University College London

Faculty of Engineering Sciences

COMPGC22 Software Engineering

Supervisor:
Rae Harbird

Email:
r.harbird@ucl.ac.uk

Authors:
Ahmed Afify
Gideon Acquaah-Harrison
Daiana Bassi
Antonio Manganelli
Cameron Mory
Stefan Poechhacker
Caroline Smith
Phoebe Staab

20th March 2018

Executive Summary

The key objective of this body of work is to illustrate the process by which Group A designed the core functionalities of the web-based DIY Tool Hire Service (THS) application. During the ideation stage of the project, the team identified the scope of the project and the core requirements of the DIY THS. Through the use of a SCRUM embedded Unified Process workflow, the team developed a blue print for the application from requirements and use case evaluation through to object-oriented analysis and prototype design. These achievements were possible through the use of the Unified Modelling Language (UML) as the medium through which information was shared.

Keeping with the tenants of the Unified Process, requirements and use cases were at the core of all system designs. As such, the development of the key requirements of the system and the use cases through which they are performed were developed through a rigorous and iterative process over several weeks. The results of this effort can be seen and explored further in Chapter 1. These served as a guide for the subsequent object-oriented analysis and architecture design.

During the object-oriented analysis of the DIY THS, the team kept with the aforementioned iterative approach. After the development of plain-English activity diagrams for key use-cases, the team endeavored to create some basic pseudo code to carry out such processes. This lead to the development of class diagrams which were, again, use-case driven. The development of the design class diagram and sequence diagrams were closely linked and iterated over for several weeks to ensure consistency between the models. The design class diagram and the objects contained within it were predecessors to the development of state-machine diagrams.

The system architecture was explored through the use of further UML models such as component and deployment diagrams. The DIY THS would be implemented on a three-tier architectural model which is discussed in more detail along with the architectural UML models in Chapter 4.

Finally, a mock-up of prototype designs is presented at the close of this work to bring the UML models to life. These illustrations also help to emphasize the particulars of the role of the presentation tier in the three-tier system. Alongside the figures in Chapter 5 are detailed descriptions of the use-cases related to their respective mock-up designs.

While future iterations of this work would not only be possible, but encouraged before implementation, this report ultimately gives a realistic and pragmatic proposed approach to the development of a web-based tool hire service.

UCL

Contents

List of Tables

List of Figures

List of Acronyms

THS: Tool Hire Service

DIY: Do it yourself

USDP: Unified Software Development Process

MVP: Minimum viable product

OO: Object oriented

UP: Unified Process

UC: Use case

UML: Unified Modelling Language

1 Project Overview

1.1 Team Members

Name	Email
Phoebe Staab	phoebe.staab.17@ucl.ac.uk
Caroline Smith	caroline.smith.16@ucl.ac.uk
Daiana Bassi	daiana.bassi.17@ucl.ac.uk
Ahmed Afify	ahmed.afify.17@ucl.ac.uk
Cameron Mory	cameron.mory.17@ucl.ac.uk
Antonio Manganelli	antonio.manganelli.17@ucl.ac.uk
Stefan Poechhacker	stefan.poechhacker.17@ucl.ac.uk
Gideon Acquaah-Harrison	gideon.acquaah-harrison.17@ucl.ac.uk

Table 1.1: Team members and email addresses

1.2 Introduction

This report will elaborate on the process used by Group A to design the DIY Tool Hire Service. The assignment was completed following the use-case driven Unified Software Development Process (USDP)[1].

The iterative nature of the USDP resulted in multiple evolutions at every step in the process. While each aspect of development – from the requirements gathering and use cases to the object-oriented diagrams – underwent several iterations, only the final iterations have been included within the report. All of the core diagrams found in this report were completed in Unified Modeling Language (UML) format, the industry standard.

This chapter of the report will contain the problem statement, a description of the scope of the project, a glossary of terms used throughout the report, and a Gantt chart to display the project timeline followed by the team. Following these sections is a summary of the project management style. The subsequent chapters will cover the topics of requirements gathering, use case development, object-oriented analysis, and application mockups. Every stage of the development process prior to the delivery of a minimum viable product (MVP) was completed.

1.3 Problem Statement

These days, DIY home projects are very popular. Walk-throughs can be found all over the internet for tasks such as making a bed-side table, painting rooms with interesting patterns, or even building a shed.

One of the main problems when it comes to these tasks is that the average person may not have the tools required readily available. It may also be exceptionally expensive for people to go buy those tools as well. Spending the money to purchase the tool is unlikely when it is only needed for one specific project. DIY hobbyists require a service that enables them to rent the tools they require for a much lower fee than if they had to buy it themselves.

People who wish to complete home projects may also be limited by their lack of experience with the tools. These people need to be provided with advice and video tutorials that will show them how to properly use the tools they wish to rent.

1.4 Project Scope

This report outlines the design process for a web application for individuals who enjoy DIY projects but lack the tools to accomplish them. The goal of the service is to provide a platform for people to rent tools for short-term periods to complete their tasks. It will also offer advice and tutorials for those who may require some help getting started.

The central purpose of the web application is to allow individuals in the London area to hire out tools for their projects. The customer will be able to search through a selection of available tools, with the ability to sort by type, name, or project. Having found the desired tool, the customer will then check the calendar for its availability and then add it to the basket. Upon check-out completion, the database will be updated to reflect the dates for which the tool is booked to prevent another customer from selecting those dates as well.

Customers may also view the blog containing weekly advice and subscribe to it if they wish. They may also submit queries to the customer support if they have any specific questions.

On the other side of the operation, employees may complete a range of tasks including managing the tool listings, uploading new blog posts, and responding to customer queries.

The success of the project would be measured by the delivery of the completed stages of the software development process prior to the creation of a prototype. This would include requirements gathering, use-case analysis, object-oriented analysis and design models, and a system mockup.

1.5 Project Glossary

In order to be clear and consistent throughout the project, specific words and phrases were used. These can be found in the glossary below.

Term	Definition
DIY Tool Hire Service (THS)	The web application on which customers hire tools and employees list tools.
User	The umbrella term for customers and employees when an action is not specific to either.
Customer	A person who uses the THS to hire tools.
Employee	The umbrella term including admin and customer support when an action is not specific to either.
Admin	An employee who is responsible for creating employee accounts and managing the tool listings.
Customer Support	An employee who is responsible for responding to queries and updating the blog and advice section.
Notification Service	A service that sends automated emails to the customer confirming their order or reminding them to return the rented tool(s).
Delivery Service	A service that is responsible for delivering tools when customers have selected the delivery option.
Credit Card (CC) Company	A company that is responsible for validating the customer's credit card when he/she is attempting to place an order.
Tool	Any one of the products available for hire on the THS, ranging from hammers and screwdrivers to table saws and lawnmowers.
Listing	A page of the THS for a specific tool containing the tool name, pictures of the tool, tool summary and specifications, daily cost to hire, and the tool's availability.
Tool Availability	The dates on which a tool is available for hire as displayed by the calendar on the tool's listing page.
Tool Fault	When the tool is malfunctioning for reasons unrelated to the customer using it.
Tutorials	Videos posted in the Advice section that show customers how to use tools or perform certain tasks.
Blog	A page of the application found in the Advice section that will be updated regularly with new posts containing helpful information for customers.
Rental Agreement	The conditions to which the customer agrees when he/she completes the checkout process.
Order	The tool(s) being rented by the customer. The order is created when a tool and date are selected and added to the basket.
Digital Basket	The place where a tool booking is held until the customer completes the checkout process.
Checkout	The process of paying, selecting pick-up or delivery, and confirming after a tool has been added to the basket.
Shippable Tools	Tools that are able to be shipped and not limited to pick-up only do to any restrictions.

Table 1.2: Project glossary containing key terms

1.6 Gantt Chart

The Gantt charted shown below displays the project timeline followed by the team. It is divided into the four project phases outlined in the Unified Process: *Inception, Elaboration, Construction and Transition*.

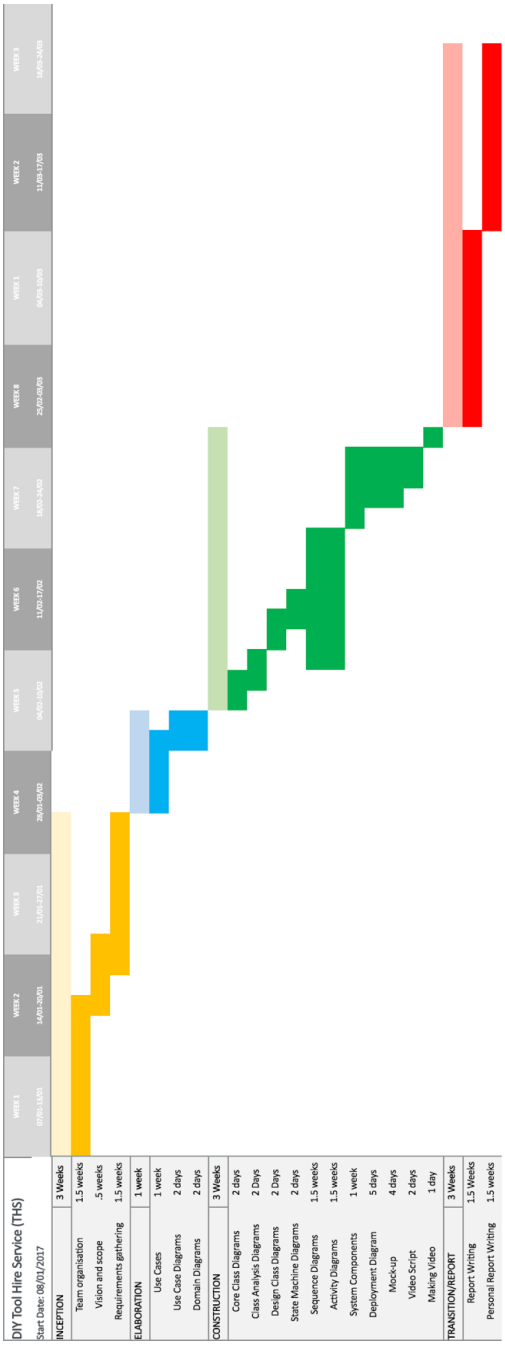


Figure 1.1: Gantt Chart Showing the Project Timeline

1.7 Project Management: SCRUM Embedded Unified Process

The Unified Process or UP was developed in the 1990s and is a pragmatic and tested method for carrying out the development of a software system and was designed specifically for use with UML. Essentially, it is a process by which the *who, what, and when* of a development project can be determined [2]. The three main principles of the UP are:

- The development process should be driven by use-cases and risk assessment
- The development process should emphasize a quality system architectural model
- Software should be developed via step-wise completion of subprojects and continuous refinement through iteration

Within each iteration, five workflows are explicitly defined by the UP. These are Requirements, Analysis, Design, Implementation and Testing [2]. The broader project lifecycle is also covered in the UP and is made up of four distinct phases which are Inception, Elaboration, Construction and Transition. Within each of these phases, multiple iterations should take place to suit the project's needs. Figure 1.2 illustrates how all of the elements of the UP are tied together into a single project lifecycle.

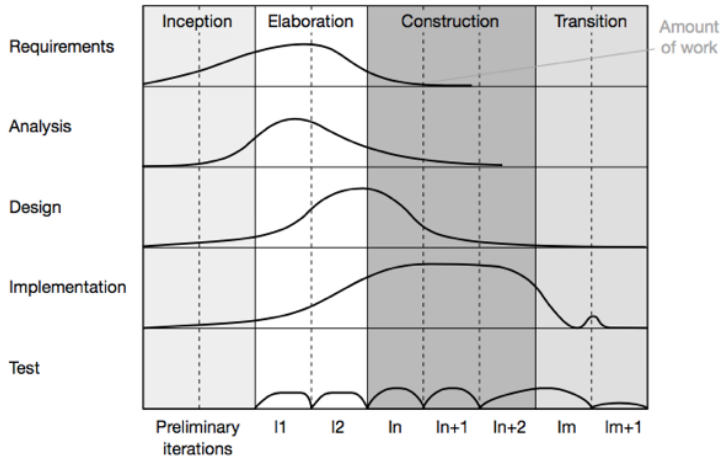


Figure 1.2: Outline of the Unified Process project lifecycle [2]

SCRUM is a subset of Agile and is project management framework that grew out of the failures of previous models such as the Waterfall model. There are three empirical pillars that uphold any SCRUM project [3]. These are:

Transparency: common standards are agreed upon within a team in order to avoid ambiguity.

Inspection: Users must inspect and update SCRUM artifacts throughout the development process.

Adaptation: Progress must be monitored by members of the SCRUM team and if the process has deviated to an unacceptable extent, the product(s) that resulted from that process must be reviewed and adapted.

These pillars underpin the SCRUM project lifecycle approach which involves:

1. The SCRUM team meets in order to prioritize the artifacts in product Backlog. This could be a list of required features for a product.
2. Before a Sprint begins, the team has a Sprint planning meeting to decide what work will be done during the Sprint.
3. The team does a Backlog refinement to confirm that the project is still on track before beginning the Sprint to ensure that work will not be wasted.
4. During the 2-4 week Sprint, the team has daily meetings which are short, lasting only 15 minutes or less.
5. At the end of each sprint, the team members present their work and have a Sprint Retrospective to analyze what could be improved during the next sprint. [4]

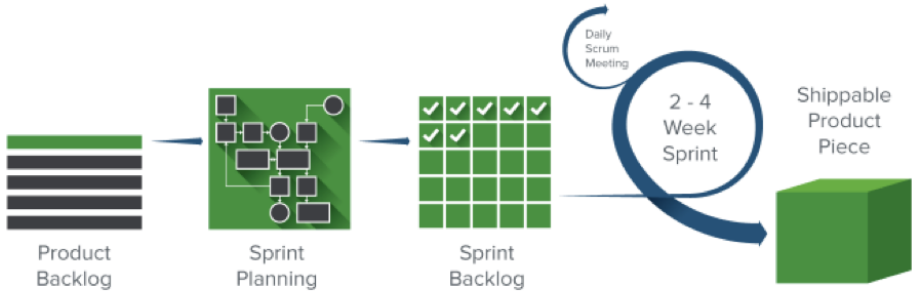


Figure 1.3: The SCRUM project lifecycle [4]

For the DIY THS project, the team utilized both of these methodologies where they were suitable. The broader project lifecycle was shaped by the tenants of the Unified Process, while more day-to-day project management was derived from some useful features of SCRUM.

The UP identifies that emphasizing system architecture and developing based on use-cases leads to successful products. This was integral to the development of the THS system. Given

that the implementation was not carried out for the THS, the team had the opportunity to rigorously refine and identify essential use cases. From these use cases, object-oriented (OO) UML models were derived. After the essential system objectives were laid out through OO models, the architecture was modelled. Additionally, the THS team made use of the four phases of the Unified Process to help structure project progress which is illustrated in the Gantt chart in Section 1.6.

While the team did not assign official SCRUM roles, certain useful aspects of SCRUM were adopted. Early in the project, the team developed a Backlog of the tasks that needed to be accomplished throughout the semester. Then, to fulfill these goals, the team worked in 1 week Sprints followed by Retrospectives to assess the challenges encountered in the previous week and plan the next Sprint. Additionally, transparency and standardized communication were essential for such a large group project and as such, a team glossary was used throughout.

Both the UP and SCRUM outline the importance of iteration, which was the absolute core of the THS project. During the weekly Retrospectives, analysis of the previous weeks work was done by the team members and the project supervisor, which shed light on improvements to be made on the current work products. In many cases, several Sprints were used to refine a single work product.

Overall, the project management was more of an adaptation of key tenants of both UP and SCRUM rather than an absolute adoption of all of their aspects. Ultimately, being able to utilize aspects of such models to suit the team's needs resulted in very smooth project coordination and consistent progress throughout the semester.

2 Requirements Gathering

2.1 Overview

The first task for this project was to develop a list of requirements for the DIY Tool Hire Service. To do so, the team began by organizing a meeting to discuss the scope of the project. Once everyone came to a mutual understanding, the process of coming up with the requirements began. During the meeting, the team reviewed the needs of the customer when using this type of service. Each member then continued to develop requirements individually by researching existing web sites to find important functionalities as well as areas which may be improved.

This initial period of requirements development involved a lot of brainstorming. The aim was to create a list of requirements that fully encompassed the website's functionalities. This would save time and effort further down the line, as making up for mistakes made during the requirement gathering period can become extremely costly [5].

The importance of the requirements gathering stage in the overall scope of the project cannot be overstated. The requirements play a pivotal role in the progression of the project, directly influencing the development of use cases. The use cases then go on to influence the classes and methods used, which in turn influence many of the future diagrams. With that in mind, it was very important that the requirements gathering was done correctly. This is reflected in the significant portion of time that was allotted to develop the requirements, as seen in the Gantt chart in Section 1.6.

2.2 Requirements

Although initially long and containing duplicates, the list of requirements was refined to fall in line with the scope of the project. These requirements were then broken down into subsections such as "Tool Hiring Process" and "Help & Advice" to aid in navigating the list and ensuring the user's needs were accounted for. In Table 2.1, all of the functional and non-functional requirements have been assigned IDs and descriptions, as well as indications about the type of activity and the actors involved. Additionally, every requirement was given a level of priority utilizing the MoSCoW system [6].

ID	Description	Activity	Actors	Priority
FUNCTIONAL REQUIREMENTS				
USER ACCOUNT MANAGEMENT & REGISTRATION				
RQ1	The THS shall validate the username and password of the user and logout when they are done.	Login	All Users	Must
RQ2	The THS shall allow the customer to Logout when they are done making an order	Logout	All Users	Must
RQ3	The THS shall allow new users to register an account.	Registration	All Users	Must
RQ4	The THS shall require the customer to agree to the company's liability statement when registering for their account.	Registration	Customers	Must
RQ5	The THS shall allow a user to sign-in to their account and edit their details.	Account Management	All Users	Must
RQ6	The THS shall allow the user to reset his password.	Account Management	All Users	Must
LISTING TOOLS FOR HIRE				
RQ7	The THS shall allow admin to add new listing of tools available for hire.	Tool Listing	Admin	Must
RQ8	The THS shall allow admin to update existing listings of tools available for hire.	Tool Listing	Admin	Must
RQ9	The THS shall allow admin to delete listings of tools available for hire.	Tool Listing	Admin	Must
RQ10	The THS shall allow the user to sort tools by type, name and functionality.	Tool Listing	All Users	Must
RQ11	The THS shall show a summary of each tool's specifications.	Tool Listing	All Users	Must
RQ12	The THS shall show users up-to-date information about a tool's availability for hire.	Tool Listing	All Users	Must
TOOL HIRING PROCESS				
RQ13	The THS shall allow a customer to enter a date and time they would like to rent a tool.	Tool Hiring	Customers	Must
RQ14	The THS shall allow a customer to select from a range of available rental periods based on a specified desired time/date.	Tool Hiring	Customers	Must
RQ15	The THS shall allow a customer add tools to a digital basket before ordering.	Tool Hiring	Customers	Must
RQ16	The THS shall allow a customer to insert their postal code to see if they qualify for local delivery on shippable tools.	Tool Hiring	Customers	Must
RQ17	The THS shall deliver rented tools to customers if they qualify for delivery.			
RQ18	The THS shall provide information on collection of the tools if they don't qualify for local delivery.	Tool Hiring	N/A	Must
RQ19	The THS shall take a deposit from the customer at checkout to be refunded on return of the hired tools, if returned on time, clean and undamaged.	Tool Hiring	Customers	Must
RQ20	The THS shall allow the customer pay by credit card.	Tool Hiring	Customers	Must
RQ21	The THS shall ensure customers have accepted the terms and conditions before confirming tool rental.	Tool Hiring	Customers	Must

RQ22	The THS shall keep track of when tools are hired.	Tool Tracking	Admin	Must
RQ23	The THS shall allow a customer to track an order.	Tool Tracking	Customers	Should
RQ24	The THS shall send a reminder email to customers when their rental period is due to end.	Tool Tracking	Customers	Should
RQ25	The THS shall deduct an amount from the deposit before returning it to the customer if tools are returned late, dirty or damaged.	Tool Hiring	Customers	Must
RQ26	The THS shall allow a customer to write tool reviews.	Feedback	Customers	Should
HELP & ADVICE				
RQ27	The THS shall allow customers to receive advice on DIY problems by submitting a question on the service.	Help & Advice	Customers	Must
RQ28	The THS shall allow admin to respond to customer queries; tool fault queries shall be given highest priority than all other queries ranked in a first in first out manner.	Help & Advice	Admin	Must
RQ29	The THS shall allow admin to maintain a blog for DIY advice and tips.	Help & Advice	Admin	Should
RQ30	The THS shall allow a customer to view and sign up for weekly emails from the advice blog.	Help & Advice	Customers	Could
HEALTH & SAFETY / SECURITY				
RQ31	The THS shall check the ID of customers requesting to hire tools that have age restrictions.	Health & Safety	Customers	Must
NON-FUNCTIONAL REQUIREMENTS				
RQ32	The THS shall support all modern web browsers and both iOS and Android mobile devices.	Compliance to Standards	N/A	Must
RQ33	The THS shall be written in Java and conform to up-to-date object-oriented standards.	Compliance to Standards	N/A	Must
RQ34	The THS shall log in a user within 3 seconds.	Performance	All Users	Must
RQ35	The THS shall support 500 customer logins per minute.	Capacity	N/A	Must
RQ36	The THS shall support 5000 concurrent sessions.	Capacity	N/A	Must
RQ37	The THS shall support translation into 5 languages that represent the local area (English, Polish, Punjabi, Urdu, Bengali).	Compliance to Standards	N/A	Should
RQ38	The THS shall be accessible globally but hiring shall be restricted to England.	System Access	N/A	Must
RQ39	The THS shall support a maximum of 600 characters for the tool listing description.	Capacity	N/A	Must
RQ40	The THS shall ensure all logins are encrypted.	Security	N/A	Must
RQ41	The THS shall ensure all external communications with clients are encrypted.	Security	N/A	Must
RQ42	The THS shall be online 361 days of the year.	System Maintenance	Admin	Should
RQ43	The THS shall not be down for more than 1 hour during system failures.	System Maintenance	Admin	Should
RQ44	A THS shall be set up with a system manual.	System Maintenance	Admin	Should

Table 2.1: MoSCoW prioritization of requirements for the THS system

2.3 Domain Model Diagram

The domain model is used to represent the real-world actors and entities, as well as the relationships between them in the context of the problem space. It is a static model, in that it does not show any time or information flow. To develop the diagram, the actors and entities were derived from the requirements list, with nouns becoming entities and verbs becoming relationships. They were then used to help reduce ambiguities that may arise during the design period as well as managing the complexity [7]. Additionally, the actors and entities represented some possible classes for the later stages of the development.

The domain model diagram was finalized over a sequence of iterations, with the latest version seen in Figure 2.1 below. The actors, entities, and relationships between them are shown using UML notation. Once the entities were known, it was easier to construct the use cases more effectively.

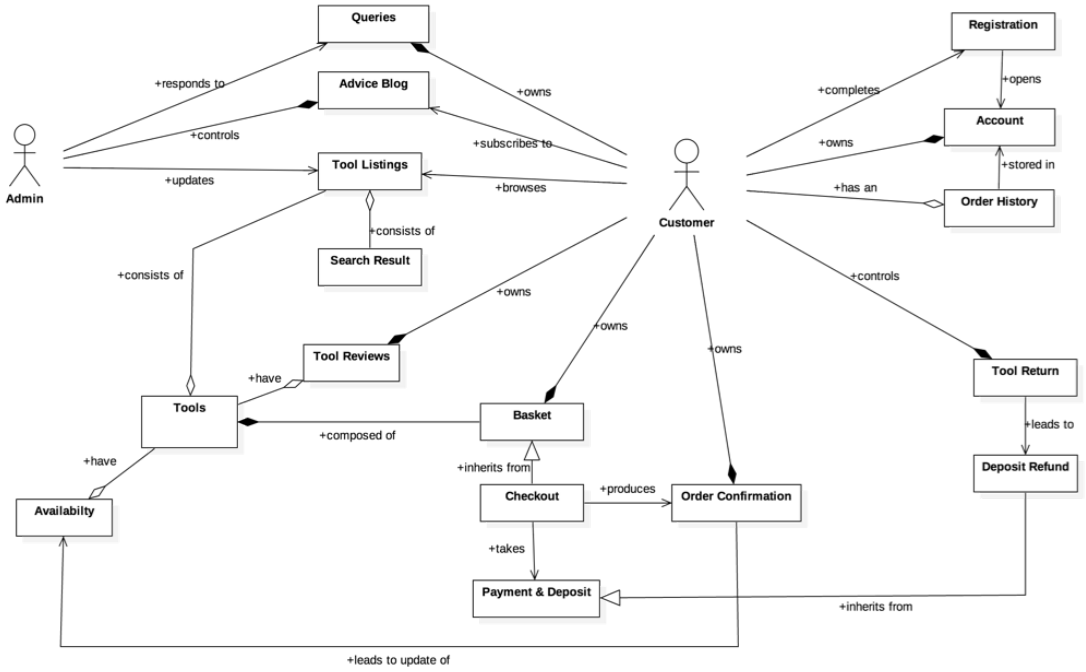


Figure 2.1: Domain model diagram

3 Use Cases

3.1 Overview

Having finished with the requirements gathering stage, the next task in the process involved developing and analysing a list of use cases. Each use case represents some interaction taking place between an actor and the DIY Tool Hire Service [8].

The use case analysis stage is essential to the development of a proper software system. It is vital that each of the requirements listed in Table 2.1 is accounted for when creating the use cases. This was accomplished using an iterative process, during which the requirements were expanded and modified, and the use cases were adapted to support those changes.

During the use case analysis period, the team completed the use case list, use case diagram, use case specifications, and the use case/requirements matrix. These tables and figures are found in the following subsections of this chapter.

3.2 Use Case Listing

When brainstorming the potential list of use cases, it was necessary for the team to determine who the actors would be when interacting with the DIY Tool Hire Service. The list of actors that was decided on based on the requirements came to: Customer, Admin, Credit Card Company, Notification Service, Customer Support, Database, and Delivery Service. With the requirements and these actors in mind, a list of use cases was established. Over the course of several iterations, the list was adapted until the final list below was developed.

ID	Use Case Name
UC1	Creates Account
UC2	Manages Account
UC3	Logs In
UC4	Forgets Password
UC5	Logs Out
UC6	Manages Tool Listing
UC7	Creates Order
UC8	Checks Out
UC9	Checks For Delivery Qualification
UC10	Validates Credit Card
UC11	Confirms Order
UC12	Updates Tool Availability
UC13	Tracks Order
UC14	Receives Tool
UC15	Sends Return Reminder
UC16	Returns Tool
UC17	Manages Tool Deposit
UC18	Reviews Tool
UC19	Raises Query
UC20	Responds to Queries
UC21	Reads Blog
UC22	Subscribes to Blog
UC23	Posts Advice Blog & Video Tutorials
UC24	Checks ID
UC25	Delivers Tool

Table 3.1: Use case listing

3.3 Use Case Diagram

The use case listing from above was translated into a use case diagram. The diagram presents the use cases in a more clearly illustrated manner. The team utilized the *«extend»* relationship as a way to add optional behaviours to certain use cases, as well as the *«include»* relationship to simplify larger use cases into more manageable parts [9].

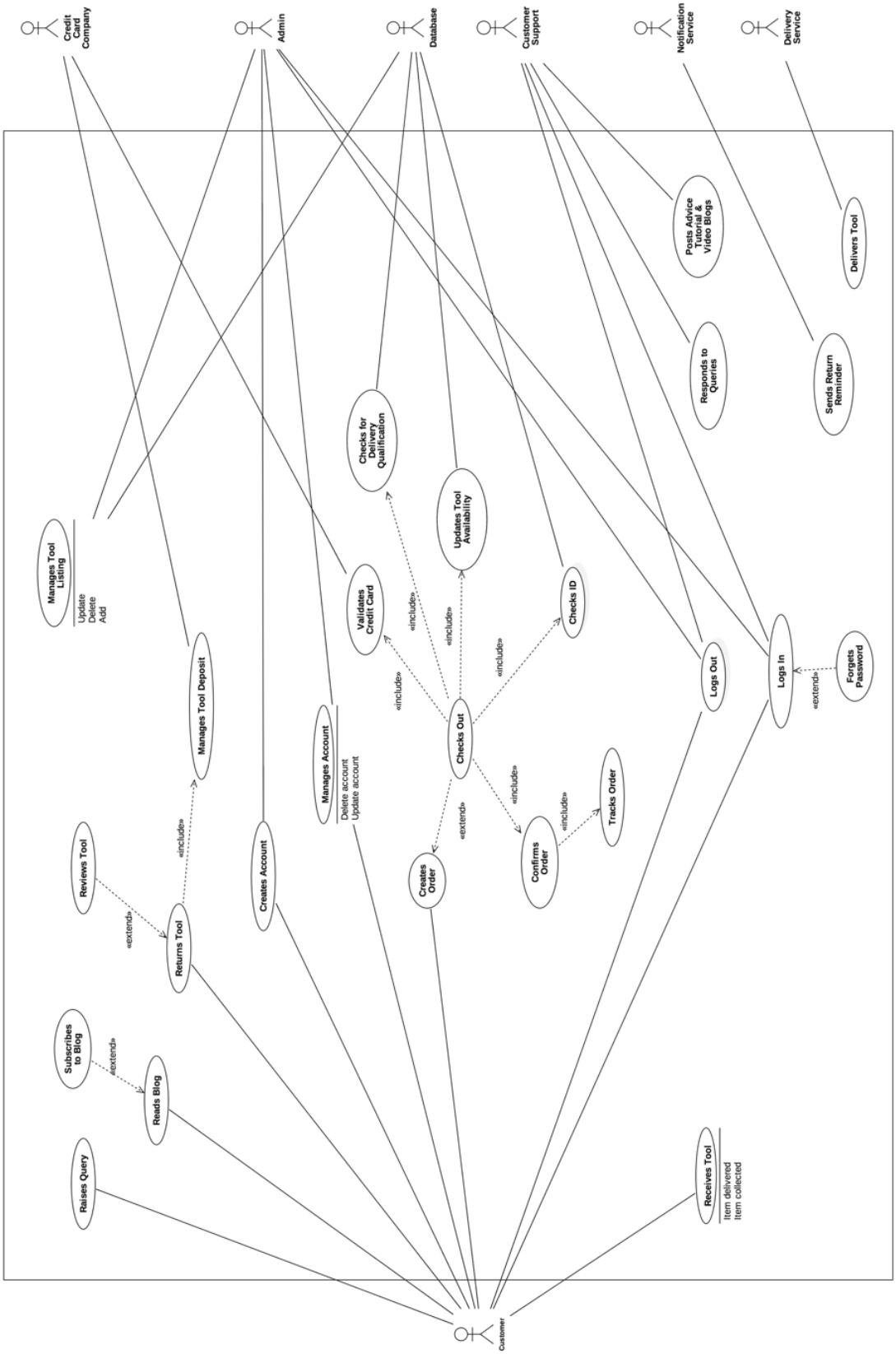


Figure 3.1: Use case diagram

3.4 Use Case Specifications

Continuing the development process, the team took the use cases depicted above and provided much more detailed information for each including a description, the actors involved, pre- and post-conditions, as well as the main and alternative flows. As the use case specifications were written, further alterations took place with the list of use cases. The list of use cases was refined, and the specifications continuously updated throughout the project as more thought went into the behaviours of the actors. The following specifications for the finalized list of use cases played a key role in the upcoming design period of the project [8].

Use-Case	**Creates Account**
Use-Case ID	UC1
Description	An admin may make an account for other employees, or a customer may make an account for himself/herself.
Primary Actors	Admin, Customer
Secondary Actors	Notification Service
Preconditions	
Main Flow	1. The use case starts when customers without an existing account click 'Create an Account', or the admin clicks 'Add User' on the THS's administration panel. 2. The THS redirects to the registration page. 3. The user must provide a unique email, first and last name, address, phone number, and alphanumeric password. 4. The user must then check off a box confirming that he/she is over the age of 18 and agrees to the THS's terms and conditions. 5. The user is informed that a confirmation email has been sent to the email address supplied and that he/she must click on the link in the email to conclude registration. 6. The notification service sends the email. 7. The user clicks on the link within the email and the system confirms the successful creation of the account.
Post Conditions	The account has been created.
Alternative Flow	1. Before proceeding to complete a transaction, the THS asks users to login or create an account. 2. Users without an existing account click on 'Create an Account'. 3. The THS redirects to the registration page. 4. The user must provide a unique email, first and last name, address, phone number, and alphanumeric password. 5. The THS displays a message if the supplied email already belongs to an existing account and prompts the user to change it. 6. The user clicks 'Cancel' and returns to the THS's main page.

Use-Case	Manages Account
Use-Case ID	UC2
Description	The admin and customer may update or delete his/her account.
Primary Actors	Admin, Customer
Secondary Actors	
Preconditions	The user is logged in.
Main Flow	1. The use case starts when the user clicks on 'Edit Account'. 2. The THS displays the user account information. 3. The user edits email, first/last name, address, contact number, or password OR deletes his/her account. An administrator can delete a customer account. 4. The user clicks 'Apply Changes'. 5. In case of editing email address, a confirmation email is sent to the address supplied and the user must click on the link sent to confirm the change.
Post Conditions	The changes to the account have been made.
Alternative Flow	1. The user accesses his/her account from the designated interface. 2. The user clicks 'Edit Account'. 3. The THS displays the user account information. 4. The user edits email, first/last name, address, contact number, or password. 5. The user clicks 'Apply Changes'. 6. The user supplied one of the fields with an invalid input. 7. The THS informs the user. 8. The user rectifies the error.

Use-Case	Logs In
Use-Case ID	UC3
Description	The user logs into his/her account.
Primary Actors	Customer, Admin, Customer Support
Secondary Actors	Database
Preconditions	The user has an account with the THS.
Main Flow	1. The use case starts when the user accesses the THS via the login page. 2. The user enters his/her email and password. 3. The THS checks the details against the database. 4. The user details are valid and the THS proceeds to the customer or admin home page.
Extensions	3.1 The user details are incorrect. 3.2 The user clicks 'Forgot Password'. 3.3 The THS sends the user an email containing a reset link. 3.4 The user selects a new password.
Post Conditions	The user has logged in.
Alternative Flow	1. The user accesses the THS via the login page. 2. The user enters his/her email and password. 3. The THS checks the details against the database. 4. The user details are invalid, and the THS informs the user.

Use-Case	Forgets Password
Use-Case ID	UC4
Description	The user forgets his/her password.
Primary Actors	Customer, Admin, Customer Support
Secondary Actors	
Preconditions	The user has already created an account.
Main Flow	1. The use case starts when the user clicks on 'Forgot Password' on the login page. 2. The THS redirects the user to a new page where he/she is asked to provide the email used the account creation. 3. The THS sends an email to the user with the link to reset the password. 4. The user clicks on the link and is redirected to a page where he/she can reset the password. 5. The THS confirms that the password is valid. 6. The THS redirects the user to the login page.
Post Conditions	The user's password has been reset.
Alternative Flow	N/A

Use-Case	Logs Out
Use-Case ID	UC5
Description	The user logs out of his/her account on the website.
Primary Actors	Customer, Admin, Customer Support
Secondary Actors	
Preconditions	The user has already logged into his/her account.
Main Flow	1. The use case starts when the user clicks the 'Log Out' button. 2. The user confirms that he/she would like to log out. 3. The logout is confirmed and the user is redirected to the login page.
Post Conditions	The user has logged out.
Alternative Flow	1. The user clicks on the 'Log Out' button. 2. The user presses cancel. 3. The log out is cancelled and the user remains logged in.

Use-Case	Manages Tool Listings
Use-Case ID	UC6
Description	The admin has the ability to update, add, or delete tool listings.
Primary Actors	Admin, Database
Secondary Actors	
Preconditions	The admin has the privileges to manage the listings.
Main Flow	1. The use case starts when the admin decides to edit the listings. 2. The THS will display a selection of all of the current listings that are available. 3. The admin then has the option to delete an existing listing, update the listing (with number of tools, description, etc.), or add a new tool listing. 4. The database will then update its information. 5. The THS will display the newly updated selection of the current listings.
Post Conditions	The listings page is updated.
Alternative Flow	1. The admin may go to a specific listing and select the 'update' option. 2. The THS will display the listing with its information made editable. 3. The admin can make the desired changes and click the 'submit' option. 4. The database will then update its information. 5. The THS will display the updated listing.

Use-Case	Creates Order
Use-Case ID	UC7
Description	The customer can search for tools, check their availability, and then add them to the basket.
Primary Actors	Customer
Secondary Actors	Database
Preconditions	
Main Flow	1. The use case starts when the customer clicks on the option to show all of the tools offered by the THS. 2. The THS will automatically display the tools in order of popularity. 3. The customer may select the option to sort the tools by type, name, or project to make their search easier. 3.1 The THS will update the display with tools that fall under the category of the selected type, name, or project at the top. 4. Once the customer has found the tool he/she is looking for and clicked on it, the THS will display the tool's description, rates, and a calendar with its availability. 5. The customer can look at the calendar to see if the tool is available on the desired date(s). 6. If the tool is available on the desired date, the customer may select it. Otherwise, he/she may select another date. 7. The customer may then click 'Add to Basket'. 8. The THS will then display the tool in the basket.
Extensions	8.1 The customer may then click 'Check Out'. 8.2 The THS will direct the customer to another page where he/she may see if his/her address qualifies for delivery and enter the payment information. 8.3 The customer may then click 'Confirm Order' to submit the payment and finalize the order.
Post Conditions	The customer has selected a tool and added it to his/her basket.
Alternative Flow	1. The customer may select one of the tabs listing the different tool types, names, or projects. 2. The THS will then display tools from the selected category. 3. Once the customer has found the tool he/she is looking for and clicked on it, the THS will display the tool's description, rates, and a calendar with its availability. 4. The customer can look at the calendar to see if the tool is available on the desired date(s). 5. If the tool is available on the desired date, the customer may select it. Otherwise, he/she may select another date. 6. The customer may then click 'Add to Basket'. 7. The THS will then display the tool in the basket.

Use-Case	Checks Out
Use-Case ID	UC8
Description	The customer may pay for and rent the tools.
Primary Actors	Customer
Secondary Actors	
Preconditions	The customer has tools in the basket.
Main Flow	1. The use case starts when the customer is looking at the basket and there are tools contained in it. 2. The customer then clicks 'Check Out'. **Includes:: Checks ID** 3. The THS will redirect the customer to the next page where he/she will enter payment details and select delivery or pickup. **Includes:: Check for Delivery Qualification** 4. After the customer has decided whether the tools will be delivered or picked up, he/she must check off the box accepting the rental agreement before clicking 'Confirm Order'. 5. Once the customer has entered all of the details, he/she will click 'Submit Order'. **Includes:: Validate Credit Card** **Includes:: Updates Tool Availability** **Includes:: Confirms Order**
Post Conditions	The tools have been rented.
Alternative Flow	N/A

Use-Case	Checks for Delivery Qualification
Use-Case ID	UC9
Description	The customer checks to see if his/her zip code qualifies for delivery.
Primary Actors	Customer, Database
Secondary Actors	
Preconditions	The customer has added tools to the basket and is in the process of checking out.
Main Flow	1. The use case starts when the user enters his/her zip code and clicks 'Check for Delivery'. 2. The database will check if the zip code is contained among those approved for delivery. 3. The THS will notify the customer if he/she qualifies for delivery or not.
Post Conditions	The customer knows if the tool will be delivered or must be picked up.
Alternative Flow	N/A

Use-Case	Validates Credit Card
Use-Case ID	UC10
Description	When a customer is checking out, his/her credit card details are validated by the credit card company.
Primary Actors	Credit Card Company
Secondary Actors	Customer
Preconditions	The customer has clicked 'Check Out'.
Main Flow	1. The use case starts when the customer clicks 'Check Out'. 2. The credit card details are assessed by the credit card company. 3. If the details are correct, the payment goes through. 4. If the details are incorrect, the THS displays that the payment has failed.
Post Conditions	The customer knows if his/her credit card has been accepted.
Alternative Flow	N/A

Use-Case	Confirms Order
Use-Case ID	UC11
Description	The THS confirms the customer's order.
Primary Actors	Customer
Secondary Actors	Notification Service
Preconditions	The customer has completed all parts of the checkout process.
Main Flow	1. The use case starts when the customer clicks on the 'Submit Order' button. 2. The THS will direct the customer to a confirmation page displaying the order ID number and information about the order. 3. The notification service will send the customer an email containing the order confirmation details. 3.1 If the customer selected delivery, the confirmation email will let him/her know another email will be sent with tracking information once the tool is shipped. 3.2 If the customer selected pick up, the confirmation email will let him/her know another email will be sent with pick-up information once the tool is ready in the warehouse. **Includes:: Tracks Order**
Post Conditions	The customer's order is completed, and the customer receives confirmation.
Alternative Flow	N/A

Use-Case	Updates Tool Availability
Use-Case ID	UC12
Description	The THS will update the availability of a tool.
Primary Actors	Customer, Database
Secondary Actors	
Preconditions	The customer has clicked 'Check Out'.
Main Flow	1. The use case starts when a customer has selected a tool and completed the checkout process. 2. The database will update the information about the tool's availability based on the dates it has been rented. 3. The THS will then display the updated availability on the listing.
Post Conditions	The THS will display the updated availability of the tool for other customers.
Alternative Flow	N/A

Use-Case	Tracks Order
Use-Case ID	UC13
Description	The customer can track his/her order if it is shipped.
Primary Actors	Customer
Secondary Actors	Notification Service
Preconditions	The customer has confirmed an order.
Main Flow	1. The use case begins when the customer's order has shipped or has been returned to the warehouse. 2. The notification service will send the user an email containing details about the order. 2.1 If the customer selected pick-up, the email will let him/her know the tool is ready for pick-up in the warehouse. 2.2 If the customer selected delivery, the email will contain tracking details for the order. 3. The customer can click on the tracking number. 4. The customer will then be redirected to the page containing the details of the order's location.
Post Conditions	The customer knows where his/her order is currently located.
Alternative Flow	1. The customer clicks on the 'Account' button. 2. Then the customer clicks on the 'Orders' button. 3. The THS will display the list of the customer's orders. 4. Then, the customer can click on the tracking number for the desired order.

Use-Case	Receives Tool
Use-Case ID	UC14
Description	The customer may have the tool delivered if he/she qualifies or may come collect the tool.
Primary Actors	Customer
Secondary Actors	
Preconditions	The customer has rented a tool.
Main Flow	1. The use case starts when the customer checks out and rents one or more tools. 2. Depending on what option the customer selected during check out, the customer may have the tool delivered or pick it up.
Post Conditions	The customer has possession of the rented tool.
Alternative Flow	N/A

Use-Case	Sends Return Reminder
Use-Case ID	UC15
Description	The THS notification service will send automated emails to the customer two days before, one day before, and on the day that the tool is due to be returned.
Primary Actors	Notification Service
Secondary Actors	
Preconditions	Customer has rented a tool and it has been delivered or collected.
Main Flow	1. The use case starts when the THS notification service runs its daily database check to see which tools are due to be returned. 2. The THS notification service then sends automated emails to all customers whose tools must be returned within the next two days. 3. The customer will then see the email as a reminder in case he/she forgot that the tool needed to be returned.
Post Conditions	The customer can view the reminder.
Alternative Flow	N/A

Use-Case	Returns Tool
Use-Case ID	UC16
Description	The customer returns the rented tool.
Primary Actors	Customer
Secondary Actors	
Preconditions	The customer has rented the tool and it has been delivered.
Main Flow	1. The use case starts when the customer has finished using the tool or has reached the end of the rental period, whichever comes first. 2. The customer brings the tool back to the THS. 3. The THS registers that the tool has been returned. **Include:: Manages Tool Deposit** 4. The notification service sends the customer an email to confirm the return.
Extensions	4.1 The customer is particularly happy or unhappy with the tool or the THs. 4.2 The customer clicks on the link in the email sent by the notification service to write a review. 4.3 The customer writes a review and clicks 'Submit'. 4.4 The THS now displays the review.
Post Conditions	The tool is now back with the THS.
Alternative Flow	N/A

Use-Case	Manages Tool Deposit
Use-Case ID	UC17
Description	The THS will let the credit card company know if the customer's deposit on the tool should be returned or not.
Primary Actors	Credit Card Company
Secondary Actors	Notification Service
Preconditions	The customer has returned the tools that were rented.
Main Flow	1. The use case starts when the tools are examined by an admin of the THS. 2. The admin decides the tools are in good condition. 3. The admin then tells the credit card company to return the customer's deposit. 4. The notification service sends the customer an email to inform him/her that the deposit has been returned.
Post Conditions	The customer's deposit will either be returned or kept by the THS, and he/she will be informed.
Alternative Flow	1. The tools are examined by an admin of the THS. 2. The admin decides the tools are in bad condition. 3. The admin then tells the credit card company to transfer the customer's deposit to the THS. 4. The notification service sends the customer an email to inform him/her that the deposit will not be returned and explains the reason why.

Use-Case	Reviews Tool
Use-Case ID	UC18
Description	The customer writes a review about a tool.
Primary Actors	Customer
Secondary Actors	Notification Service
Preconditions	The customer has previously rented the tool about which he/she wants to write a review.
Main Flow	1. The use case starts when the customer returns the tool(s) he/she rented. 2. The notification service will then send the customer an email thanking him/her for using the THS and including a link to review the tool(s). 3. The customer can then click on the link. 4. The THS will redirect the customer to a page listing the tools that were rented and a place to write reviews about each of them. 5. The customer then writes a review about any of the tools he/she rented. 6. The customer can then click 'Submit'. 7. The THS will then display the new review along with any existing reviews whenever a customer looks at that specific tool.
Post Conditions	The review appears on the THS for future customers to see.
Alternative Flow	N/A

Use-Case	Raises Query
Use-Case ID	UC19
Description	The customer can submit a query to the customer support with any questions he/she may have.
Primary Actors	Customer
Secondary Actors	
Preconditions	The customer is logged into his/her account.
Main Flow	1. The use case starts the customer clicks on the 'Support' button. 2. The customer will then be redirected to the customer support page. 3. The customer will then be directed to choose an option from the categories for his/her query. 4. The customer then fills in the message box with his/her question. 5. The customer then clicks 'Submit'. 6. The query will then be sent to the customer support who will answer it as soon as possible.
Post Conditions	The customer support will see the queries raised by the customer.
Alternative Flow	N/A

Use-Case	Responds to Queries
Use-Case ID	UC20
Description	The customer support answers any queries that have been submitted by the customers.
Primary Actors	Customer Support
Secondary Actors	
Preconditions	The customer has made a query to be answered by the customer support.
Main Flow	1. The use case starts when the customer support views the queries. 2. The queries are organized by priority – high priority for fault reporting, otherwise low priority. 3. The customer support will answer the queries, in order, and send an email to the customer containing the response. The response is also shown on their THS account.
Post Conditions	The customers can view the responses to their queries.
Alternative Flow	N/A

Use-Case	Reads Blog
Use-Case ID	UC21
Description	The customer can read the advice blog.
Primary Actors	Customer
Secondary Actors	
Preconditions	The customer is on the THS or has subscribed to the blog already.
Main Flow	1. The use case starts when the customer clicks on the 'Advice' section. 2. The THS redirects the customer to the page containing the DIY blog and the video tutorials. 3. The customer then clicks on the DIY Blog. 4. The THS redirects the customer to the blog. 5. The customer can now read the different blog posts.
Extensions	5.1 The customer may enter his/her email into the subscribe option. 5.2 The notification service will send the customer weekly emails about the blog entries.
Post Conditions	The customer has the information provided within the blog posts.
Alternative Flow	1. The notification service sends the customer an email about a new blog entry. 2. The customer clicks on the link to take him/her to the blog. 3. The customer may read the new entry.

Use-Case	Subscribes to Blog
Use-Case ID	UC22
Description	The customer subscribes to the weekly advice blog.
Primary Actors	Customer
Secondary Actors	Notification Service
Preconditions	The customer has opened up the blog.
Main Flow	1. The use case starts when the customer is viewing the DIY advice blog section of the THS. 2. The THS will display an option to the customer, allowing him/her to sign up for weekly tips from the company. 3. The customer subscribes to the blog by entering his/her email as prompted by the THS.
Post Conditions	The customer will receive notifications about weekly blog entries from the notification service.
Alternative Flow	N/A

Use-Case	Posts Advice Tutorial & Video Blogs
Use-Case ID	UC23
Description	The customer support can post video tutorials and blog posts on the website to offer advice to the customers.
Primary Actors	Customer Support
Secondary Actors	
Preconditions	
Main Flow	1. The use case starts when the customer support clicks on the "Advice" section of the website. 2. The THS will display the page where the customer support may choose to upload a video or write a blog post. 3. Then the customer support can upload the desired video or write the blog post on the specific topic. 4. The THS will then display this new video or post to the customers when they click on the "Advice" section.
Post Conditions	The customers can see the tutorials and blog posts.
Alternative Flow	N/A

Use-Case	Checks ID
Use-Case ID	UC24
Description	The THS verifies the ID of the customer and keeps track of his/her age.
Primary Actors	Database
Secondary Actors	
Preconditions	The customer is in the process of checking out.
Main Flow	1. The use case starts when the customer begins the checkout process. 2. The THS will check the customer's details against the database. 3. The details in the database show that the customer is old enough to rent the desired tool. 4. The THS will display the approval and the customer will be able to continue with the checkout.
Post Conditions	The customer is informed if he/she is able to rent the tool.
Alternative Flow	1. The customer begins the checkout process. 2. The THS will check the customer's details against the database. 3. The details in the database show that the customer is not old enough to rent the desired tool. 4. The THS will display the restriction, and the customer will not be able to rent the tool.

Use-Case	Delivers Tool
Use-Case ID	UC25
Description	The rented tools are delivered to the customer.
Primary Actors	Delivery Service
Secondary Actors	
Preconditions	The tools have been rented and the delivery option selected.
Main Flow	1. The use case starts when the delivery service is notified that there is a tool to be delivered. 2. The delivery service picks up the tool from the warehouse. 3. The delivery service then brings the tool the to the customer's address. 4. If it is a small tool, it is left at the customer's door, but for the larger expensive tools, the customer must sign off on the delivery.
Post Conditions	The tool has been delivered.
Alternative Flow	N/A

3.5 Use Case/Requirements Matrix

The last task to accomplish during the use case analysis stage was to create a use case/requirements matrix. This matrix, which displays the requirements on the left side and the use cases across the top, is intended to confirm that each of the requirements is being satisfied by one of the use cases and that each use case serves the purpose of meeting a requirement [4]. As seen below, the requirements and use cases established by the team satisfy these conditions.

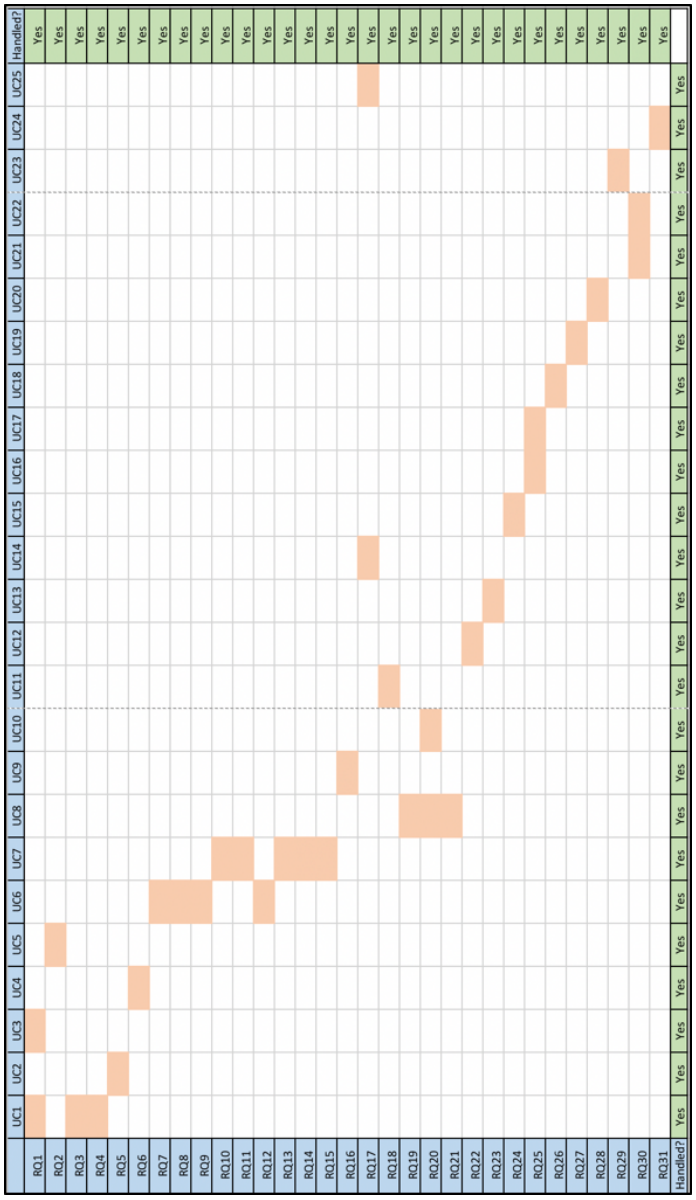


Figure 3.2: Use case/Requirements matrix to describe how use cases fulfill system requirements

4 Object-Oriented Analysis and Design

4.1 Overview

With the functional requirements of the THS established and use cases developed to model the user interaction with the system, it was necessary to begin to move from the problem domain to the solution domain and move more towards the language of the developer, structured by classes and packages rather than use cases. [1]

In doing so, the domain model and use cases were analysed to identify the key objects to be modelled in the system. With the objects identified, the data attributes of the objects and method call sequences required to realise the use cases were established through an iterative process of role play and pseudo-code development. This resulted in an analysis class diagram - showing the key objects and their relationships, a design class diagram – a more comprehensive and defined representation of the system's classes, sequence and activity diagrams, state machine diagrams and a deployment diagram, which will be discussed in the following sections.

4.2 Activity Diagrams

Activity diagrams help to model system processes through a series of activities. They have convenient notation, similar to flow charts and can be thought of as Object Oriented flowcharts. [9] Unlike class diagrams or sequence diagrams, actions are written in plain English as these diagrams serve as predecessors to more complex, class-oriented models.

The THS team used activity diagrams to model complete processes including both software and real-world activities, and in doing so indicated the relationship between such activities and how they are carried out to fulfill system requirements. Included in these representations are start and stop states, indicating the beginning and end of an activity and forks and joins, to illustrate when activity states occur concurrently. [9] Alternative flows are modeled by decisions, represented by a diamond, and indicate different flow directions that can be taked based on a certain condition. The final iterations of activity diagrams for the THS are shown below as well as a key to activity diagram syntax.

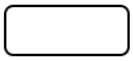
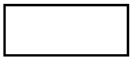
	The initial node: This indicates the start of the activity and automatically transitions to the first activity state
	The final node: This indicated that the activity has been completed
	Decision: Like an if-else statement, decisions indicated the different actions that are taken depending on a condition, labelled like: **[condition].**
	Actions: these are the steps from the start to the end of the activity. These can represent a computational action or a physical action.
	Forks and Joins: These are used to indicated when several actions occur concurrently.
	Time events: These are used to model a period of waiting time.
	Edges: There are incoming and outgoing edges from each activity. These are used to indicated the direction of flow of the actions.
	Objects: Rectangles are used to indicate a software object.
	Signal and Signal Received: There are no actions associated with signals sent and received, they simply indicate that the signal has been sent and received.

Figure 4.1: Key for activity diagram symbols

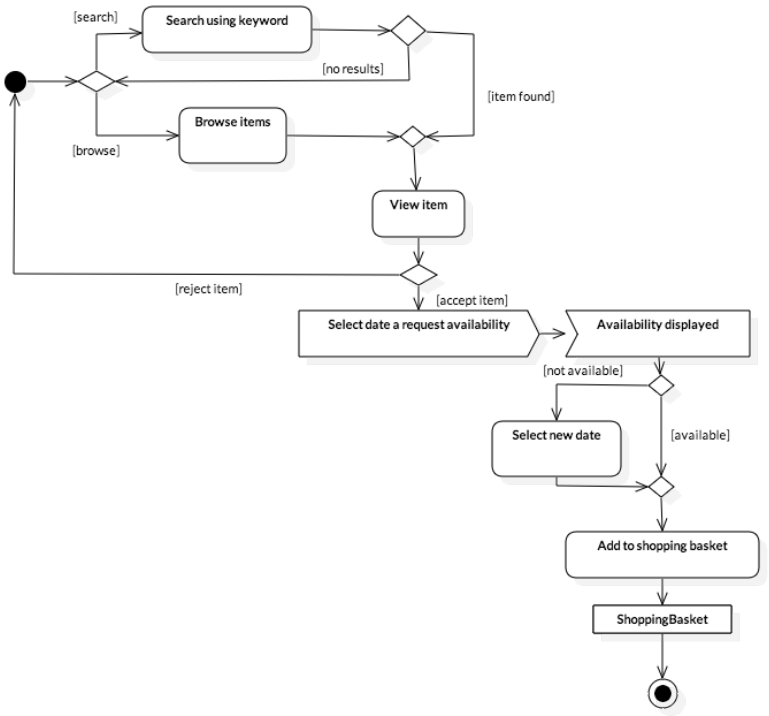


Figure 4.2: Activity diagram for UC7:Creates Order

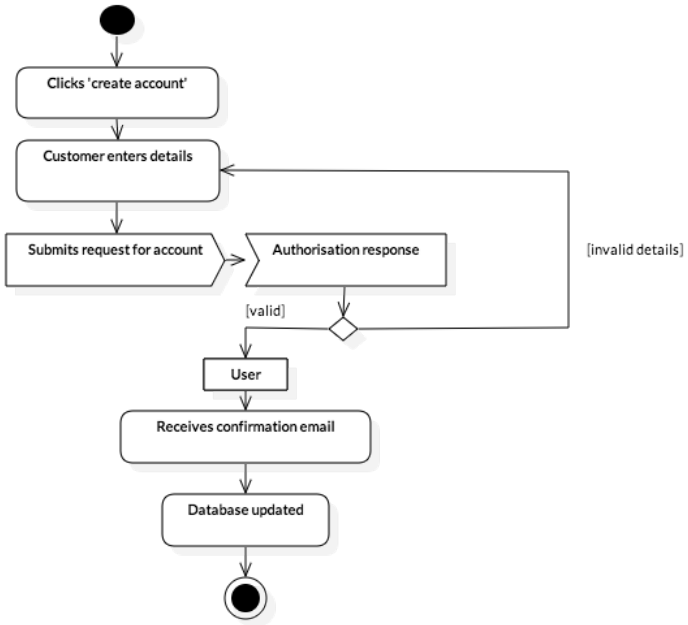


Figure 4.3: Activity diagram for UC1: Creates Account

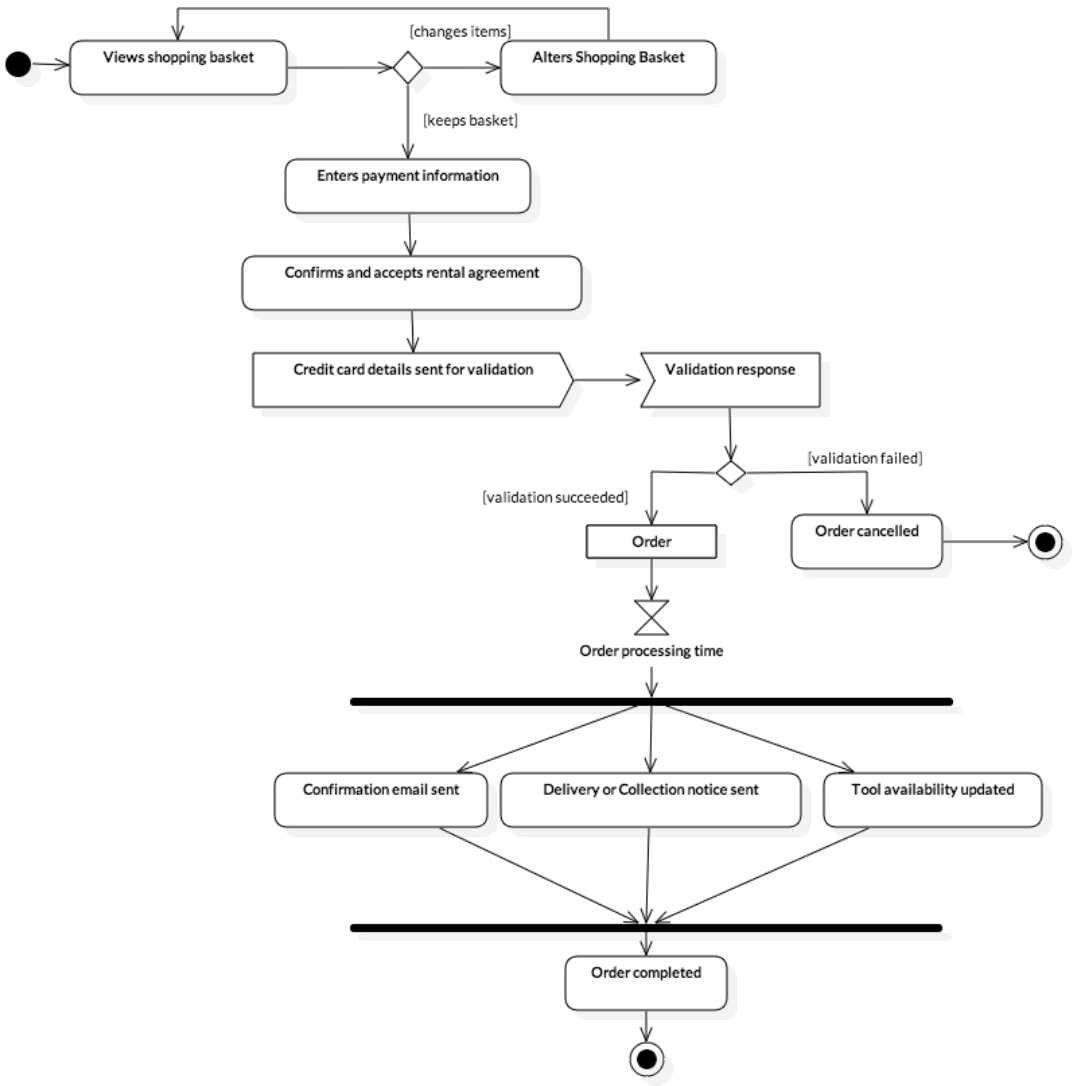


Figure 4.4: Activity diagram for UC8: Checks Out

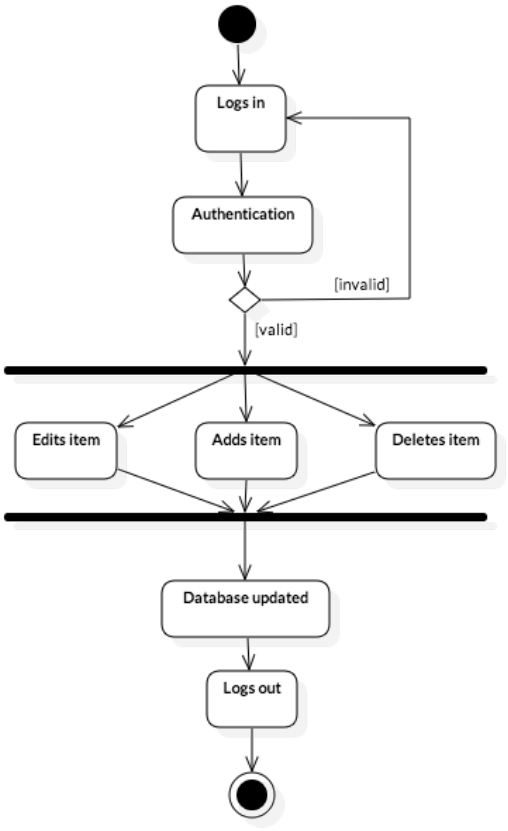


Figure 4.5: Activity diagram for UC6: Manages Tool Listings

4.3 UML Class Diagrams

4.3.1 Analysis Class Diagram

A use case driven approach was used to identify the key nouns and verbs and iteratively develop these into the system's objects and method calls. The analysis class diagram in Figure 4.7 shows a middle stage iteration. The classes are labelled as either boundary, controller or entity classes in line with the Jacobson categorisation style [10]. In this style, the boundary classes encapsulate output data and generates output in the required form, the controller classes deal with controlling behaviour and completing tasks and the entities classes store the data and define operations on the data. The advantage of this is that it enforces clear partitioning of responsibilities in the class structure early on in the object-oriented design process.

The class names were kept close to the functionality of the class which will allow for clarity in the development phase. The relationships (associations, generalisations, dependencies, ag-

gregations and compositions) were mapped between the classes only where a relationship was necessary to achieve a use case.

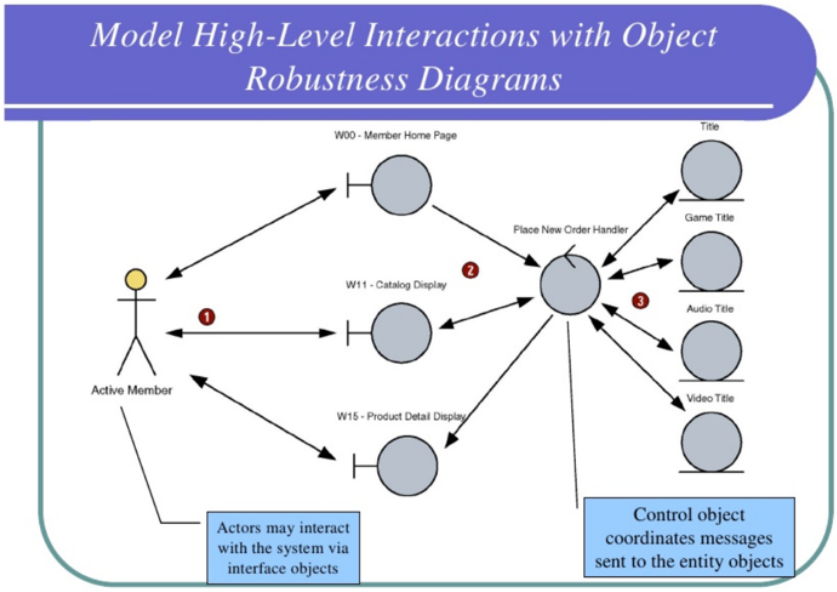


Figure 4.6: A model of Boundary-Control-Entity architecture [11]

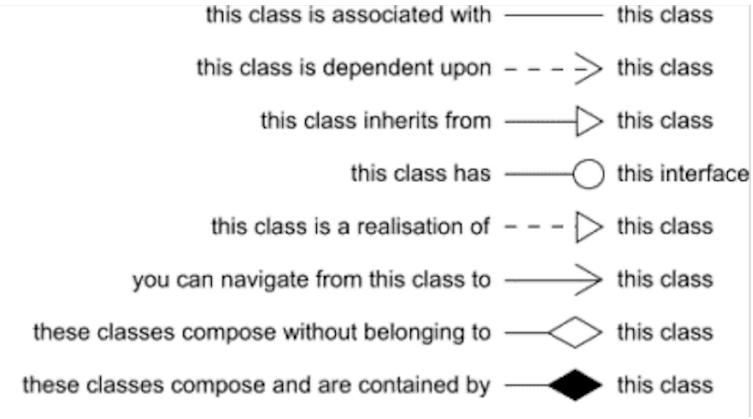


Figure 4.7: Class diagram key

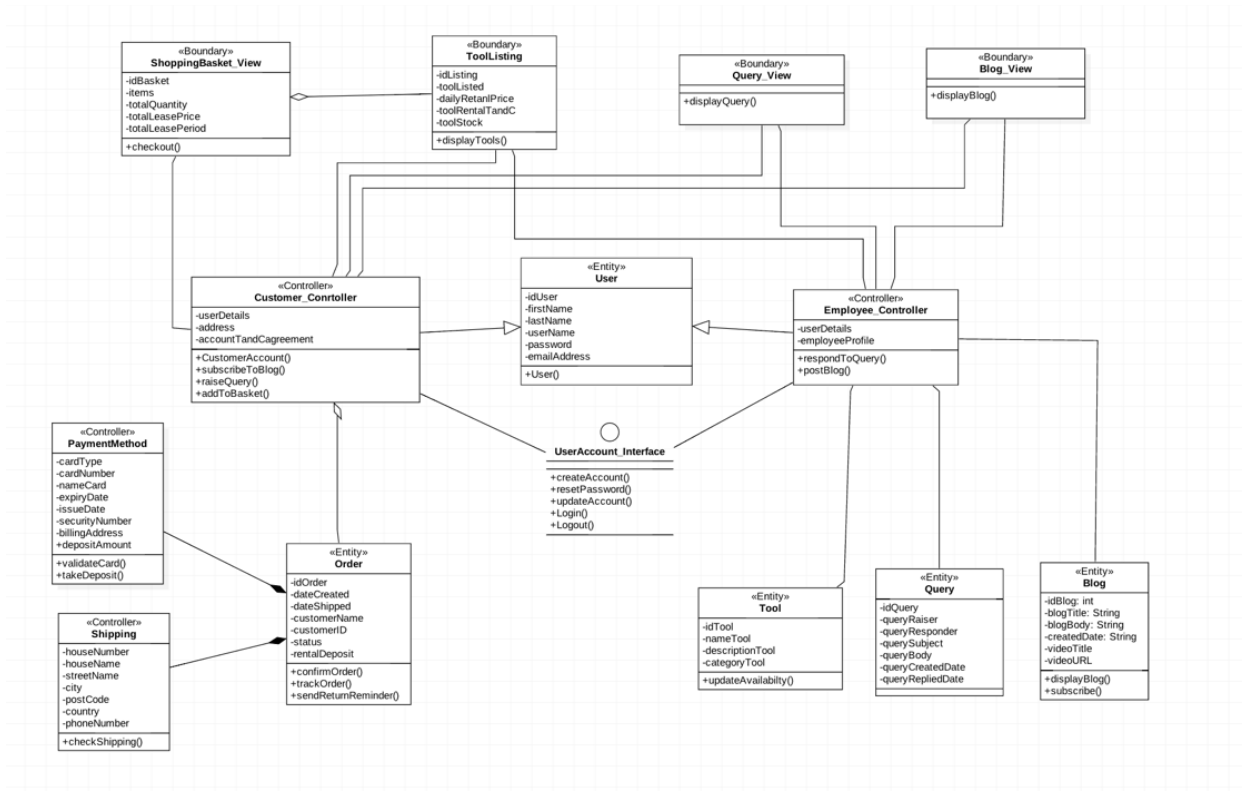


Figure 4.8: Analysis Class Diagram

4.3.2 Design Class Diagram

The analysis diagram was used as a basis to build the design class diagram, which specifies in more detail how the system will be implemented, getting closer to the solution domain. There was a strong focus on adhering to object-oriented principles. Some of the considerations that went into the design class diagram are discussed here:

Adding Implementation Details

Variable types, method parameters and return types were added, along with variable and method visibility. Adhering the object-oriented principle of **encapsulation** most class member variables were given private visibility with public methods that allow access to them. (NB for the sake of coherence of the diagram, not all getters and setter were modelled for every class).

Static methods were also used where there was no need for the method to use instance variables and the method therefore belongs to the class rather than an instance of the class. For example - the methods in the database connector are static, as is the 'connectionStatus' variable. This

makes the class a **singleton class** [12] whereby only one instance of the class need exist in the Java Virtual Machine and providing a global point of access to other classes that need it.

Inheritance was used only where needed – to generalise a customer and employee to a user – thereby eliminating duplication of user details in each class. It was also used in the 'Payment-Method' (parent) and 'Deposit' (child) relationship.

Interfaces were used to help separate the specification of an entity from implementation of a task. For example - the employee controller manages a tool listing via a 'manageToolListing' interface. This has the advantage that in the future, should the GUI for the tool listing change, it is easier to change the implementation of the 'manage tool listing' methods.

Completeness & Primitiveness

Completeness: this is the specification that the design classes should cover all the data and functionality needed. We ensured this by iteratively mapping out use case flows and ensuring that classes are related and can therefore access instances of each other when required. Classes were also added from the analysis class where needed.

Primitiveness: this is the specification that there should never be more than one way of doing things. For example – in our system there is just a single 'respondToQuery()' method that encompasses different types of query rather than several separate methods.

High Cohesion & Low Coupling

High cohesion: this specifies that each class should model a single abstract concept [9]. We ensured this by separating out the implementation for managing queries/blog/tool listing/shopping basket/user accounts – so that the employee and customer controllers do not take on several different responsibilities. This is also the case for the database connector which is a simple class, just concerned with connecting to the database, the create/read/update methods are present in the respective entities.

Low coupling: This specifies that classes should associate with just enough other classes to realise its goal – e.g. which was achieved by using focused aggregation rather than inheritance in most places.

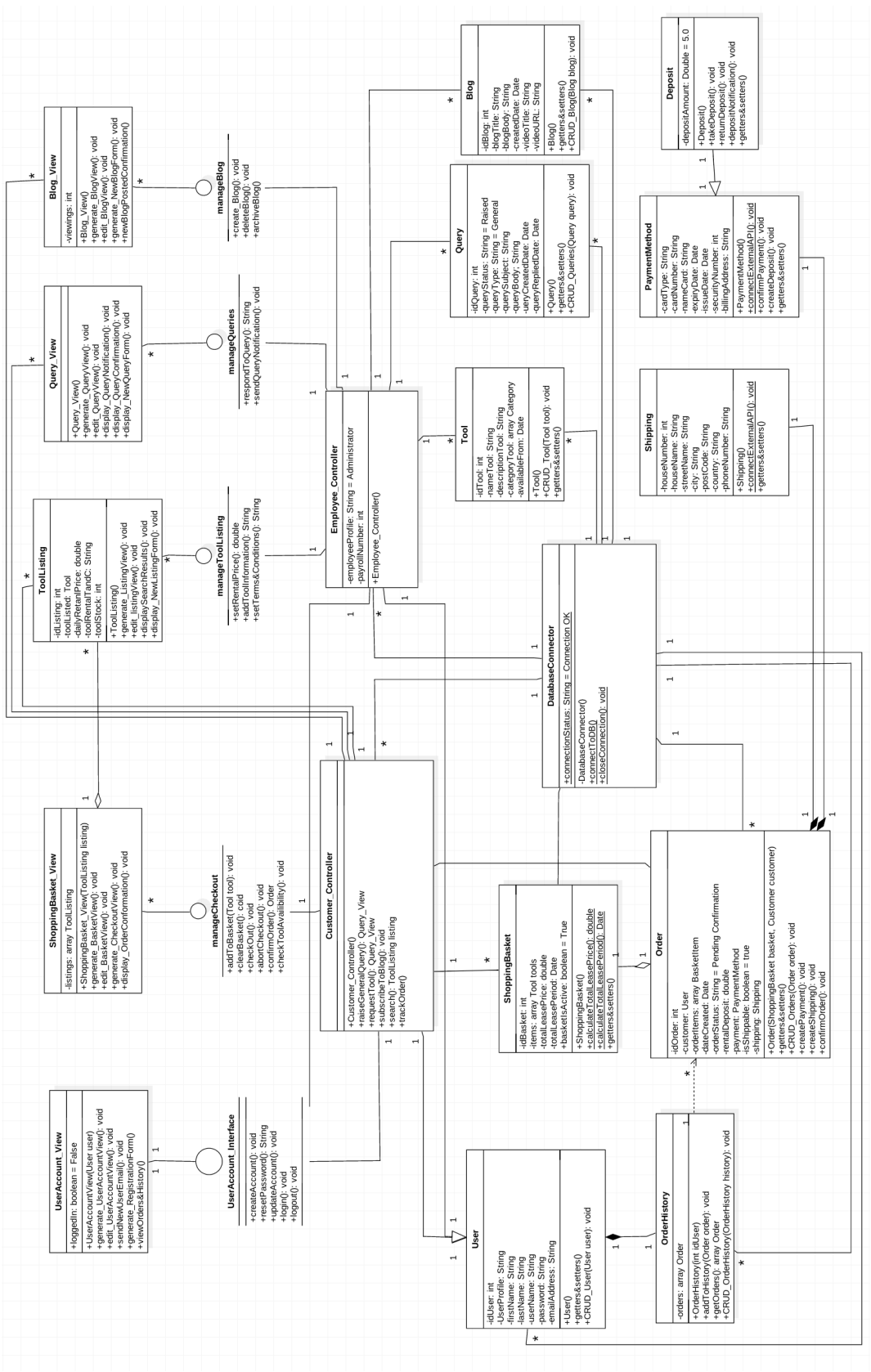


Figure 4.9: Design Class Diagram

Below, there is a class table describing the functionality of the classes in this model.

Class Name	Function
User	This class instantiates a user object. It has attributes that represent the user details. A user can be a customer or employee so this class is the super class of Customer_Controller and Employee_Controller
Customer_Controller	This class is a controller class and represents a customer. It manages the required functions of the customer – by interacting with the boundary classes and entities. It has only a few key functions (primitiveness) with many functions being implemented by helper interfaces/classes.
Employee_Controller	This class is a controller class and represents an employee. It manages the required functions of the employee – by interacting with the boundary classes and entities. It co-ordinates functions through helper interfaces/classes (primitiveness).
DatabaseConnector	This class provides the connection to the database. It has only a couple of static functions to connect and close the connection, and a connection status attribute. Therefore, it is a singleton class – most create/read/update/delete functions for the database are contained within the entities that the DatabseConnector is associated with.
UserAccount_Interface	This is the interface that Customer_Controller and Employee_Controller implement to manage their account views. It ensures all users can carry out necessary functions such as login.
UserAccount_View	This is a boundary class that controls the front-end view of the user's account.
manageCheckout	This is the interface that Customer_Controller implements to manage their shopping basket views. It ensures customers can carry out necessary functions in the front end to manage the items in their shopping basket and the checkout process.
ShoppingBasket_View	This is a boundary class that controls the front-end view of the shopping basket. As well as generating the view of the basket (which contains tool listings) it manages the view through the checkout process and to order confirmation.
ShoppingBasket	This is an entity class representing the customer's shopping basket. It aggregates with Order class – as an Order is created with the contents of a shopping basket.
Order	This is an entity class representing the customer's Order. Orders are controlled via the database and the Customer_Controller classes.

OrderHistory	This is an entity class representing the customer's Order History. Each user instance is linked to an order history instance through composition.
Shipping	This is an entity class representing the shipping information which is linked to the Order class via composition. The static method connectExternalAPI() links the system to a 3rd party shipping system which manages the shipping of the order.
PaymentMethod	This is an entity class representing the payment information which is linked to the Order class via composition. The static method connectExternalAPI() links the system to a 3rd party payment system (e.g. CC company, PayPal) to validate and take payment.
Deposit	This is an entity which is a child class of PaymentMethod. It holds the information relating to the customer's deposit.
manageToolListing	This is the interface that Employee_Controller implements to manage the tool listing view.
ToolListing	This is a boundary class that controls the front-end view of the tool listing. A tool listing is view that presents information about tools to hire to the user.
Tool	This is the entity class that represents the tools being hired. Tools are controlled via the database and the Employee_Controller classes.
manageQueries	This is the interface that Employee_Controller implements to manage the query view e.g. to send query notifications and respons to queries posted by customers.
Query_View	This is a boundary class that controls the front-end view of a query.
Query	This is the entity class that represents a query. A query is either a general query e.g. a question about a tool or a customer request for a tool. The customer_controller usually creates the instance of Query and the employee_controller responds.
manageBlog	This is the interface that Employee_Controller implements to manage the blog view.
Blog_View	This is a boundary class that controls the front-end view of a blog.
Blog	This is the entity class that represents a blog. A blog is either a written informational entry with advice/tips for the customers or a video.

Table 4.1: Class table describing the funtionality of classes in the THS system.

Additionally, earlier iterations of the design class diagram can be seen in Appendix 8.2.

4.4 Sequence Diagrams

Sequence diagrams serve as the bridge between use cases and class diagrams. They are a type of interaction diagram which describes the order in which interactions in a system occur. The

events in sequence diagram are shown chronologically, with the earliest event at the top of the diagram and the last event at the bottom. Events occur between participants, which in the case of the THS, are actors, databases, and classes derived from the class diagram. Each event has a message which describes the interaction between participants. Methods from the design class diagram are used as messages and allow the classes to interact. Interactions between the actors and boundaries have messages given in pseudo code to describe the real-world interaction between the user and the interface. [9]

For the purposes of this project, messages were given without arguments. Given that team determined that the diagrams were useful and adequately descriptive using this high level approach, the arguments were intentionally left out to reduce clutter. Six sequence diagrams are given below to illustrate some of the most relevant use cases in addition to a key describing the features that appear. For early iterations of key sequence diagrams, please see Appendix 8.3.

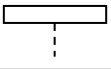
	Participant (class): This is a part of the system, in this case a class, that interacts with other participants to achieve a goal. The dotted line (lifeline) represents when a participant exists in the sequence diagram.
	Participant (actor): Like a class participant, an actor participant interacts with participants in the system in order to carry out a use case. These actors appear in the use-case diagram.
	Participant (boundary): Boundary participants represent the first tier with which users interact with the THS system. They serve as the bridge between the actors and the other participants.
	Participant (database): A database participant represents THS data that is stored in the back-end of the system.
	Activation Bars: After a message has been sent to a participant, it is said to be active. Activation bars indicate when a participant is active.
<<creates>>	Creates notation: This notation appears when a participant creates another participant in a sequence. The participant that has been created now has a lifeline.
X	Destroy notation: This notation appears when a participant is destroyed in a sequence. The lifeline for this participant ends when it it destroyed.
	Synchronous arrow: Arrows appear with messages above them. A synchronous message waits for the receiver to carry out an action before carrying on with further steps in the interaction.
	Return messages: These indicate the reply of a participant after a message has been sent to it. They are not always necessary, but they can be useful in illustrating the result of important interactions.

Figure 4.10: Sequence diagram symbol key

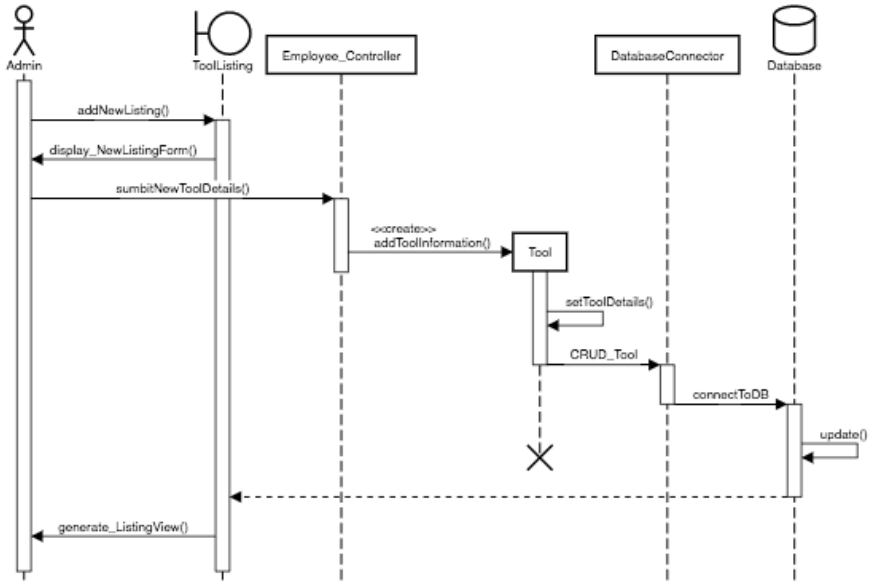


Figure 4.11: Sequence diagram for adding a new tool listing to the THS system. This is a version of UC6: Manages Tool Listing

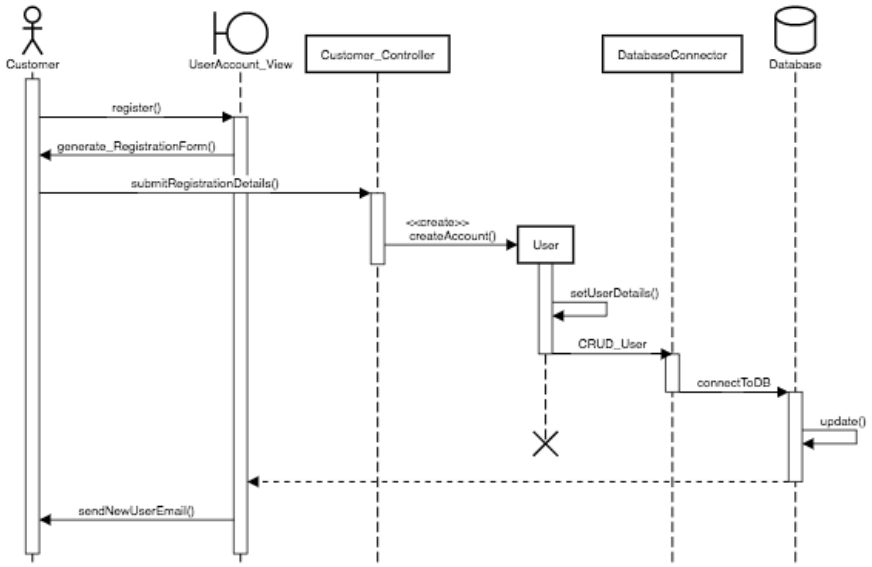


Figure 4.12: Sequence diagram for registering a new customer to the THS system. This is a version of UC1: Creates Account

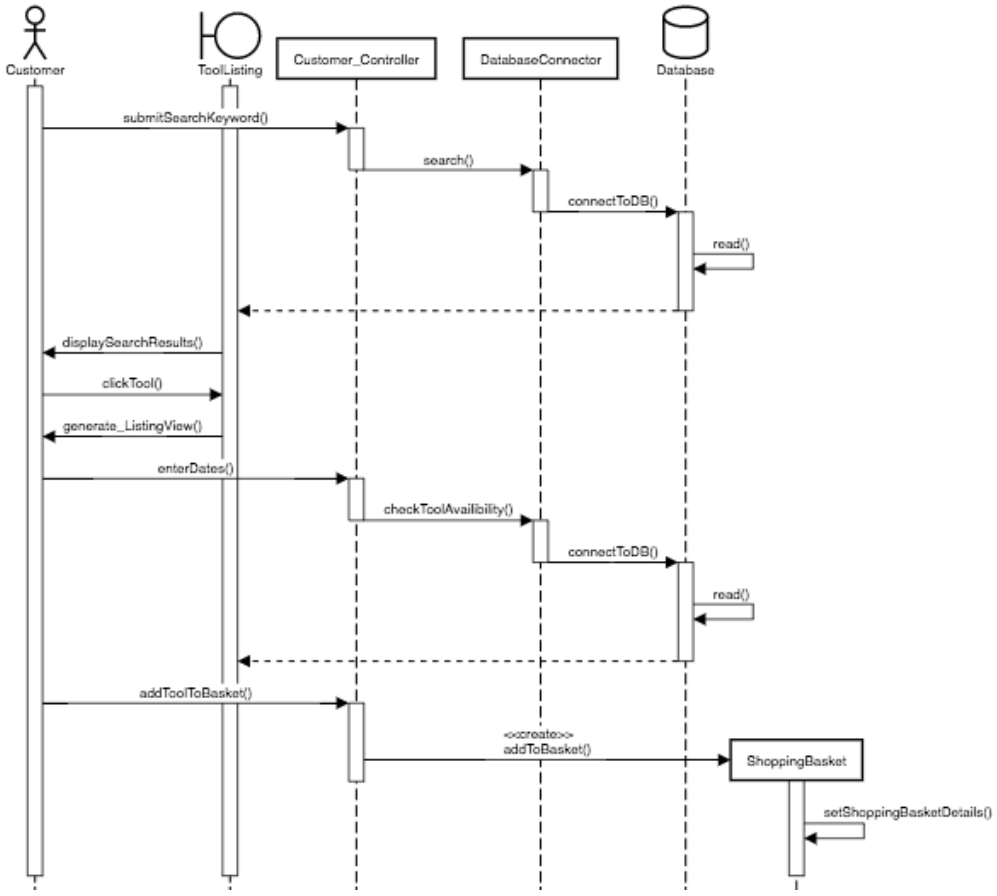


Figure 4.13: Sequence diagram for creating an order in the THS system. This is a version of UC7: Creates Order

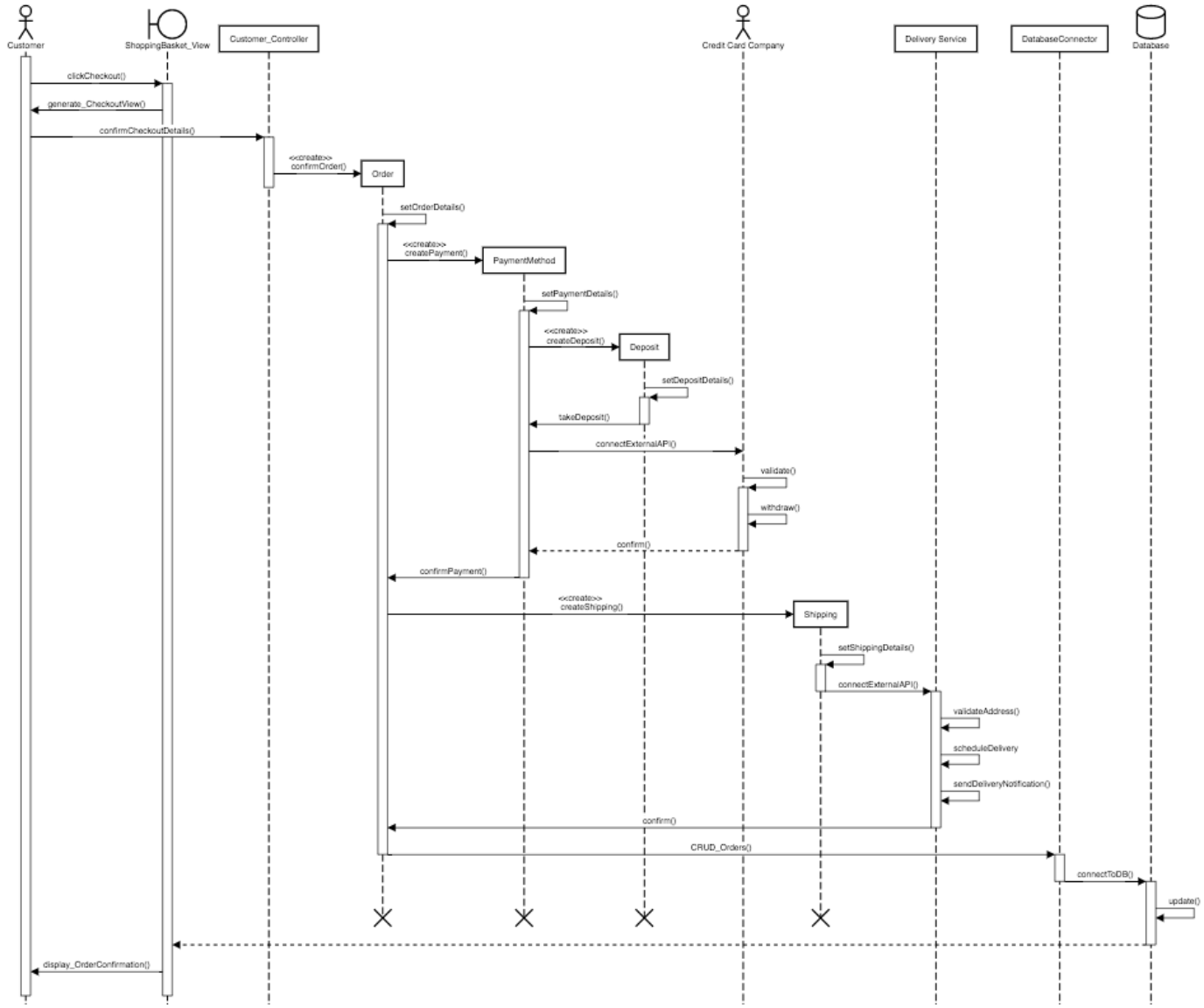


Figure 4.14: Sequence diagram for checking out. This is a version of UC8: Checks Out

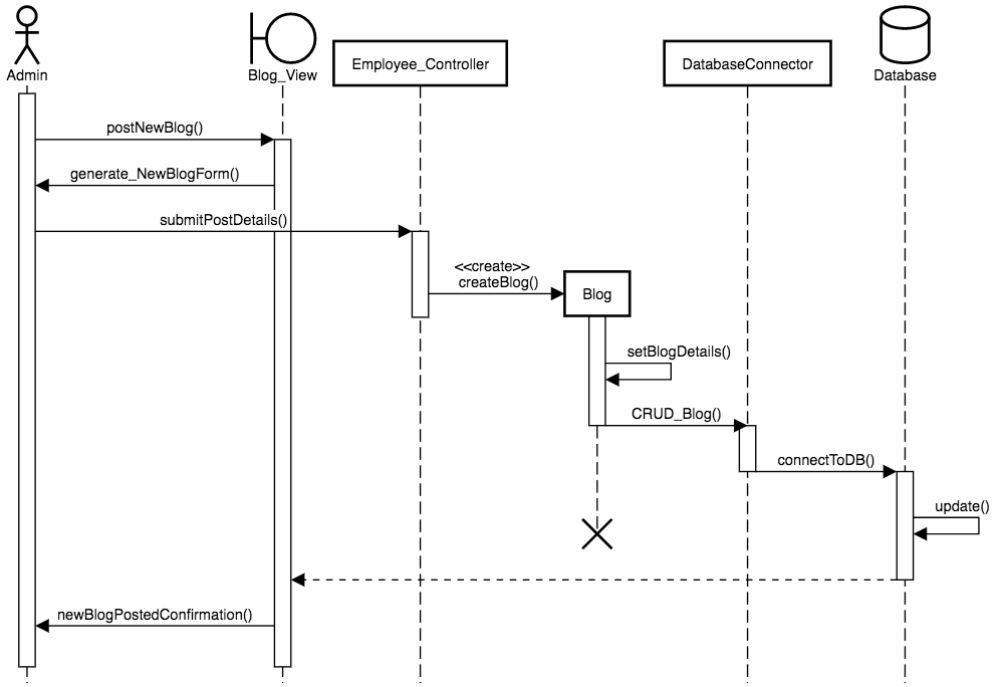


Figure 4.15: Sequence diagram for an admin creating a blog post. This is a version of UC23: Posts Advice Blog Video Tutorials

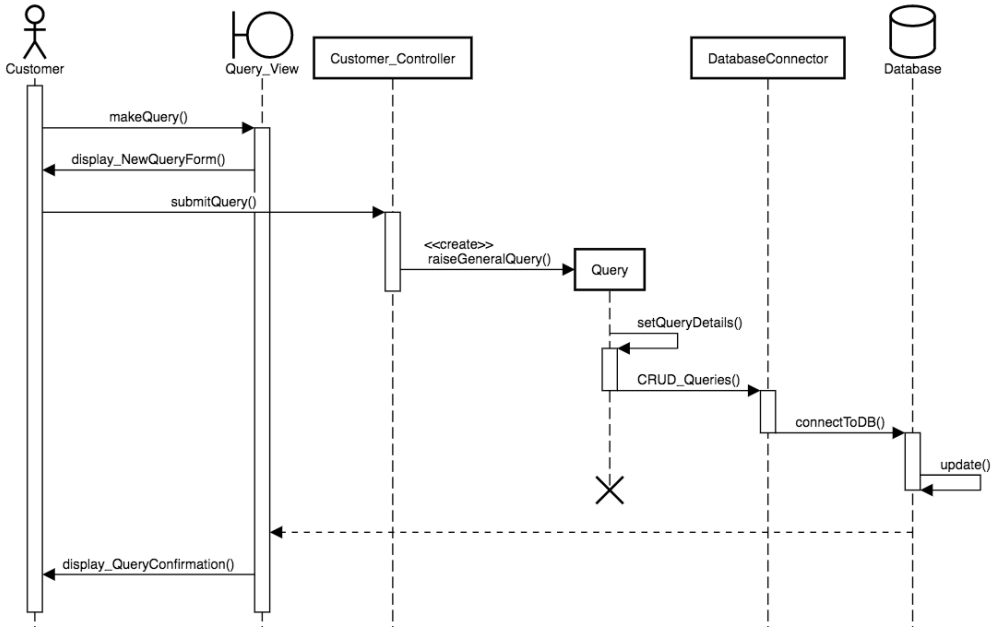


Figure 4.16: Sequence diagram for a customer entering a query into the THS system. This is a version of UC19: Raises Query

4.5 State Machine Diagram

In this section three state machine diagrams are displayed and explained.

Tool Rental Lifecycle

This diagram represents the lifecycle of a tool hire from the perspective of the tool itself. The tool can switch between four states which are: *Available, Booked, Serviced and Maintained*. The lifecycle begins at the available state. After the tool is booked by a client the tool changes its state to booked. Subsequently, once the tool is returned it is checked on functionality and observable flaws. Consequently, the tool is either serviced, if it needs repair or it is put back in the state of maintained. After this last step, the tool is ready for rental again and the state of the tool is changed to available. This closed the cycle and the loop continues.

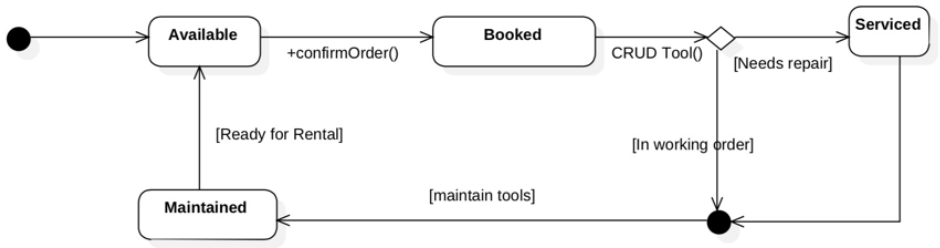


Figure 4.17: State machine diagram for a tool rental

Order

This diagram visualises the five different states from the shipping perspective. The five states are: *Pending, Packed, Dispatched, Delivered, Returned and Closed*.

Once the client completed his order, the order changes its state to pending, after the shipping is initialized the state of the order is changed to packed. Consequently, the ordered tool will be dispatched. If the order could not be delivered, it will be returned and dispatched again. Once the shipping is completed it will change its state to delivered. As soon as the order is cleared the order will change to the final state, which is the closed status.

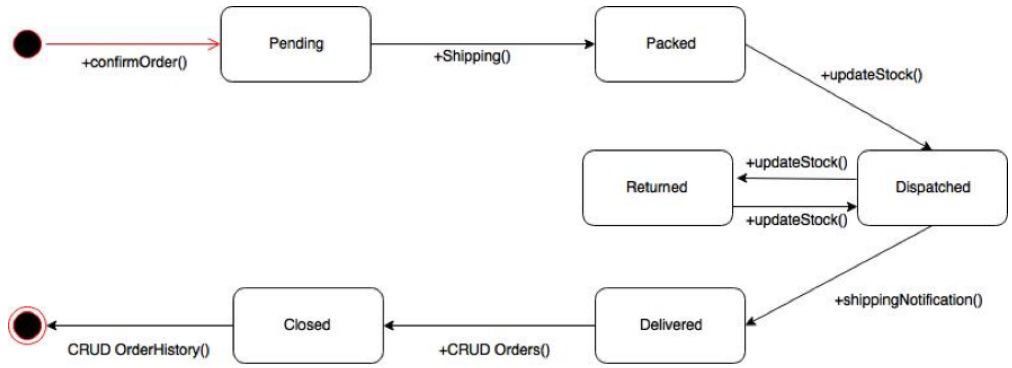


Figure 4.18: State machine diagram for an order

Shopping Basket

The following state machine diagrams displays the four stages of the shopping basket. The four states are: *Empty, Collecting, Checked Out and Ordered*.

The initial state of the shopping basket is empty, once tools are added it changes to the collecting state, if the added tool is deleted again we find ourselves again in the empty state.In the collecting state, arbitrarily many products can be added or deleted. As soon as 1 product is in the basket, the collecting state is initialized. Once the client confirms that the order the status is updated to checked out. If the summary is accepted the shopping basket will convert to the final state ordered. However, if the order is abandoned, the whole basket will be cleared and it goes back to the initial state, empty.

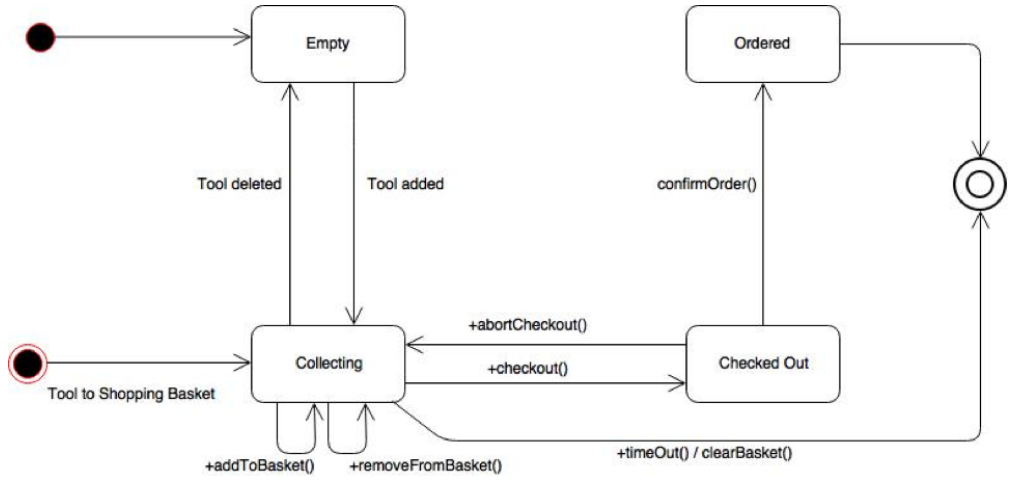


Figure 4.19: State machine diagram for a shopping basket

4.6 Component Diagram

Figure 4.19 shows the component diagram, which indicates a static representation of the projected system with regards to the logical elements. This diagram represents part of the architectural features of the system. It is more abstract, and it has less details than the previous class and object-oriented diagrams, therefore, it helps to familiarize the viewer with the system architecture. The DIY Tool Hire Service component diagram has four fundamental but interconnected groups.

Groups of the components in the DIY Hire Service system:

- **Components related to the User interaction.** It allows the users to interact with the system. We have a browser engine, that allow users to find the products; shopping basket, which manage client's tools basket and the User Session, such as the log-in process.
- **Components related to the Warehouse management.** It manages which tools are available and in which quantity.
- **Components related to the Payment System.** It interacts with the APIs of several third-party payment services, such as Visa, MasterCard, PayPal. It allows to complete the transaction
- **Components related to the Administration control.** It controls the Orders flow and consequentially it depends on the other components.

Each of the above stated groups includes of components of various types such as: executables, libraries, files, ports, connectors and interfaces. The figure below shows the components diagram which complies with UML 2.0 notation standard.

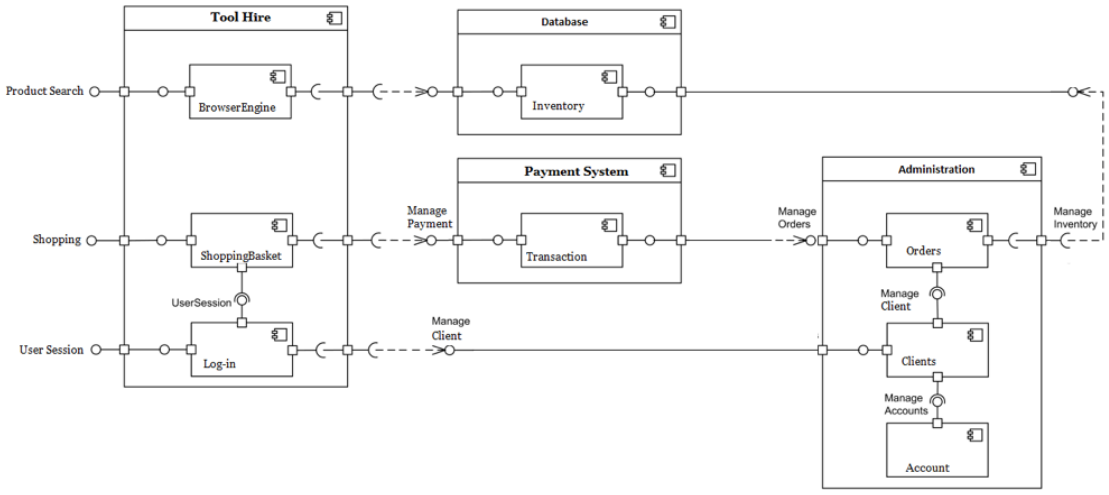


Figure 4.20: System components diagram

4.7 Three Tier Architecture

A three tier model partitions a system's component's into three layers of services. These are more a representation of the logical tiers in a system as opposed to a direct physical representation.

The outermost tier that is responsible for user interaction with the system is called the Presentation Tier. [13] For the Tool Hire Service, this means the website's interface with which the user interacts. Through the use of Dynamic HTML a THS customer or employee can interact with the services in the second tier in a simple and intuitive manner. The second tier is known as the Logic or Business Services layer. The processes contained in the second tier carry out the actual logic and functionality of the application and have access to the third tier. Since many clients can access this tier from different devices simultaneously, it is important that this tier manages its transactions in order to keep the data structures in the third tier consistent and reliable. [14] An example of how this may be used in the THS system would be if multiple customers were trying to book a tool at the same time. The logic tier must be able to "decide" which customer will get to rent the tool, update the database, and send a message to each customer indicated the result. Managing such conflicts in the second tier greatly reduces the burden on the third tier services. [13]

Finally, the third and innermost tier comprises of a system's data. The database management system (DBMS) and the data it manages can only be accessed via the second tier. [13] This is

where all data for the system is stored. In the case of the THS, some example of data stored here would be Order data, Customer data and Tool data. Access to this layer can be seen in the design class diagram as the DBConnector class.

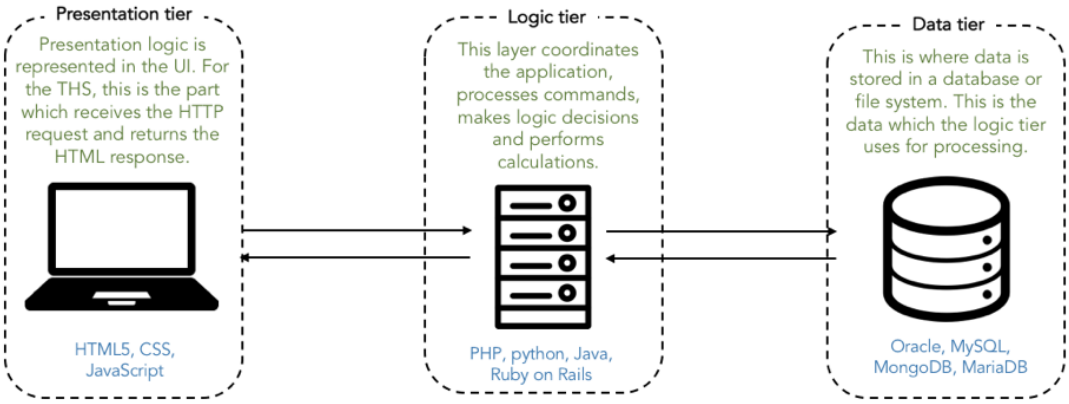


Figure 4.21: Three tiers for the THS system

Even though this architecture is more complex than a single tier, the benefits of using a three tier architecture outweigh the drawbacks. The three most important reasons a three tier architecture is suitable for the THS are:

- If the THS were to expand, the system is easily scalable since the data layer is independent from the other two.
- It is easier to maintain and modify the underlying code of the system if it is divided into separate layers, for example the UI can change without modifying the logic layer.
- Increased security with security defined for each layer. Since users can only access the second tier and not the third, if proper security measures are taken, a three tier architecture is more secure than a two tier architecture. However, having an additional tier means that during system development, security measure must be carefully discussed between developers working of different tiers in order to ensure that increased surface area doesn't lead to increased risk.

4.8 Deployment Diagram

The Deployment Diagram shown in Figure 4.21 shows the deployment of various software artefacts generated by the Tool Hire Service (THS) to deployment targets. Deployment targets are called nodes and can represent a device such as a computer or a software execution environ-

ment. The deployment diagram shows the full hardware architecture of the THS. Included is the customer's desktop, a webserver, an application server and finally a database. The customer's pc connects to the web server where the website is hosted via HTTP (Hypertext Transfer Protocol). The web server then connects to the application server via RMI (Remote Method Invocation) which handles the business logic for components such as orders, shipping and blog posts. Lastly the application server connects to the database via JDBC (Java database connectivity).

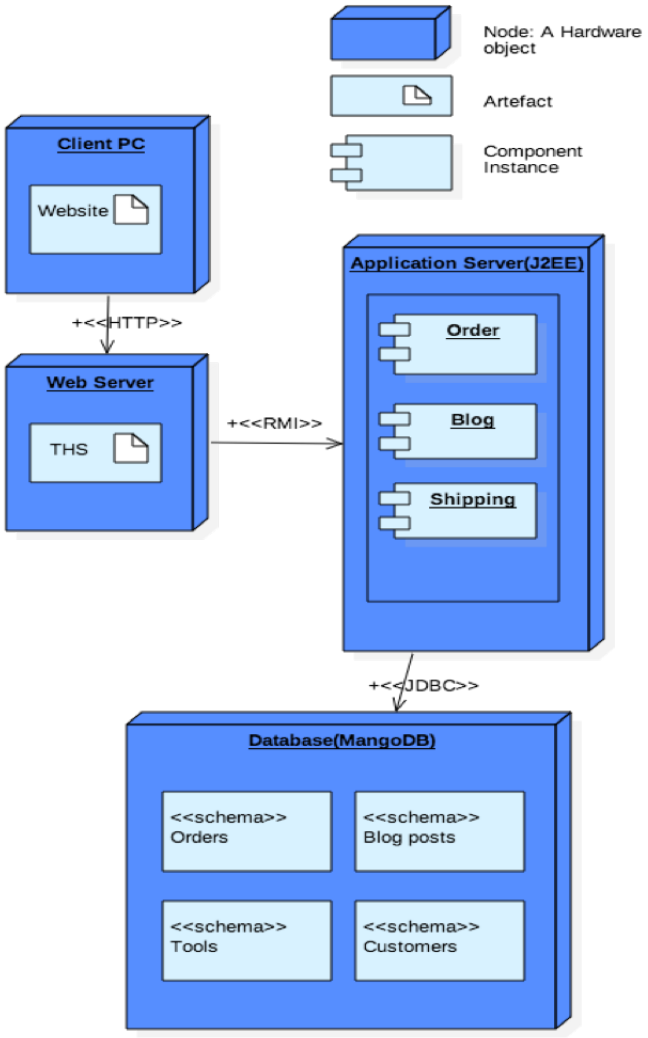


Figure 4.22: Deployment diagram

5 Application Mock-up

5.1 Home Page

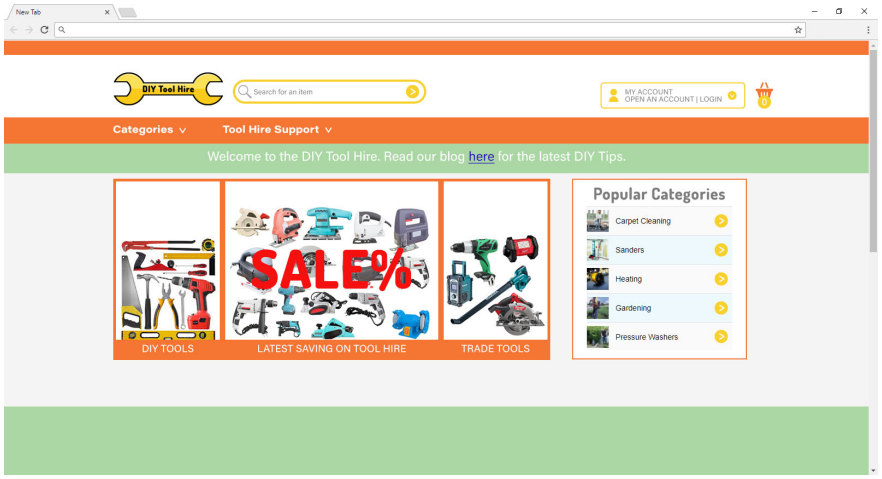


Figure 5.1: Home page for DIY THS

Presenting Do-It-Yourself Tool Hire! This is the home page of the site and the first one that a customer encounters when they visit.

5.2 Create Account

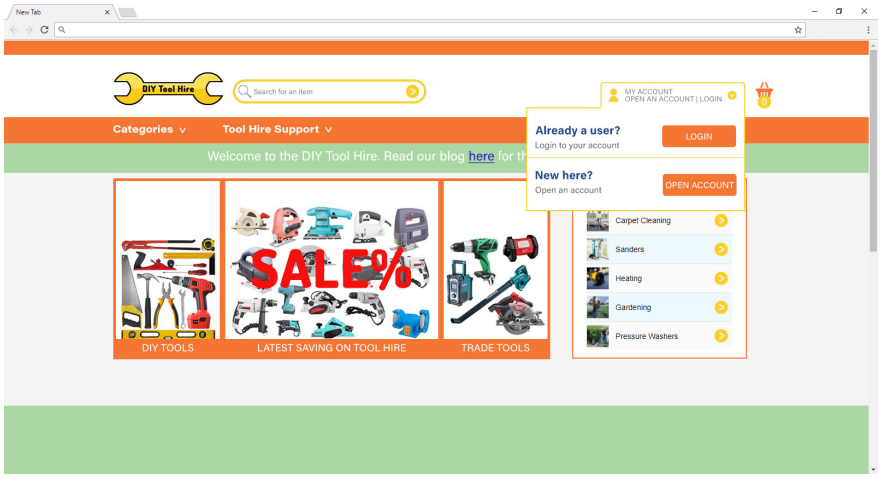


Figure 5.2: Create new account or log in page for DIY THS

Before using the service, you can create a new account or log in if you are an exsisting user.

5.3 New Customer Registration Page

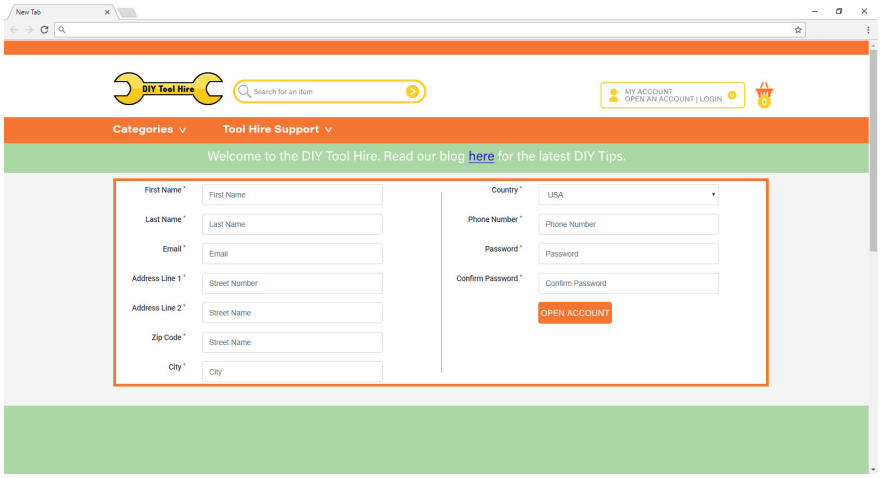


Figure 5.3: Customer registration page for DIY THS

You can create a new account by filling in your account details, all fields are required and once filled out you can click open account to start hiring some tools.

5.4 Log-In Page

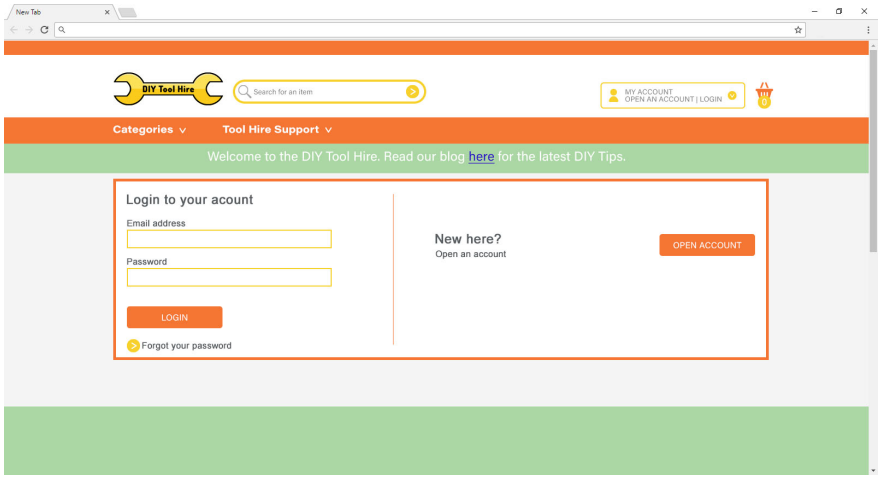


Figure 5.4: Log-in page for DIY THS

To start using your new account enter the email address you used to create your account and your password.

5.5 Password Recovery

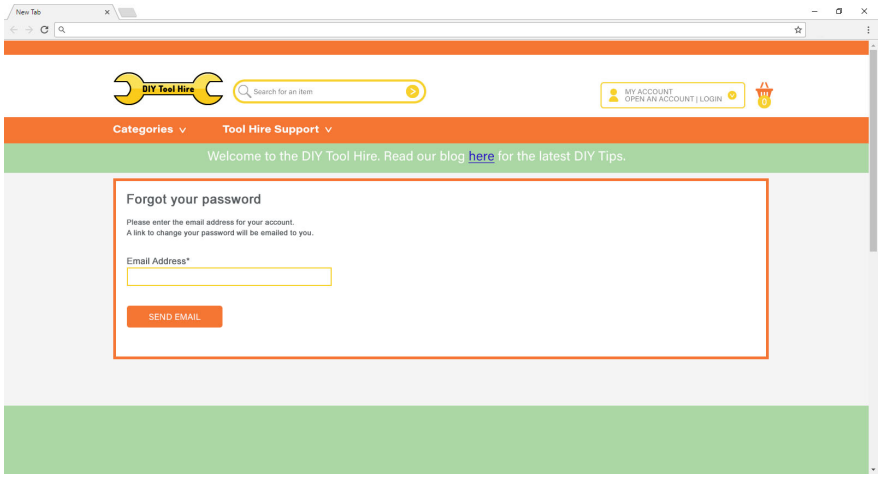


Figure 5.5: Password recovery page for DIY THS

In the event that you forget your password you can click on the forgotten password link located on the home page. Type in the email address you used to register your account and click the send email button. You will be sent an email containing your password.

5.6 Manage Account

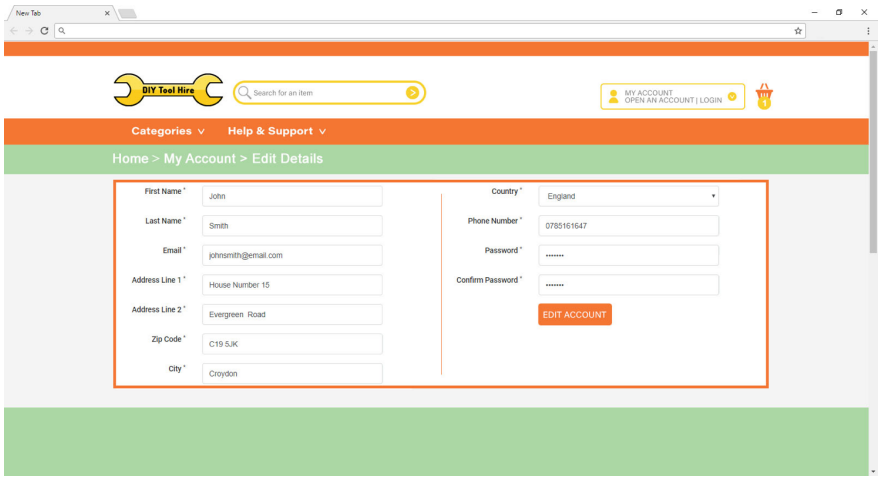


Figure 5.6: Manage Account page for DIY THS

Once logged in you also have the option of changing your account details by clicking on the My Account button. The form will be filled with your old account information and once you change a field you can click edit account to save your new account details.

5.7 Check Categories

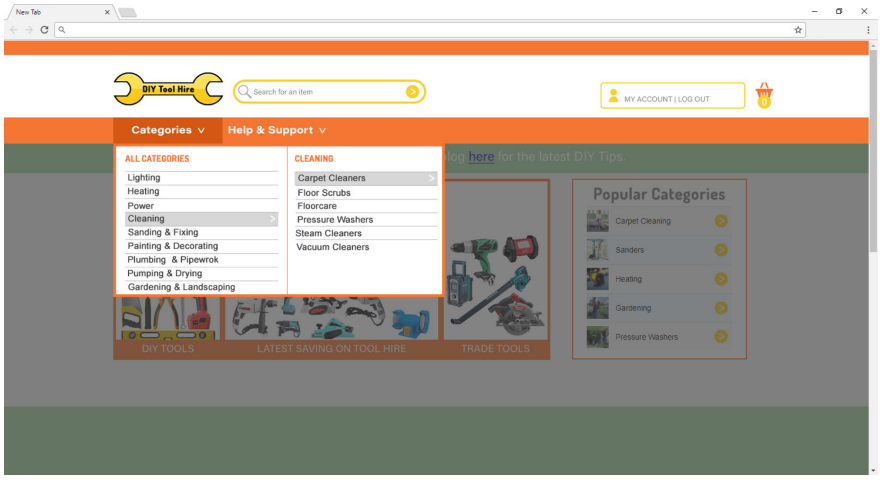


Figure 5.7: Check Categories for DIY THS

Under the checking categories we offer a different selection of tools for different tasks. As can be seen in the diagram below if you require a tool to aid in cleaning, the website offers many different cleaning tools which can be rented from floor scrubs to vacuum cleaners.

Once a tool has been selected you can view a short description of the item's functionalities as well as the pricing plan for the item. You have the option of renting the item for one day, two days, a weekend or a full week. After deciding what option is best you can select your required start and end date.

5.8 Checks for Delivery Qualification

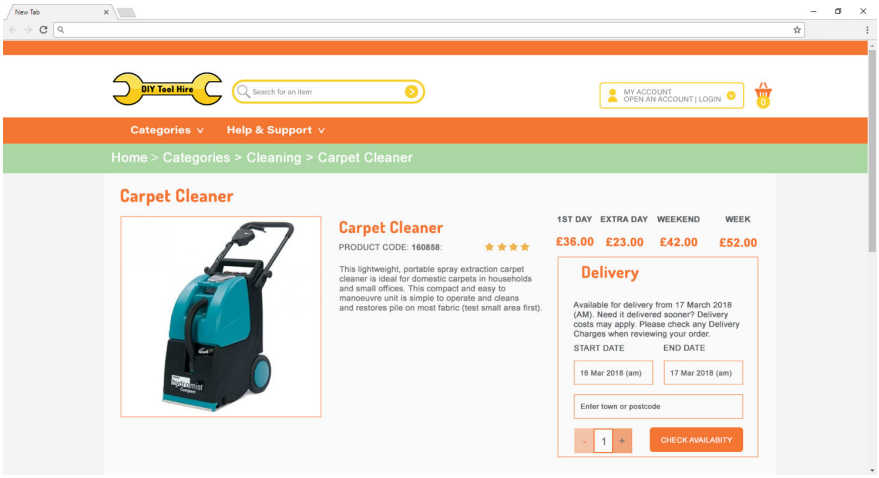


Figure 5.8: Checks for Delivery Qualification page for DIY THS

After selecting how much of the item you wish to rent you can click on check availability to see if it's possible to deliver the item(s) to your postcode. To add the item to your checkout basket, click the orange basket icon next to 'My account'.

5.9 Shopping Basket/Creates Order

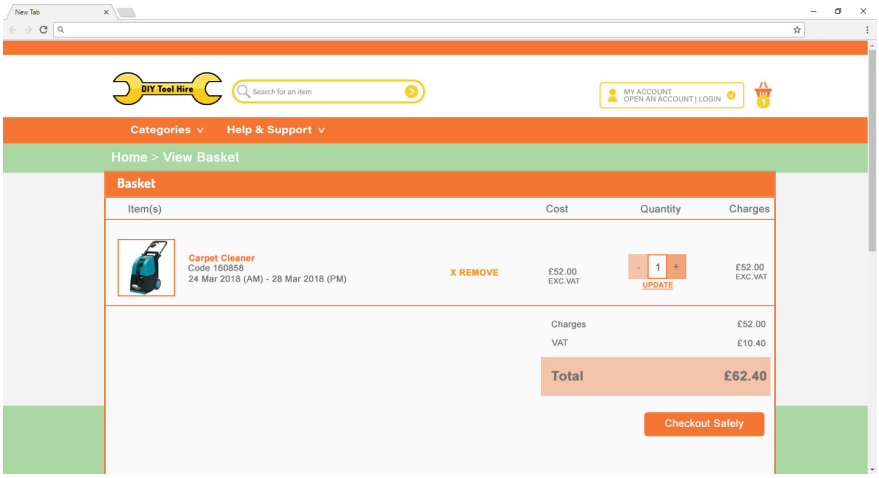


Figure 5.9: Shopping Basket/Creates Order page for DIY THS

After adding the item, you wish to rent to your basket you can review the final information about your order such as the required rental time and the final price including VAT. Click on 'Checkout Safely' to be taken to the Checkout and Payment page.

5.10 Checks Out

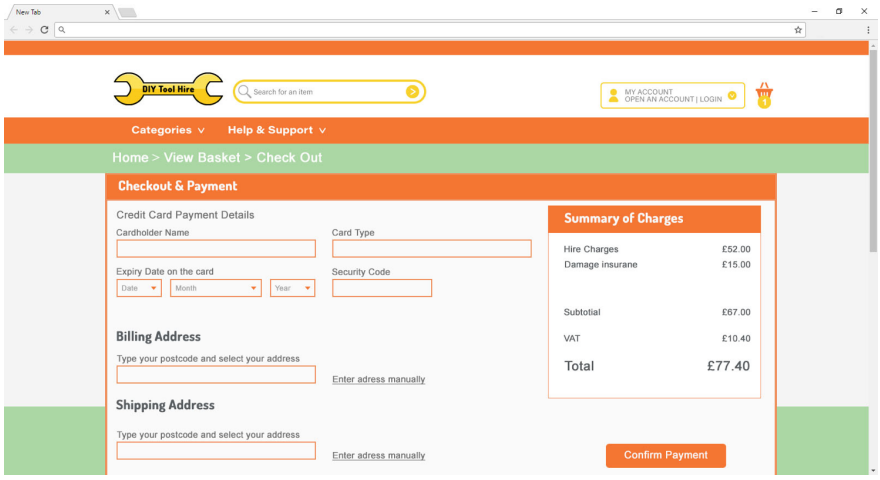


Figure 5.10: Check out and payment details page for DIY THS

On the payment page you will be able to fill out your debit/credit card details, billing address and shipping address.

5.11 Confirms Order & Updates Tool Availability & Track Order

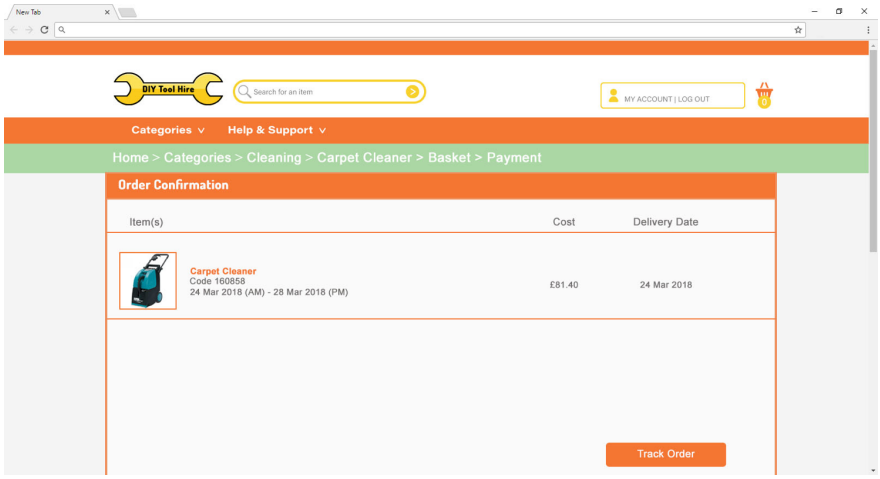


Figure 5.11: Confirm order page for DIY THS

After confirming your card details and it's successful, you will be taken to the order confirmation page where you can track your order and view the expected delivery date.

5.12 Raises Query

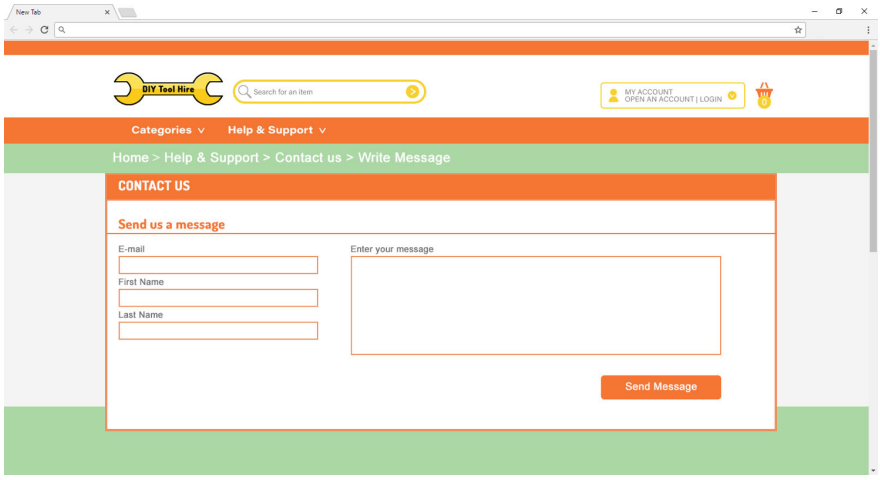


Figure 5.12: Raises Query page for DIY THS

In the event that there is any additional information required. You can click on the help & support tab and fill out a form sending us a query about anything from delivery items to damaged equipment.

5.13 Reads Blog & Subscribes To Blog

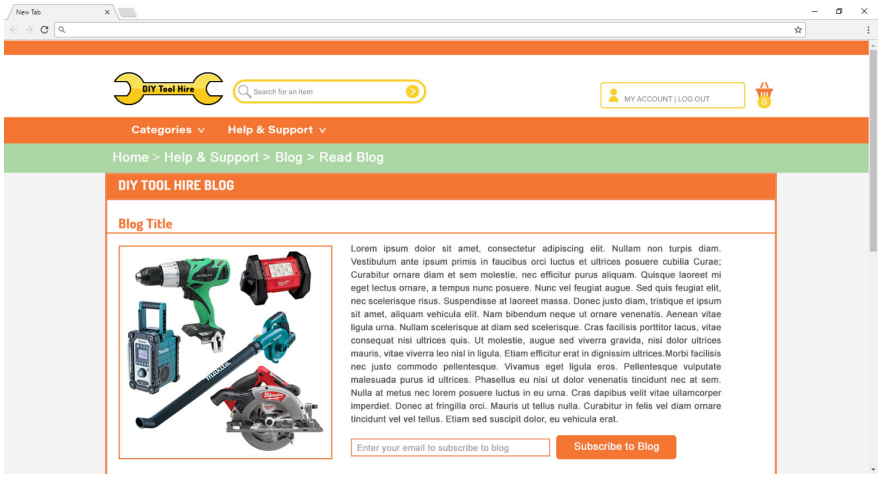


Figure 5.13: Blog page for DIY THS

You can find the blog under the help and support tab, where we release exciting new blogs about the latest tools, and tips and tricks on how to get the most out of your rentals!

5.14 Log Out

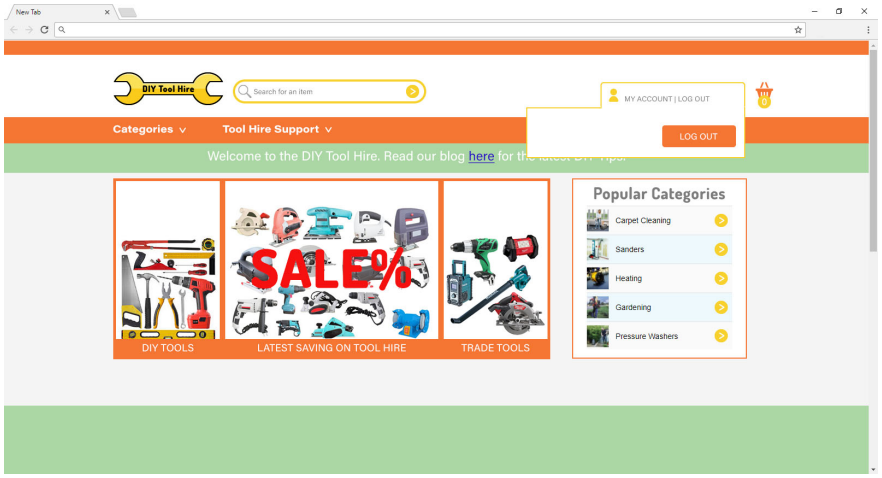


Figure 5.14: Log out page for DIY THS

After a customer has completed their session, they can log out of their account.

5.15 Employee Login

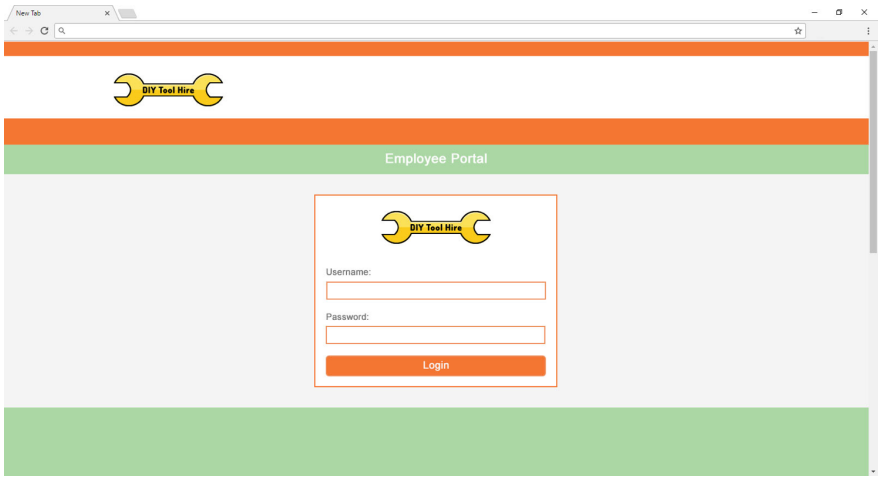


Figure 5.15: Employee Login page for DIY THS

All employees can login with their given account through the site's employee portal.

5.16 Employee View & Manage Tools

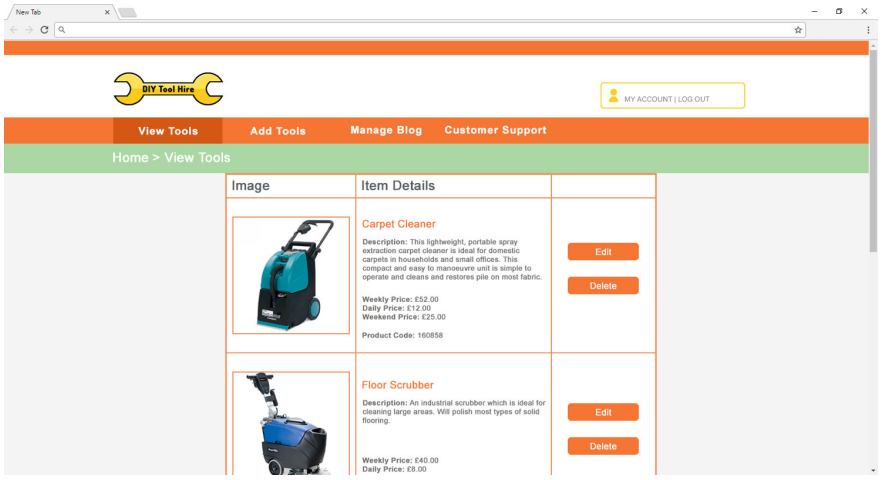


Figure 5.16: Employee View for DIY THS

An employee has the ability to view all the tools currently on the site, as well as the ability to edit the description and to delete items.

5.17 Add Tools

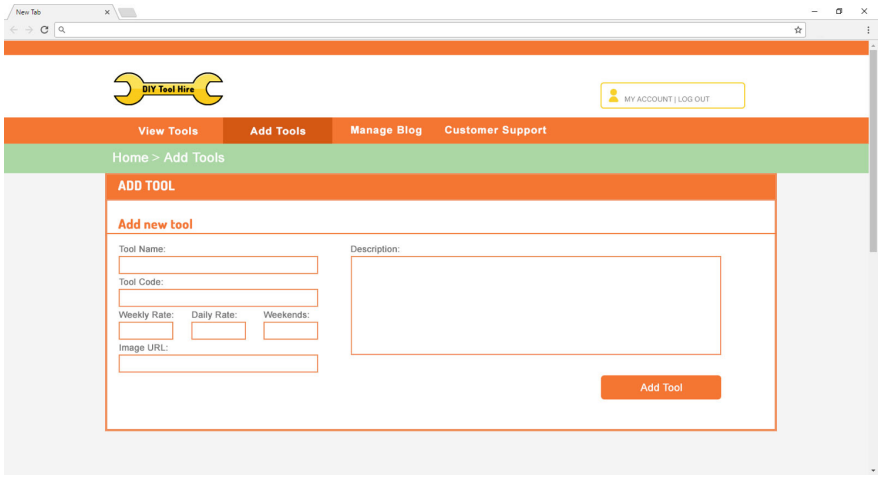


Figure 5.17: Add Tools page for DIY THS

To add a new item, the employee must click on the add tools tab and fill out all the required information for the item before adding it.

5.18 Employee Add New Blog Entry

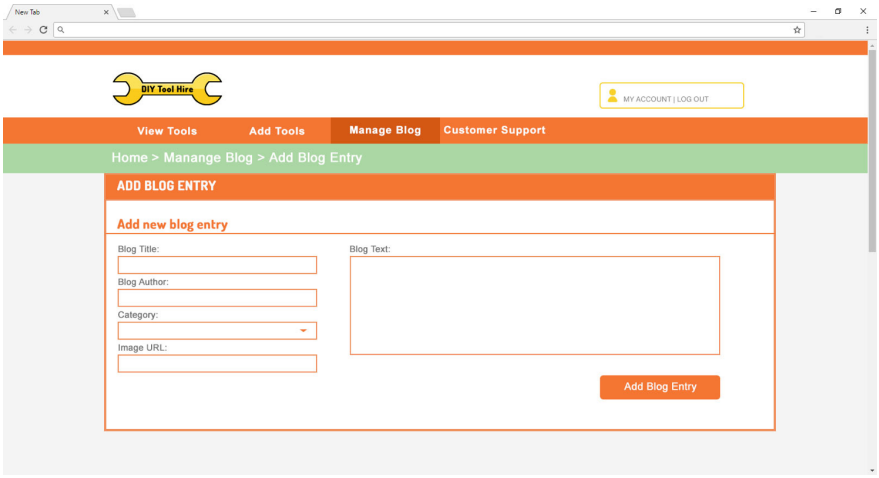


Figure 5.18: Add New Blog Entry page for DIY THS

Similarly, an employee can post a new blog by navigating to the add blog tab and filling out the form.

5.19 Employee Reply To Query

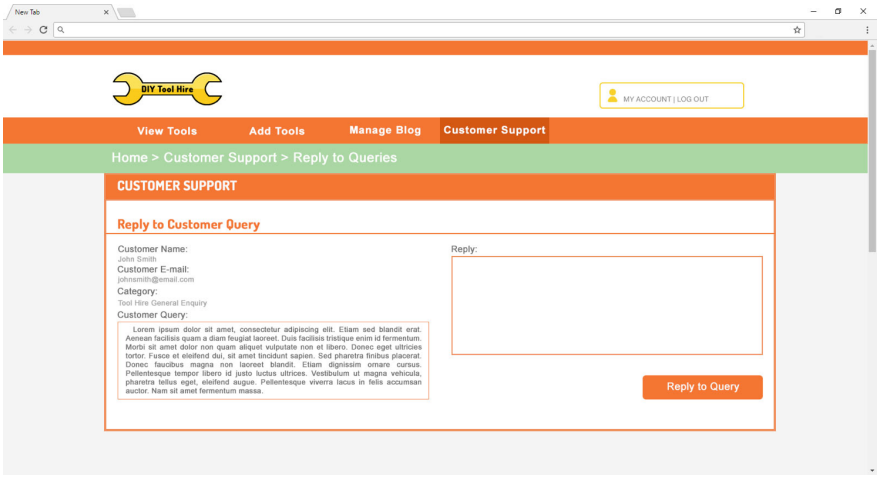


Figure 5.19: Reply To Query page for DIY THS

The site also allows employees to reply to queries. The employee will see the name of the customer and their query and will have the ability to reply.

6 Conclusion

In this chapter, an evaluation of the strategy utilized to design the THS will be provided, including a reflection on successes and challenges and some considerations for the future.

The team met on a weekly basis and, to increase the productivity, the team was divided into sub-groups to carry out varying work concurrently and efficiently. All documents were shared via OneDrive enabling everyone access to all stages of diagram iteration. During the weekly group discussion, the sub-group's results were reviewed by the whole team with an open discussion in order to achieve high quality and consistency between all members. The weekly progress was monitored by the team's advisor who gave feedback and advice on the tasks for the following week.

The development approach was based on strong requirements and use cases. The declaration of actors and their functionalities helped to create the classes and methods required for the Object-Oriented Analysis stage. As the team wanted to ensure that the flow of use cases was represented properly, many iterations of object oriented diagrams were made. (See Appendix). The current model is quite an advanced iteration, that would enable a development team to start implementation. The modular (object-oriented) nature of the design means further iteration of the classes and front-end development is possible without requiring major changes. This in-built design focused on short, simple iteration phases is in line with the tenants of the Unified Process.

Sustained communication throughout development and regular meetings comes across in the fluidity of the design, report and presentation. Both internal and external deadlines were met. These regular reviews and short sprints of individual work allowed us to overcome the challenge of synchronizing different backgrounds, knowledge bases and writing styles.

Successes
Management of a team of 8 with different time schedule and workload.
Pragmatic design of a use case driven model for the DIY Tool Service using UML.
Good use of the unified process and SCRUM techniques.
Comprehensive set of use requirements, developed early provided a crucial base for further development.
Sustained communication throughout development and regular meetings.
Challenges
Deciding on a standard UML style based on the choice of tools/resources used.
Maintaining consistency throughout report writing.
Ensuring class diagrams covered every possible use case and followed an object-oriented model that would allow correct implementation.
Lack of experience with deployment of large systems and the hardware involved made component and deployment diagrams more challenging to design.

Figure 6.1: Successes and challenges for Group A.

Future Considerations

A fully functional website design and implementation specification has been produced for the THS but there are some considerations which will need to be researched as the system is built and scaled:

Database Design: Much of the data the system requires has been modelled in class variables. However, an entity-relationship diagram and data schema will need to be produced ahead of database development. This is crucial for the THS which is dependent on a database to function.

Concurrency: One of the non-functional requirements was that the THS would allow 5000 concurrent users at any time. Therefore, consideration needs to be given to the implementation of this – for example building a multi-threaded application.

Front-end development: A template should be built from the wire frame early on in the development of the system – to enable linking with the boundary classes, and to identify, early on, any iteration of the boundary classes/ interfaces needed.

Security: This is crucial in a system with many users that stores sensitive information such as payment details and personal information. Consideration needs to be given to password and message encryption, firewalls and database security before the system can be used.

7 References

1. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999.
2. J. Arlow and I. Neustadt, UML and the Unified Process. London: Pearson Education, 2002.
3. K. Schwaber and J. Sutherland, "The Scrum Guide", scrumguides.org, 2017.
4. "What's the Difference? Agile vs Scrum vs Waterfall vs Kanban", www.smartsheet.com, 2018.
5. F. Brooks, The Mythical Man Month, Addison-Wesley, 1975.
6. A. Kline, Agile Development in the Real World, Apress, 2015.
7. S. Ambler, Agile Model-Driven Development with UML 2.0, Cambridge University Press, 2004.
8. A. Cockburn, Writing Effective Use Cases, the Crystal Collection for Software Professionals, Addison-Wesley, 2000.
9. R. Miles K.Hamilton, Learning UML 2.0, O'Reilly, 2006.
10. A. Kalmback (2015). 'Entity-Boundary-Interactor; A modern application architecture'. readthedocs.io [Online]. Available: http://ebi.readthedocs.io/en/latest/
11. Whitten, L. Bentley and K. Dittman, Systems analysis and design methods. Boston, Mass.: Irwin/McGraw-Hill, 1998.
12. E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design patterns. Boston, Mass.: Addison-Wesley, 2016.
13. "Three-tier architectures", ibm.com, 2018.
14. "Using a Three-Tier Architecture Model", msdn.microsoft.com, 2018.

8 Appendices

8.1 Meeting Minutes

Meeting 1: 23rd Jan 2018

Work Flow:

List of tasks for next 3 weeks compiled:

Week 1: Requirements gathering

- Industry Research
- Everybody to research how to use Latex
- Shared resources to be set up: Github, Latex, One Drive

Week2: Refinement of requirements (including categorisation)

- MoSCoW prioritisation and effort estimation for requirements
- Begin thinking about use cases

Week 3: Continue developing use cases

- Need to consider who stakeholders for the app are (company, renters of tools).
- The application should be designed as a web app – for use in a browser and mobile.
- Phoebe to share UML 2.0 book to use for UML syntax / diagrams.
- Daiana to act as SCRUM master for project.

Plan for next meeting

To meet every Tuesday 11am - 1pm, possibly in room in Foster Court.

Next Meeting to bring post-its (!) to help organise requirements visually.

Meeting 2: 30th Jan 2018

- This week focus on use cases and organise requirements.
- Brief discussions of Object oriented diagrams, sequence diagrams, activity diagrams
- Came up with non-functional requirements for the project
- Came up with non-functional requirements for the project

First Meeting with Rae

Review of vision and scope

Suggested we should convert requirements to excel and number them

She suggested that performance tests run on key functions can help determine non-functional requirements in an actual implementation

Meeting 3: 6th Feb 2018

Discussed: Actors:

User – anyone that interacts with the system, Customer, Employee, Admin, Customer Support, Credit Card Company, Delivery service, Notification Service, Database

To Do this Week:

- Finish writing up Use Cases and make sure they match names on use case diagram
- Finish analysis class diagram

Meeting 4: 13th Feb 2018

- Refining Use-cases
- Discussion about state machine diagrams- Rae gave some examples and the group plans to review these
- Class diagrams- inclusion of getters and setters
- Sequence diagrams- making sure they are consistent with class diagrams

Meeting 5: 6th March 2018

- Discussion of analysis and design class diagrams final changes- adding JSP/JavaClass notation to design and Boundary, Controller, Entity notation to the analysis class diagrams
- Discussion of final modification for sequence diagrams
- Clarification about deployment diagrams- modelling how a three tier architecture system would be deployed

Meeting 6: 15th March 2018

- Final diagrams discussed
- Discussion of report section allocation

- Discussion of presentation assembly

Meeting 7: 20th March 2018

- Final report looked over by Rae: edit the alternative flow of use cases
- Making presentation for video and shooting video

8.2 Design Class Diagram Iterations

8.2.1 Iteration One

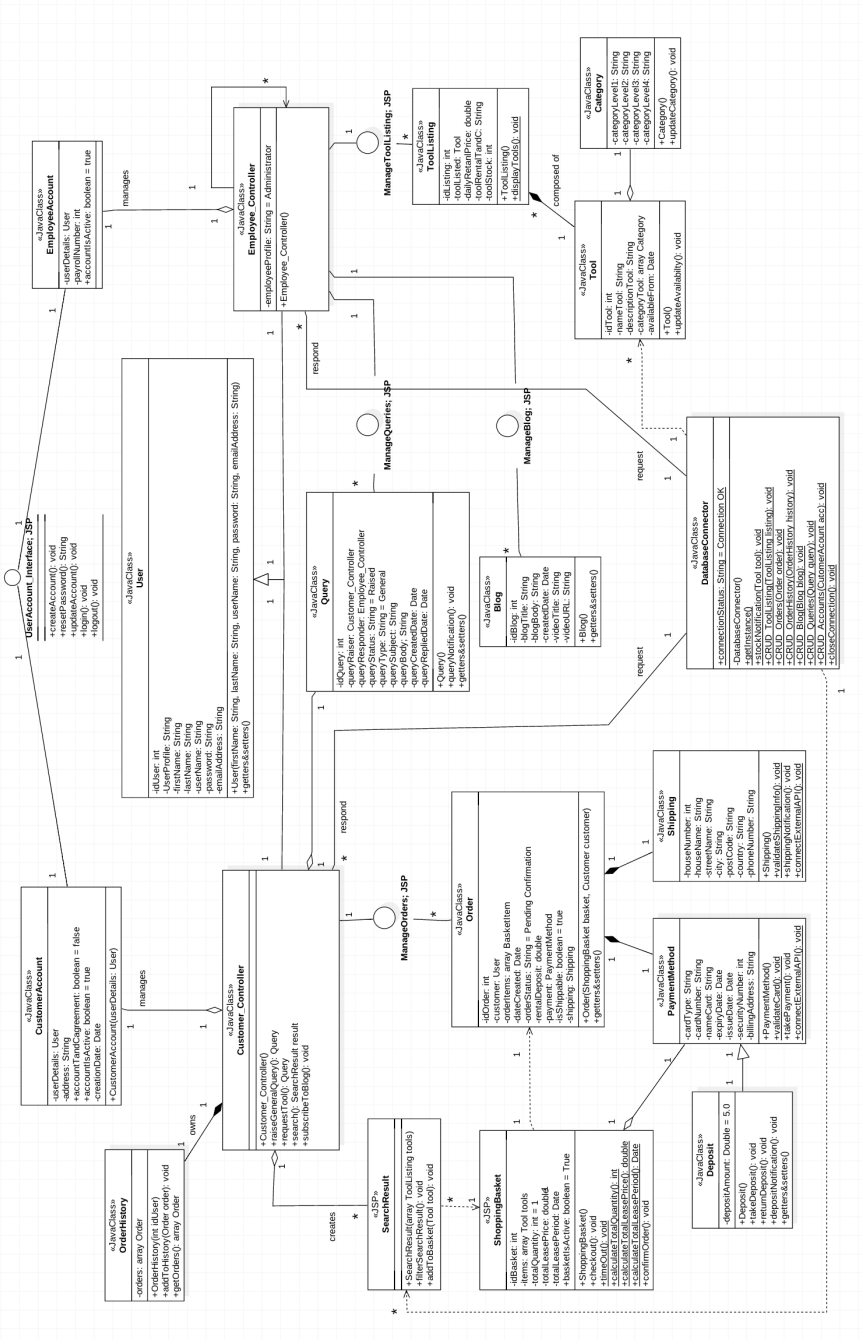


Figure 8.1: First iteration of design class diagram

8.2.2 Iteration Two

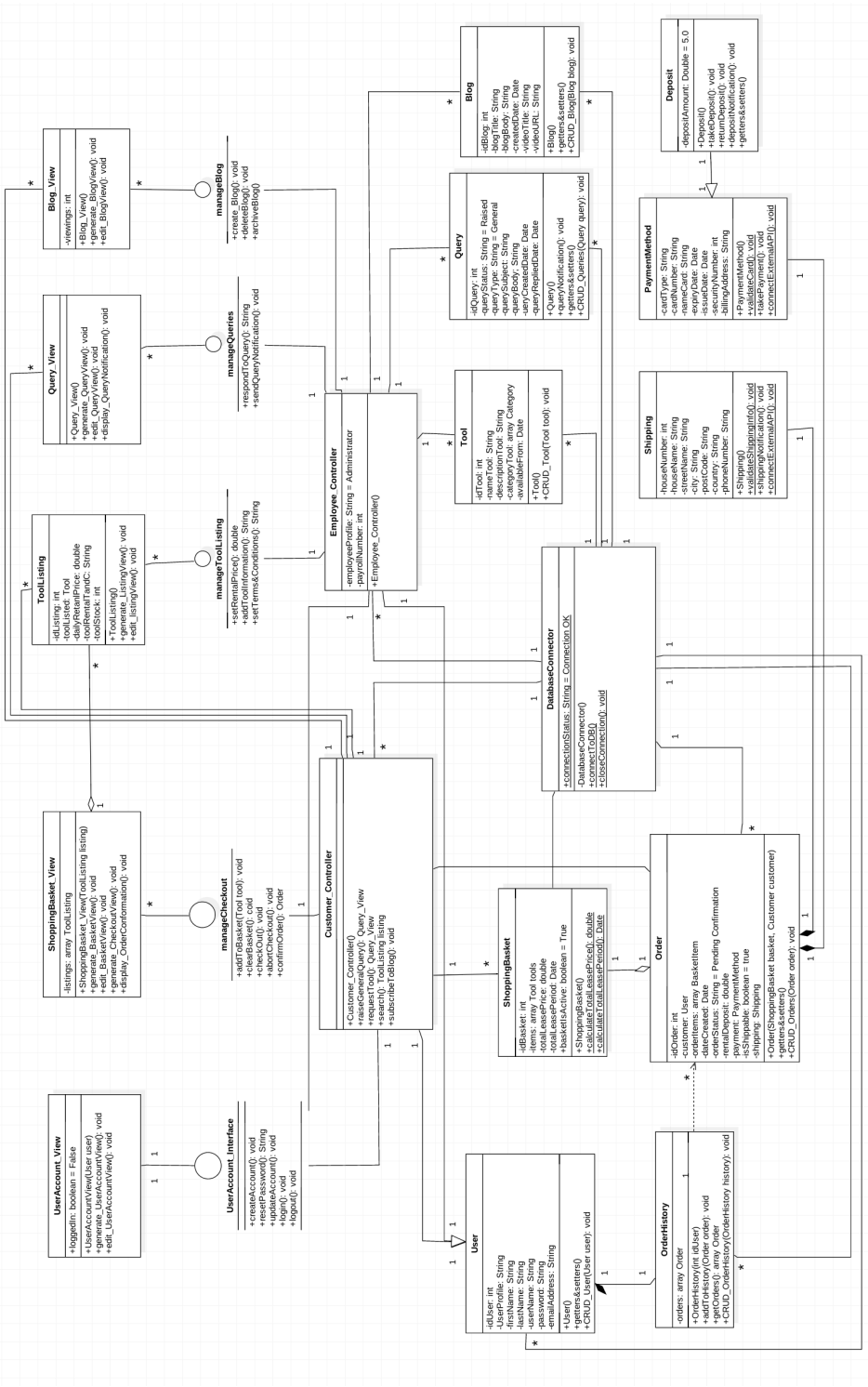


Figure 8.2: Second iteration of design class diagram

8.3 Sequence Diagram Iterations

8.3.1 Iteration One: Creates Order

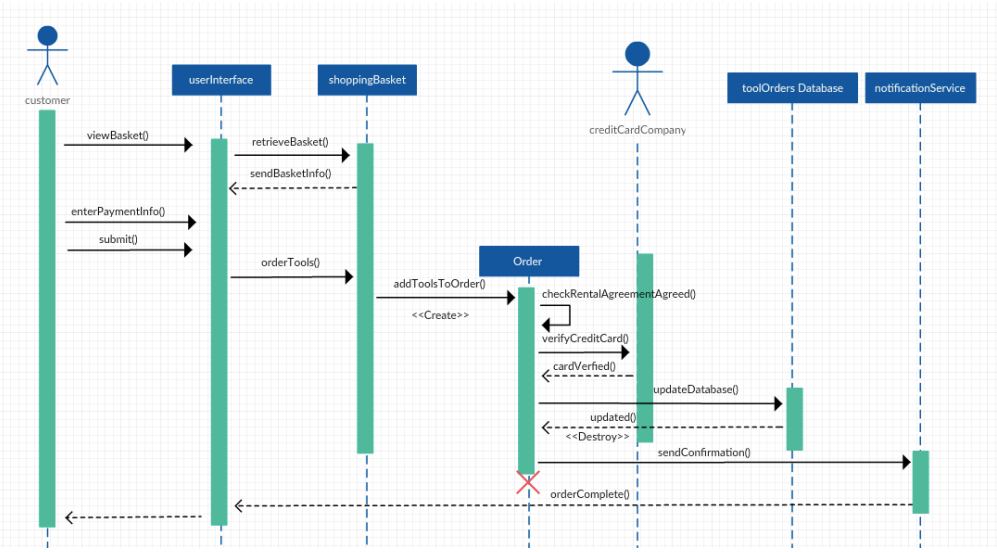


Figure 8.3: First iteration of "Creates Order" sequence diagram

8.3.2 Iteration Two: Creates Order

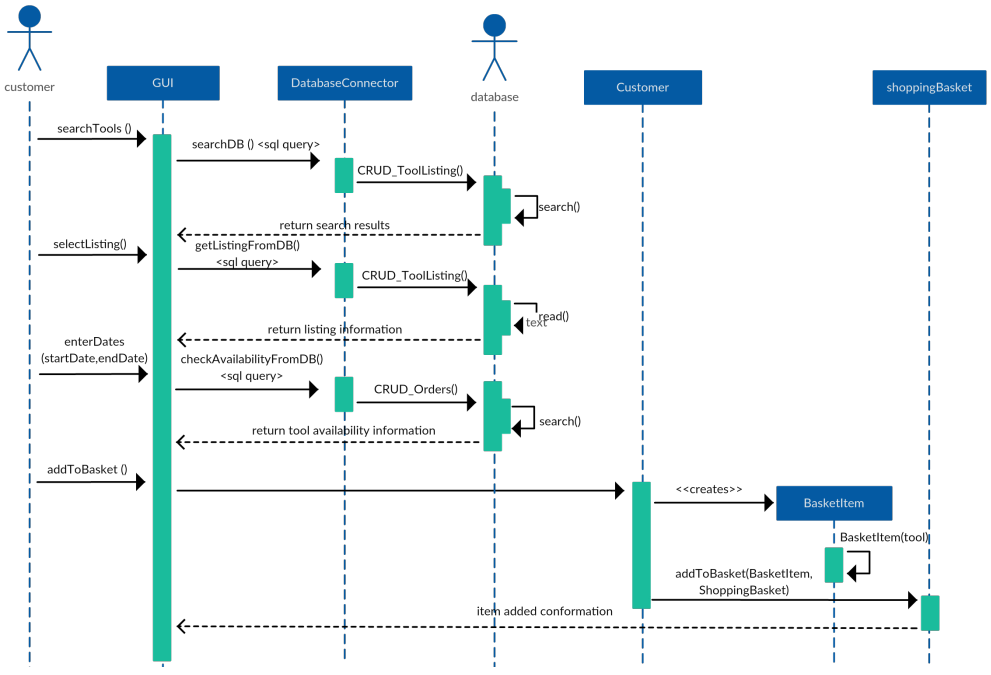


Figure 8.4: Second iteration of "Creates Order" sequence diagram

8.3.3 Iteration One: Checks Out

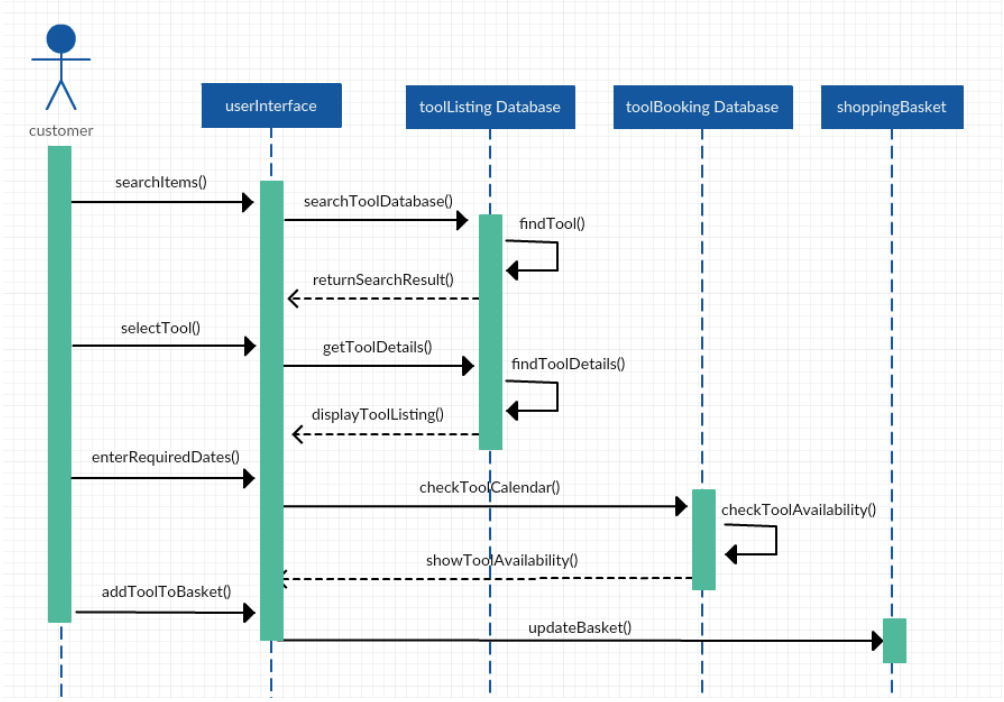


Figure 8.5: First iteration of "Checks Out" sequence diagram

8.3.4 Iteration Two: Checks Out

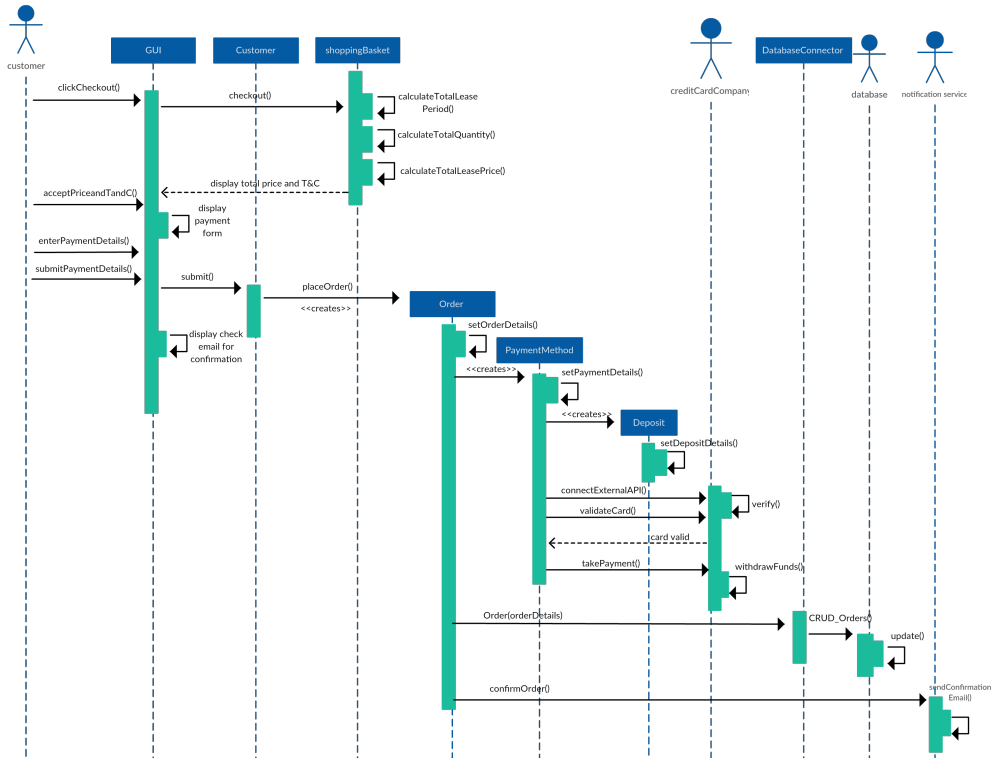


Figure 8.6: Second iteration of "Checks Out" sequence diagram

9 Team Member Contribution

Team Member Name:	Main Contributions
Everyone	Initial gathering of requirements and use cases.
Stefan Poechhacker	State Machine Diagrams, Component diagram, Conclusion, Vision & Scope (with Antonio)
Caroline Smith	Domain Model, Iteration of Analysis Class diagram, Design Class diagram, OO analysis & design report section
Phoebe Staab	Sequence Diagrams, Activity Diagrams, Use case diagram, Three-tier architecture, Report structuring (LaTeX), Executive summary
Cameron Mory	Helped design use case diagram, Development/Iterations of Use Cases, Iteration of Use Case/Requirement Matrix, Introduction, Requirements, and Use Case sections of the report
Antonio Manganelli	State Machine Diagrams, Component diagram, Conclusion, Vision & Scope (with Stefan)
Daiana Bassi	Use case diagram, Created all mock-ups in the report, Class diagrams, Video editing
Ahmed Afify	Use Case diagram, Class Diagrams
Gideon Acquaah-Harrison	Domain model diagram, Deployment diagram, Descriptions for mock-ups in the report

Table 9.1: Overview of Group A team member contributions