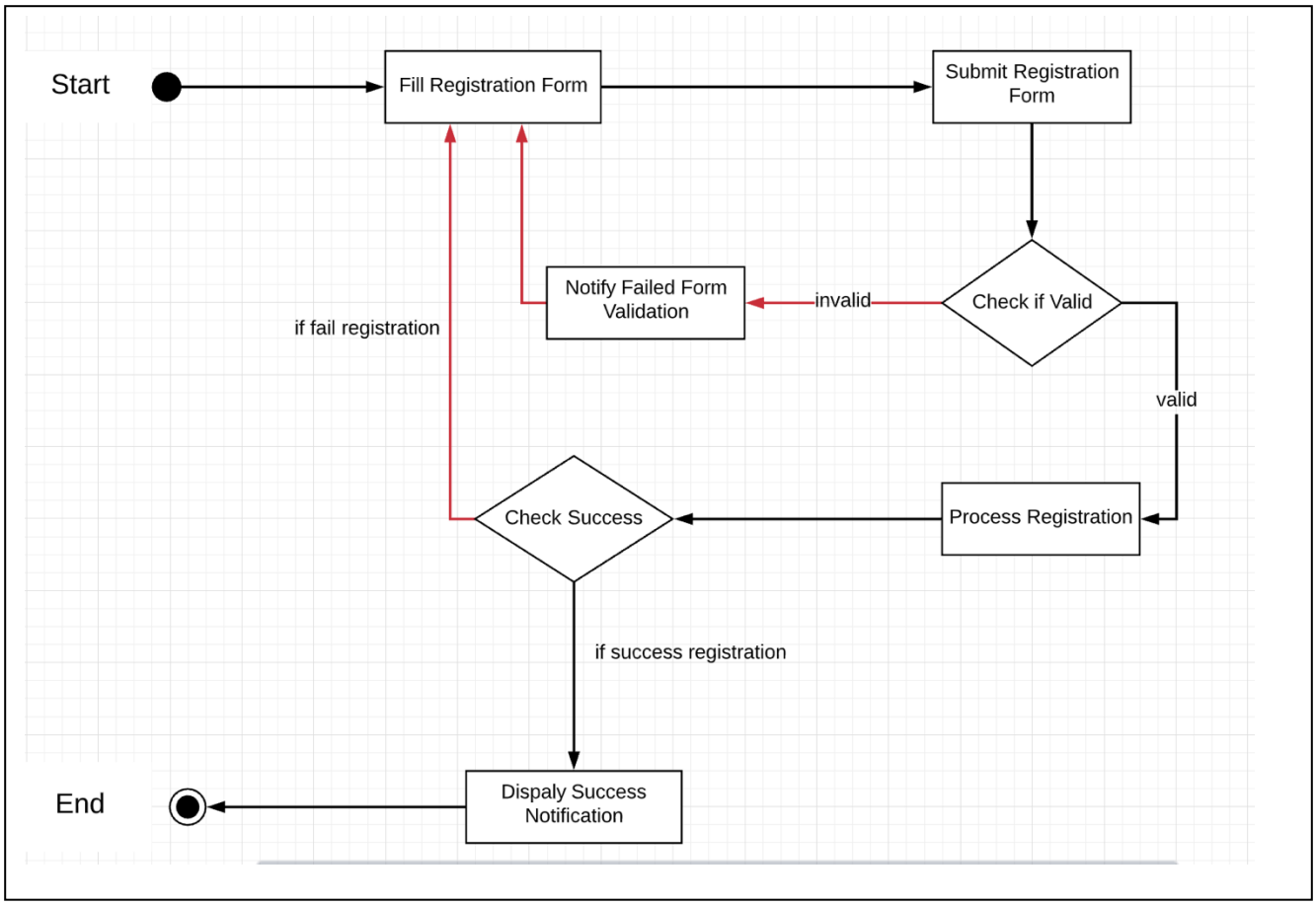


Figure 5.13: Activity Diagram for client registration

5.2 DATABASE

5.2.1 CLASS DIAGRAM

Class diagrams show the structure of a system, including its classes and the attributes of and relationships between those classes.

The Tool Rental System has two superclasses of actors, Users and Clients. The non-actor object classes include Rental and Tool. A tool is a representation of a specific real-world tool. A Rental represents a tool rented out to a user, and a Reservation is a type of rental with rental status ‘reserved’ which represents a tool reserved by a user.

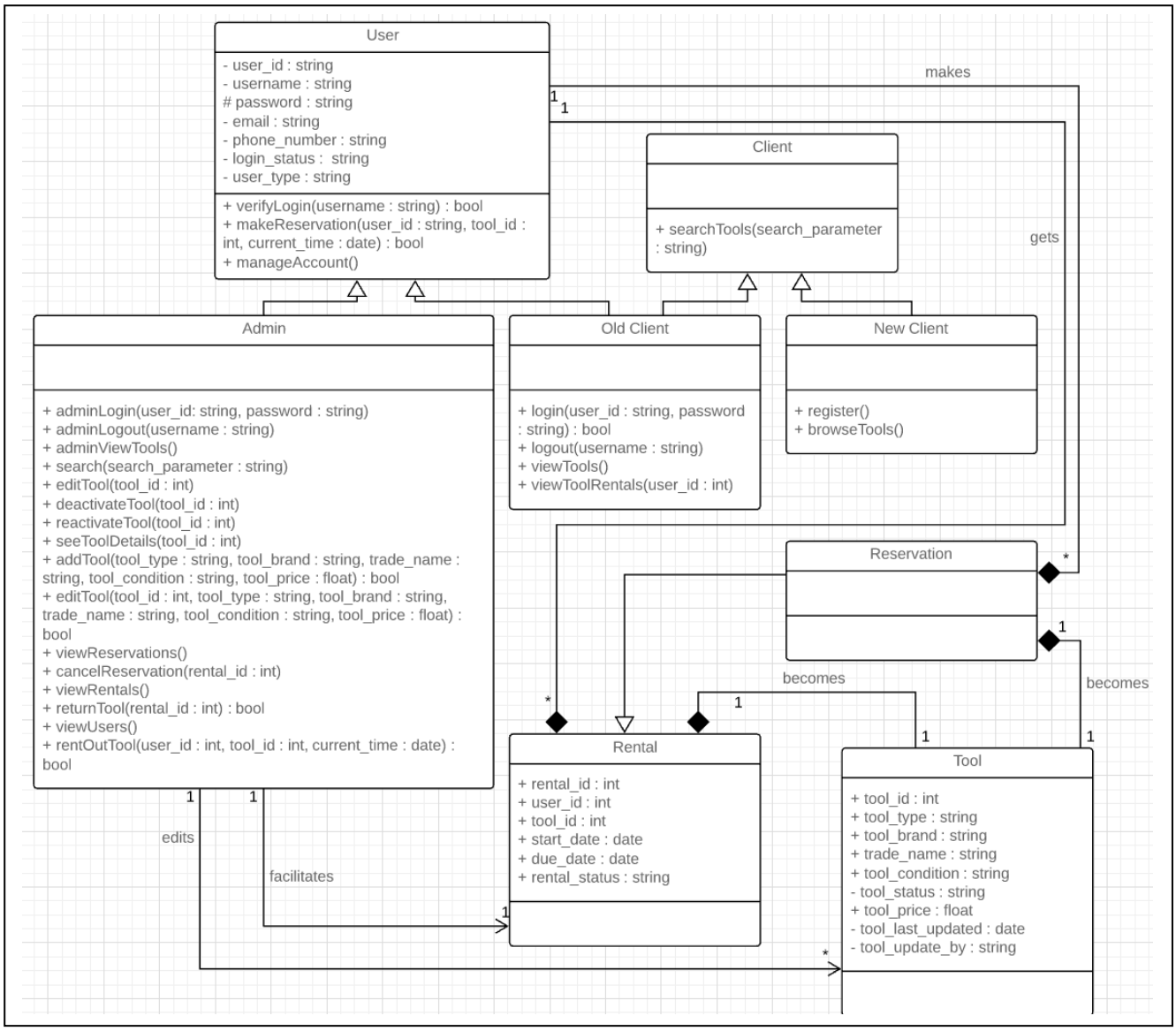


Figure 5.14: Class Diagram

5.2.1.1 SYSTEM CLIENT

Clients are those actors who are not employed by the company running the Tool Rental System but who use its services. New Clients are clients who do not have an account with the system and thus can only browse tools and register. Once Clients register with the system they become Old Clients, which have more functionality including the ability to get rentals.

5.2.1.2 SYSTEM USER

Users are actors who have accounts with the Tool rental system, and thus have a record in the database including their user id, user name, password, etc. This includes Old Clients and Admins. Admins have much more functionality than Old Clients, including the ability to add and edit tools as well as rent tools out to Users.

6.0 PROJECT SYSTEM

6.1 SYSTEM FUNCTIONALITY

We are designing a tool to enable our client companies to catalogue and keep track of tools rented to their customers. Our system will allow the customers to browse the client company's selection and choose what they need without the client company having to maintain hard paper records.

- **Browse/View Tools**

  Clients will be able to view tools available for rent.

View Tools			
1.	User issues a request to see the tools in the system		
		2.	System returns a list of the tool inventory

Table 6.1: Use Case table for View Tools

- **Admin View Tools**

  Admin users can view all tools in the system.

- **Search**

  Users and Clients can search tools, rentals, etc.

- **Register**

  New Clients can register in order to use the system.

Register			
1.	Client issues a request to register		
		2.	System asks for client's info (i.e. first name, last name, email, password,etc...)
3.	Client enters their information		
		4.	System uses said information to generate an account with a unique id number for this client and stores it

Table 6.2: Use Case table for Register

- **Login and Logout**

  Users can login to their account to access more features and then logout of their account.

Login			
1.	User Issues a request to log in to the system		
		2.	System asks user for an email address and password
3.	User enters their email address and password		
		4.	System Verifies the user name and password
		5.	System allows user access to their corresponding account

Table 6.3: Use Case table for Login

Logout			
1.	User issues request to log out		
		2.	System logs the user out

Table 6.4: Use Case table for Logout

- **View Tool Rentals**

  Old Clients can view the tools they have reserved and tools they have rented both currently and in the past.

View Rentals (Client)			
1.	Client Issues a request to see all of their tool rentals		
		2.	System returns a list of all of the tools they have rented.

Table 6.5: Use Case table for View Rentals (Client)

- **View Rentals**

  Admins can view all current rentals.

View Rentals (admin)			
1.	Admin Issues a request to see all of the tool rentals		
		2.	System returns a list of all of the tools that have been rented.

Table 6.6: Use Case table for View Rentals (admin)

- **View Reservations**

  Admins can view current reservations.

- **Reserve Tool**

  Admins can reserve an available tool out to any User, Old Clients can reserve an available tool for themselves.

Reserve Tool			
1.	Client Issues a request to reserve a tool		
		2.	Displays the Reserve tool view prompting for reserve infromation(userID, Tool ID, start date, end date)
3.	Client enters the requested info for the tool to be rented and presses "Reserve Tool" button		
		4.	The system then confirms the reservation for the client and informs them that the tool is available for pickup

Table 6.7: Use Case table for Reserve Tool

- **Cancel Reservations**

  Admins can cancel reservations.

- **Add Tool**

  Admins can add tools to the database

Add Tool			
1.	Admin issues request to add a tool		
		2.	System asks for the information for the tool (i.e. tool name, tool description, and tool type ID)
3.	Admin enters the tool information		
		4.	System generates a record of this tool with a unique tool ID and adds it to the collection of tools with confirmation given to the admin

Table 6.8: Use Case table for Add Tool

- **Edit Tools**

  Admins can edit tools currently in the database

- **View Tool Details**

  Admins can see more details about individual tools.

- **Deactivate and Reactivate Tools**

  Admins can deactivate tools so that Clients cannot view them, and reactivate inactive tools so that Clients can view them again.

Deactivate Tool			
1.	Admin presses deactivate button from edit tool screen		
		2.	System changes the status of the tool to "inactive" removing it from the list of available tools
3.			System saves changes and gives the user confirmation

Table 6.9: Use Case table for Deactivate Tool

Reactivate Tool			
1.	Admin presses reactivate button from edit tool screen		
		2.	System changes the status of the tool to "active" removing it from the list of available tools
3.			System saves changes and gives the user confirmation

Table 6.10: Use Case table for Reactivate Tool

- **Rent Out and Return Tool**

  Admin users can rent tools to and return tools from clients.

Rent Out Tool			
1.	Admin Issues a request to rent out a tool		
		2.	Displays the Rent out tool view prompting for rental infromation(userID, Tool ID, start date, end date)
3.	Admin enters the requested info for the tool to be rented and presses "Rent Out Tool" button		
		4.	System confirms the rental of the tool

Table 6.11: Use Case table for Rent Out Tool

Return			
1.	Admin issues a request to return a tool		
		2.	System asks for the tool ID for the tool to be returned
3.	Admin enters the tool ID for the tool		
		4.	System updates the client and tool records to reflect the return of the tool. Then gives confirmation for the admin and the Client

Table 6.12: Use Case table for Return Tool

- **View Users**

  Admin users can view users of the system.

View Users			
1.	Admin issues a request to see the tools in the system		
		2.	System returns a list of the users that have registered

Table 6.13: Use Case table for View Users

6.2 NON-FUNCTIONAL REQUIREMENTS

In order to maintain software standards and reliability, there are a couple aspects of the Tool Rental System that must be addressed:

1. The Tool Rental System, should have dynamic host switching if ever servers go down. This can easily be accomplished with EC2 service by AWS. This makes web service elastic. This service can obtain servers dynamically. Meaning no down time through Amazon's Elastic cloud computing technology.
2. EC2 also offers resizing as a functionality which solves the problem of manual scaling while application uses change.
3. EC2 is deployable on any RDS database instance. Ours specifically, utilizing mySQL database.

6.3 SYSTEM REQUIREMENT

6.3.1 SOFTWARE REQUIREMENT

The following software was required for the development process:

- Programming Tools: ASP.NET C#, HTML, CSS, JavaScript, Bootstrap, and AWS RDS database (MySQL)
- Version Control: GitHub
- Documentation Control: Google Drive
- Messaging/Conference/Tasks: Discord and Zoom
- Diagram Drawing: Lucid Chart

6.3.2 HARDWARE REQUIREMENT

Our application is web based. Some things that are required for function: A computer and operating system (OS). We strongly recommend a computer fewer than five years old, a processor with minimum 1 GHz.

OS should be as follows: Windows 7 or above, Max OSX v10.0 or higher, or Ubuntu Linux. Internet connection is also required to use the web application. Google Chrome or Firefox is the recommended browser for use.

7.0 SYSTEM LIMITATION

- **GENERAL SYSTEM LIMITATIONS**
    - In its present state the system is set up to manage the inventory of tools at a single location
    - The system is currently not set up to handle monetary transaction.
    - The system currently does not show any images of the tools.
    - System currently has no way to track delinquent renters.
    - Tools cannot be removed from the system. This is by design since removing tools would also remove their rental history, which we want to keep a record of.
- **ADMIN USER LIMITATIONS**
    - Currently the system does not keep track of the time that an admin is logged in, this could be implemented to help when it comes to tracking admin hours.

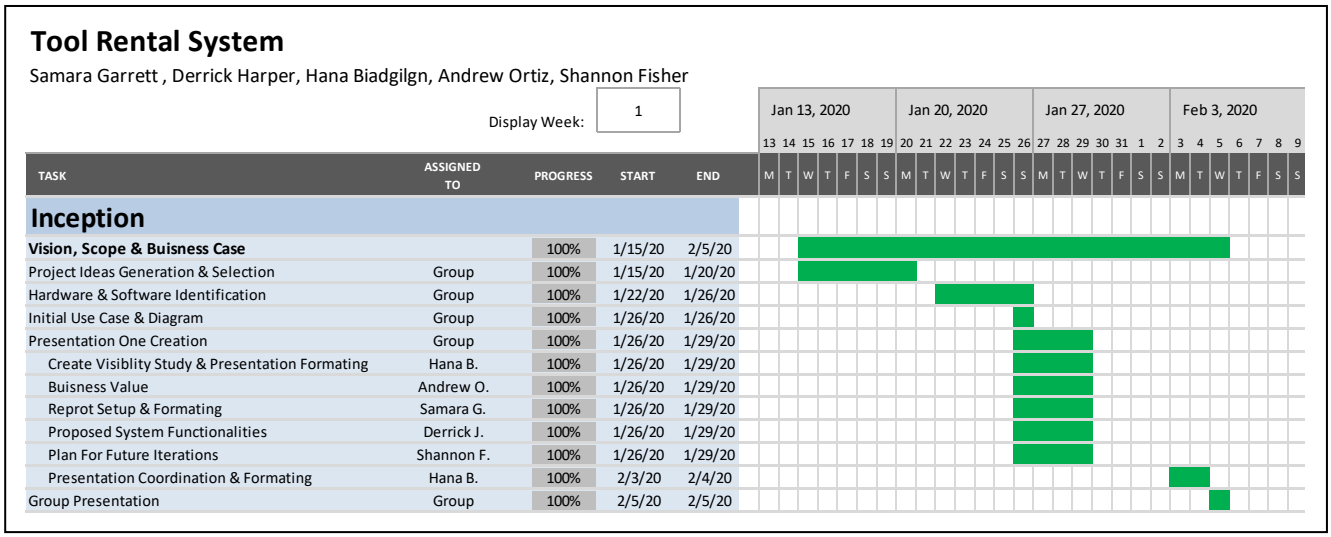
8.0 SYSTEM FUTURE ENHANCEMENT

- **GENERAL SYSTEM ENHANCEMENTS**
    - Allow for functionality that would allow the system to keep track of tool inventories at more than one location.
    - Provide images of the tools on the tool detail pages.
    - Allow some space on the home page for communications such as promos, discounts, and announcements of new tool additions
- **ADMIN USER ENHANCEMENTS**
    - Implementation of a mechanism for the system to handle monetary transactions.
    - Allow more functionality for admins to manage the list of users much like they are currently able to manage the list of tools.
    - Add the physical location from which the tool was rented on the rentals page.
- **OLD CLIENT USER ENHANCEMENTS**
    - Allow a user to enter a credit card and have the system keep it attached to their account.
    - Add more user demographic information to the database (i.e. Name, Address, Occupation ,etc).
    - Show the aforementioned info on the user's "Manage My Account" page.
    - Set up a page to detail where the users can pick up their rented merchandise.
    - Require the use of a credit card to perform a reservation
    - If the only card the user has is expired then they will have to add a new one before they can successfully reserve a tool.
    - Implement functionality to track whether or not the user account is in good standing or if they have rented tools and not returned them.
    - A purchase price on the tools so that the company may charge the user's credit card if they fail to return a tool that they have rented.
- **NEW CLIENT ENHANCEMENTS**
    - Implementation of a screen to show new users physical locations of where tools may be picked up and returned.
    - Set a constraint such that a new user cannot register without entering at least one credit card .

9.0 PROJECT SCHEDULING

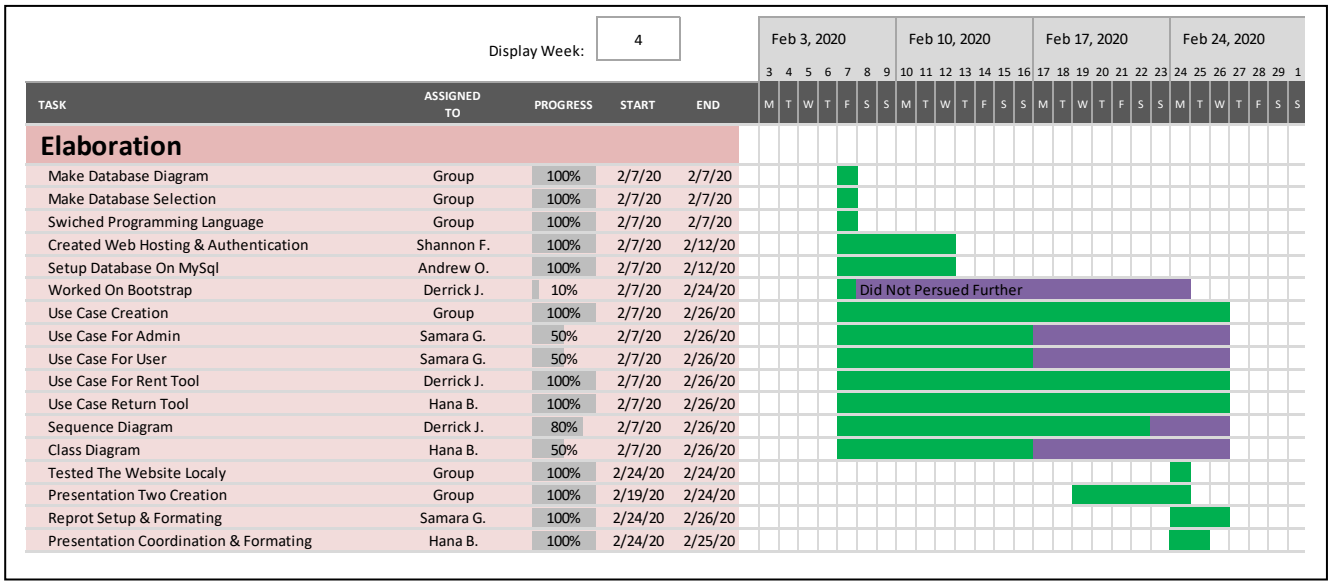
9.1 SCHEDULING TASKS DURATION

Gantt chart for visualizing project tasks.



TASK	ASSIGNED TO	PROGRESS	START	END
Inception				
Vision, Scope & Buisness Case		100%	1/15/20	2/5/20
Project Ideas Generation & Selection	Group	100%	1/15/20	1/20/20
Hardware & Software Identification	Group	100%	1/22/20	1/26/20
Initial Use Case & Diagram	Group	100%	1/26/20	1/26/20
Presentation One Creation	Group	100%	1/26/20	1/29/20
Create Visiblity Study & Presentation Formating	Hana B.	100%	1/26/20	1/29/20
Buisness Value	Andrew O.	100%	1/26/20	1/29/20
Reprot Setup & Formating	Samara G.	100%	1/26/20	1/29/20
Proposed System Functionalities	Derrick J.	100%	1/26/20	1/29/20
Plan For Future Iterations	Shannon F.	100%	1/26/20	1/29/20
Presentation Coordination & Formating	Hana B.	100%	2/3/20	2/4/20
Group Presentation	Group	100%	2/5/20	2/5/20

Table 10.1: Gantt chart for Inception phase of development of Tool Rental System



TASK	ASSIGNED TO	PROGRESS	START	END
Elaboration				
Make Database Diagram	Group	100%	2/7/20	2/7/20
Make Database Selection	Group	100%	2/7/20	2/7/20
Swiched Programming Language	Group	100%	2/7/20	2/7/20
Created Web Hosting & Authentication	Shannon F.	100%	2/7/20	2/12/20
Setup Database On MySql	Andrew O.	100%	2/7/20	2/12/20
Worked On Bootstrap	Derrick J.	10%	2/7/20	2/24/20
Use Case Creation	Group	100%	2/7/20	2/26/20
Use Case For Admin	Samara G.	50%	2/7/20	2/26/20
Use Case For User	Samara G.	50%	2/7/20	2/26/20
Use Case For Rent Tool	Derrick J.	100%	2/7/20	2/26/20
Use Case Return Tool	Hana B.	100%	2/7/20	2/26/20
Sequence Diagram	Derrick J.	80%	2/7/20	2/26/20
Class Diagram	Hana B.	50%	2/7/20	2/26/20
Tested The Website Localy	Group	100%	2/24/20	2/24/20
Presentation Two Creation	Group	100%	2/19/20	2/24/20
Reprot Setup & Formating	Samara G.	100%	2/24/20	2/26/20
Presentation Coordination & Formating	Hana B.	100%	2/24/20	2/25/20

Table 10.2: Gantt chart for Elaboration phase of development

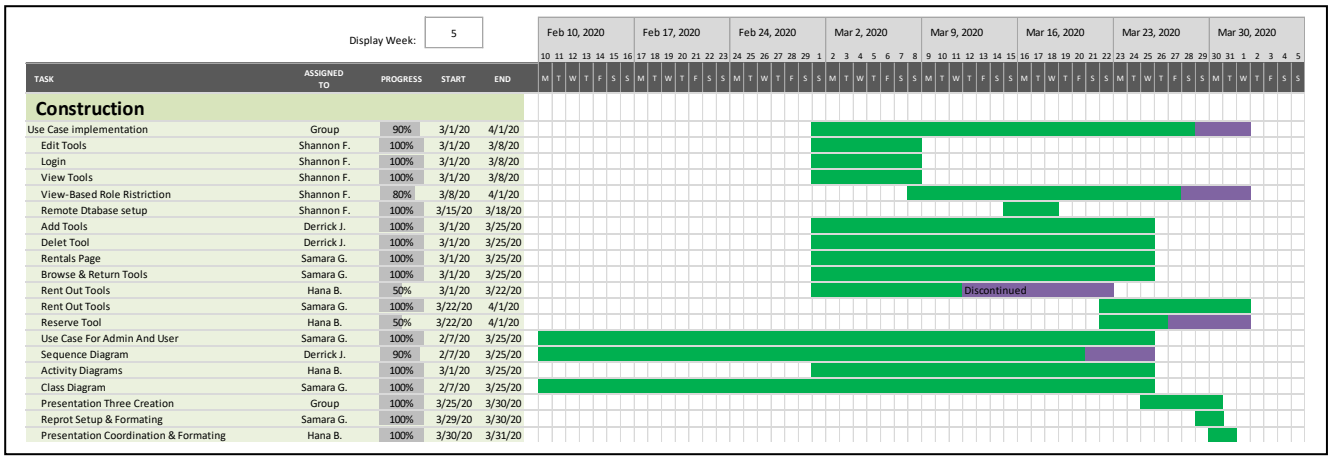


Table 10.3: Gantt chart for Construction phase of development

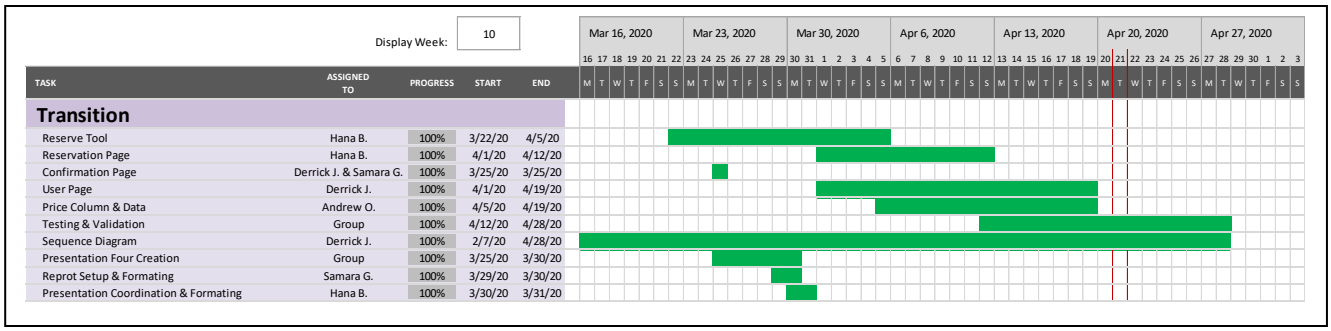


Table 10.4: Gantt chart for Transition phase of development

9.2 PROJECT TIMELINE

Iteration 1:

- Decide the scope and vision of the project
- Determine the requirements and use cases
- Research the business value and feasibility
- Draw the initial use case, sequence, and class diagrams
- Make a prototype of the user interface layout

Proposed Iteration 2:

- Refine the scope and requirements with respect to the remaining time for the project
- Identify and present any new use cases with corresponding sequence and class diagrams
- Continue implementing functionality for use cases

- Work on improving the user interface
- Present a demonstration of the system and its progress since the last iteration

Proposed Iteration 3:

- Continue implementing use-cases in application
- Make any necessary changes to diagrams
- Present Use-Case, Sequence, Class, and Activity Diagrams
- Move database to AWS RDS cloud-based instance
- Present a demonstration of the system and its progress since the last iteration

Proposed Iteration 4:

- Implement remaining use cases
- Refine and complete diagrams
- Review of the scope, vision, and requirement for the solution
- Present final use case, sequence, class, and activity diagrams
- Present the source code and user manual, time-permitting
- Final release product demonstration

10.0 CONCLUSION

Our web-based application hopes to serve those looking for a commercial solution for a rental system. Focusing specially on renting tools, our system runs on a combination of: ASP.NET, AWS RDS database, C#, among others. Our project scope targets business efficiency and end-user experience. We aim to provide a structure and benchmark for new businesses, as well as established, who are in the tool rental space. Tool Rental System can optimize proprietary systems and lower consumer costs. At the touch of a button, you can login, register, rent, check status and availability. You can use the tool for inventory management and tracking.

10.1 RECOMMENDATION

Our specified market/clients would be businesses that rent hardware. These businesses could be large or small, a chain, or sole ownership. Our system can be scaled for smaller projects and revamped for larger organizations. One area of exploration pertaining functionality could be the adding counters for frequency analysis of most used/rented tools. Another could be implementing subscription service-based models within the structure of the web application.

11.0 REFERENCES

ASP.NET Core MVC with EF Core - tutorial series

Dykstra, T., Anderson, R., Latham, L., Addie, S., Pasic, A., & Roth, D. (2017, March 27). *ASP.NET Core MVC with EF Core - tutorial series*. Retrieved from https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/?view=aspnetcore-2.1

ASP.NET Core Authorization

Trivedi, J. (2018, August 18). *Role Base Authorization In ASP.NET Core 2.1*. Retrieved from https://www.c-sharpcorner.com/article/role-base-authorization-in-asp-net-core-2-1/

Kanjilal, J. (2019, January 22). *Policy-based Authorization in ASP.NET Core - A Deep Dive*. Retrieved from https://www.red-gate.com/simple-talk/dotnet/c-programming/policy-based-authorization-in-asp-net-core-a-deep-dive/

ASP.NET documentation

Multiple Authors (n.d.). *ASP.NET documentation*. Retrieved from https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1

ASP.NET MVC Application

Microsoft. (n.d.-b). *Understanding Models, Views, and Controllers (C#)*. Retrieved February 27, 2020, from https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-models-views-and-controllers-cs

AWS DB Instantiation

Amazon Web Servies. (n.d.). *Understanding Amazon Relational Database Service*. Retrieved March 14, 2020, from https://aws.amazon.com/rds/

Azure as a Service

Microsoft. (n.d.). *Microsoft Azure SQL as a Service*. Retrieved February 10, 2020, from https://azure.microsoft.com/en-us/azure/databases/sql/

Class Diagram

Visual Paradigm. (n.d.). *What is Class Diagram?*. Retrieved April 29, 2020, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

Visual Paradigm. (n.d.). *Class Diagram Tutorial*. Retrieved April 29, 2020, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Nonfunctional Requirements

Scaled Agile. (2020, September 18). *Nonfunctional Requirements*. Retrieved April 9, 2020, from https://www.scaledagileframework.com/nonfunctional-requirements/

Server Dynamicity

Amazon Web Services. (n.d.). *Amazon EC2*. Retrieved April 10, 2020, from https://aws.amazon.com/ec2/

Use Case Diagram

Visual Paradigm. (n.d.). *What is Use Case Diagram?*. Retrieved April 29, 2020, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

12.0 APPENDIX 1: USER MANUAL

Tool Rental System Setup Instructions:

Download and install prerequisites (Note: ASP.NET Core is cross-platform):

- .NET SDK 2.1.803
- ASP.NET Core Runtime 2.1.15
- Visual Studio Code (VS Code)
- MySQL 5.7
- Git

Clone the GitHub project:
https://github.com/lancefox1979/ToolRentalSystem.Web.git

Get the "DbCreate.txt" and "DbInsert.txt" files from the following GitHub location:
https://github.com/lancefox1979/ICS499_Project_Resources/tree/master/database

Create the Service Account for the database:

```
CREATE USER 'SA_ToolRentalSystem'@'localhost' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'SA_ToolRentalSystem'@'localhost' WITH GRANT OPTION;
CREATE USER 'SA_ToolRentalSystem'@'%' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON *.* TO 'SA_ToolRentalSystem'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Create and use the Tool Rental System database at the MySQL prompt:

```
CREATE DATABASE ToolRentalSystemDB;
USE ToolRentalSystemDB;
```

Next, run the commands in the "DbCreate.txt" and "DbInsert.txt" files in that order to create the first set of tables and populate them.

Open the cloned project in VS Code and change the server address in the "appsettings.Development.json" file to where your MySQL instance resides, along with any other details in the connection strings as required.

Within the cloned ASP.NET project directory, open a terminal and run the following two commands (this will create a second set of tables in the database):

```
dotnet ef migrations add IdentityCreate -c ApplicationDbContext
```

dotnet ef database update -c ApplicationDbContext

Within the “Scaffold Refresh.txt” file in the VS Code project, change the connection string to your MySQL instance as required, the copy and paste the entire line to run it as a command within a terminal in the root of the project directory. This will update the C# Models files to capture any changes made to the database.

Within the “Startup.cs” source file is a method named ManageRoles. Tailor this method to add your own administrative accounts for the web application, and uncomment the method call within the Configure method so that it executes. This line can be commented-out again after running the application for the first time.

The web application should now run from either within VS Code or by running the following command from a terminal within the project directory:
dotnet run

13.0 APPENDIX 2: PROJECT LOG

Date	Members Present	Major Issues for Discussion	Major Decisions
1/15/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- What sort of topic are we looking for? - Do we want to work for a client?	- Preferably no client - Preferably not a web application - Each group member will come up with 3-10 topics for discussion next meeting
1/20/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Topic options - What programming language should we use?	- Topic will be Library system or similar - Web application is preferable to having to worry about UI of app or similar - Language: Ruby (on Rails) with MySQL (or similar) - GitHub for version control, Google Drive for documents - For next meeting: prepare reference you think might be useful
1/22/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Is library topic acceptable? - What functionality do we want? - What programming language should we use? - Who are the users?	- Topic changed to library of tools rather than books - Users: User (see rented tool history), Staff (check items out and in, add and remove items), Admin (handles staff?) - For next meeting: start working on presentation requirements
1/26/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Presentation requirements such as scope and use cases - What language to use? (tabled for next week) - UI	- Business should be for-profit rather than a Library - Use cases, use case diagram - Basics of UI - Progress made on powerpoint - Separate stretch goals from necessary items - Email professor about Business Value and Feasibility Studies requirements - For next meeting: Research languages, work on powerpoint, start UI prototype in HTML
1/29/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- What language - Assign things that need to get done for the presentation - Powerpoint and report logistics	- Language will be Ruby - Remaining parts of the presentation assigned - Last edits to content should go in by 11:59pm Monday so that the powerpoint and report can be formatted by Wednesday

2/2/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- 2 truths and a lie for quiz - Presentation content - Presentation speaking assignments	- Made some changes to presentation content - Presentation slides split among group members for presenting
2/7/20	Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- How presentation went - Database Diagram - Plan for the week	- Made changes to entity-relationship diagram - Decided to start with MySQL for website database - Decided on what to work on for the week
2/12/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Ruby is not going well, so switch language? - Use case elaboration - Class diagram	- Switch to .net (or php if .net doesn't work on mac) - Got some work done on class diagram
2/16/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper	- Use Case Diagrams - Class Diagram - Module due March 4th - What needs to be included in Presentation 2	- Split use cases so that each group member will work on an elaboration - Made a plan for the Teamwork and Collaboration Module Assessment due March 4th
2/19/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Use cases - What is necessary for the next presentation - Plan for following week	- Clarified some use case questions with the professor - Next meeting we will work on the presentation and work on getting the system up and running - Hana will try to reserve a room at the library with a projector if possible so that Shannon can demo some steps to getting the system to work
2/24/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Presentation content and who is presenting what - Report content - Worked on getting system running locally for everybody	- Split up presentation sections - Get all slide edits in by Monday night so that Hana can format - Everyone got system (and database) running locally
3/1/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- What do we need to do for the next presentation? - Report sections - Split up implementation of use cases - Do videos for Collaboration and	- Split up some report sections to work on - Split up some use case implementations to work on - Ask the professor for clarification on some parts of the report - Everyone will do the User Manual section for the use

		Teamwork Module	cases they work on
3/8/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Progress made since last meeting - Shannon walked us through changes made to system	- We will start working on our use case implementations based on what Shannon has done - Continue working on report sections
3/11/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Bug with tools table - Progress on code and report	- Adjust database and then code to fix bug - Create github for database files for version control - Return tool use case will include viewing currently rented tools
3/15/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Code and report section progress - Changes to database - Code help	- Decided on some changes to be made to database
3/18/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper	- Progress made on code - Database changes - Remote database setup - Requirements for next presentation - No usb required for final presentation	- Split up presentation requirements to work on - New code assignments for those that finished their previous ones - Save a version of the application next Wednesday for the presentation so that we can base our diagrams on it
3/22/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Coding progress - Stumbling blocks	- Moved around some coding assignments - Decided on new assignments to work on
3/25/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Confirmation page - Upcoming presentation, who is presenting what and how demo is run	- For confirmation screen: if nothing changes then don't update database, try to utilize ViewBag.Message such that we have one confirmation screen with different messages depending on method calling it - Presentation parts split up
3/29/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper	- Activity diagrams, sequence diagrams, use case diagrams, class diagram - Presentation changes - Report sections	- Change activity diagrams - If professor writes back about sequence diagrams today or tomorrow we will try to change them, otherwise they will be changed for next iteration

	Andrew Ortiz		- Changes made to software requirements - Demo moved to end of presentation
4/3/20	Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- To Do List - Current Assignments - How presentation went - Bug with add tool	- Picked assignments to work on currently - Refreshed database - Bug with add tool may be fixed by some changes Derrick has made? If the changes he commits don't fix it we will revisit the bug.
4/5/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Bugs - Code progress - Database changes	- Wait until we are all working from the same commit to look at bugs - Derrick will commit Add Tool confirmation page - Database changes - Change tool classification to tool type everywhere (database, models, view, etc.)
4/8/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Scaffold Refresh - Reserve Tool, Reservations page - Current progress and bugs	- Get rid of extraneous code in ToolRentalSystemDBContext so that anyone can run a scaffold refresh - Add Reserve Tool to Tools page - Add Reservations page (with table of reservations) to Admin navbar, with Cancel Reservation button - Change date in database to datetime - Reservation is 24hrs
4/12/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Did video assignment - What else do we need to get done - Discussed progress	- Prioritized remaining to-do list - Current assignments - Decide by Wednesday what we want to finish
4/15/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Progress since last meeting - What is left to do - Project Report	- Got information on what sections of Project Report we need to do (and what they should be on) from professor - Asked professor about Paper - All new code should be in by Wednesday so that we can focus on report (bug fixes can be done afterwards, but no major changes) - Split up some report sections to get working on
4/19/20	Hana Biadgilgn Shannon Fisher	- Code progress, bugs - Report progress	- Add returned & reserved to client Rentals view - Ask professor about user manual/deployment

	Samara Garrett Derrick J. Harper Andrew Ortiz	- Split up parts of presentation	instructions?
4/22/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Progress on report and code - Questions about report sections - Presentation demo	- Split up demo sections - Moved some report assignments around - Asked professor questions about report specifics
4/26/20	Hana Biadgilgn Shannon Fisher Samara Garrett Derrick J. Harper Andrew Ortiz	- Ran practice presentation and demo - Remaining report parts - Small code decisions	- Get rest of report parts in by Tuesday - Make inactive tools not reservable