**Goals of The Project**

The goal of this project is to have a fully operational rental server website. We plan to achieve many goals in this project.

One such goal is to have fully operational servers that have the ability to accomplish multiple tasks for many different types of customers. In order to be able to achieve this goal we decided to implement the capability for the customers to choose their specifications for their specific server. Choices such as how much RAM, what type of CPU, how much hard drive space or what type of graphics card. This ability of choice allows the customer to make their server's purpose benefit their requirements in many tasks such as data storage, cloud computing, scalability etc.

Another goal of the project is to achieve all the systems that we have decided to implement into the website. One such system will check if the user haves made the necessary payments and if they have not the server will automatically wipe that sever after a certain period of time. This system allows for the website to run with minimum admins checks.

Additionally, we want to implement the process of an automatic email system. For this email system we decided to achieve a system in which when a user sign up, they are sent an email welcoming them as a new user. Another use for the email system will be for sending warnings about unpaid payments. Lastly the email system is used to send a copy of the order they just made.

Another goal that we decided on was to encrypt payments. We decided to add to extra functionality as we saw it would improve security. Additionally, we saw it as a way to show that customers should trust with their business and secure data.

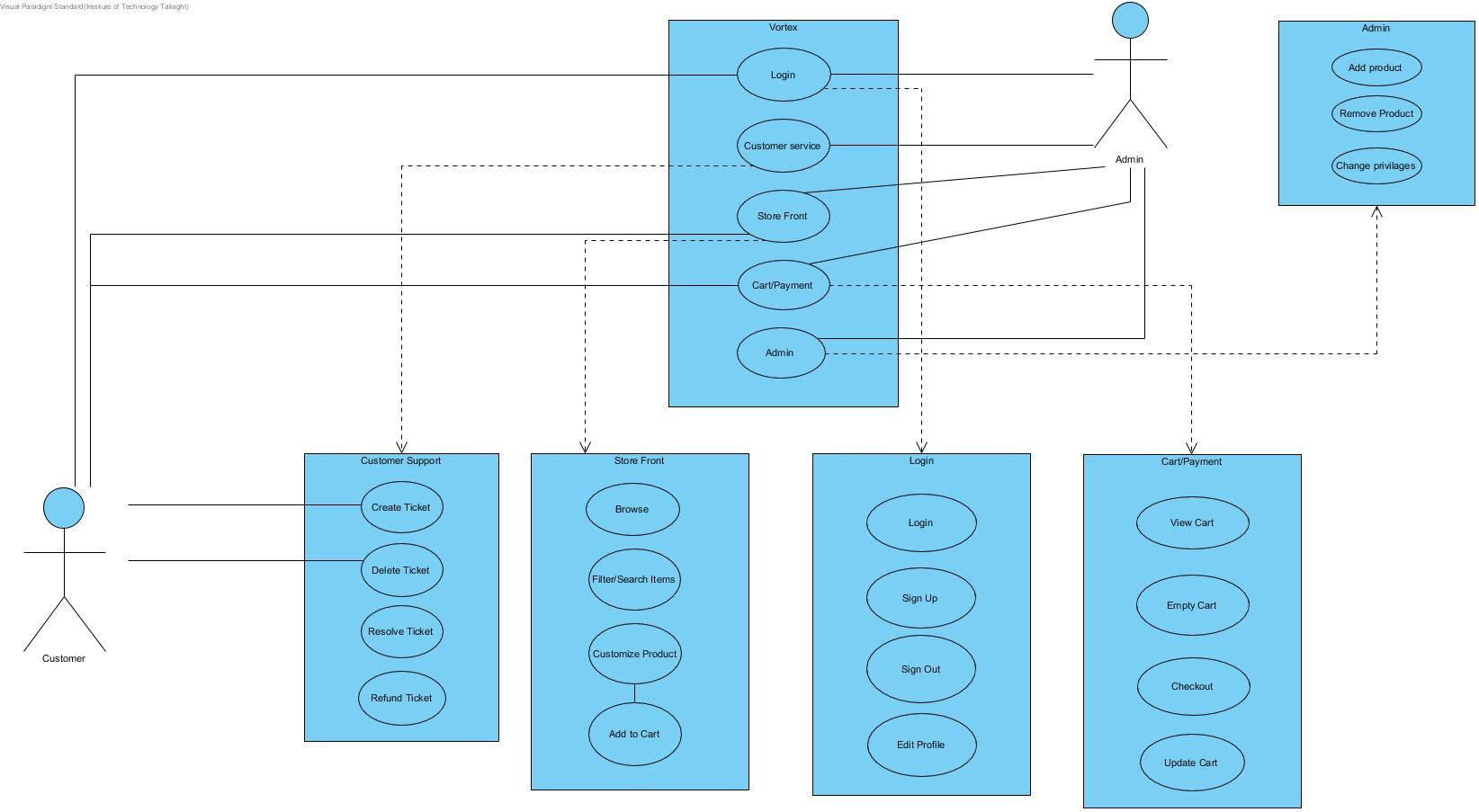
Another goal of the project is to make sure that the users achieve an easy to use website that will provide many options of usability. The main users will have the option of selecting and crafting their own unique kind of server. This is also coupled with the fact that the user will be allowed to have the server for as long as they can reach the payments or else a routine operation will wipe their server. Another user of the site will be the admins who will be adding any additionally product or taking away a certain product from the website.

Another system of the plan is to add a support ticket system. The support ticket system will allow customers to report any issues that may occur such as a payment dispute or over a server issue. The customers tickets will be checked by an admin. We also plan to have a resolved ticket table which will allow for old tickets to be rechecked if an issue occurred such as bias of an admin.

There are many goals to this project as seen above however the goals of the project are clear, The aims of the project is to achieve a fully functional Django framework website which will can the features mentioned above such as email system, support tickets, or the choice of what kind of server.

**Feasibility of the project**

Over the course of the documentation stage of the project we have divided out the sections of the project each of us is going to work on there is a total of fourteen tasks to do, we have divided these tasks up based on our personal capabilities. We have figured out that worst case scenario it will take us twenty-six days in total to program these systems, optimize them, style them and polish them. Risks are minimal the design of the features planned may end up requiring slight reconfiguration during the development process which may set us back a little time but nothing too ridiculous. As for a contingency plan we are leaving less important systems for later in the development process systems that can be completely left out if necessary. In the first iteration we should have all the main systems in place with basic styling but entirely functional, In the second iteration we will add some extra functionality to the site that is less important but nice additions either way and we will test the functionality thoroughly and optimize all the features as well as possible. In iteration three we will style the site to make it look as nice as possible and putting on the finishing polish. All of this is well feasible within the given time frame as the team is a relatively capable team.



# Use Case: Add product (Admin)

## Description

The admin adds a product to the store front, making it available to buy by customers.

## Trigger

The admin adds a product using the Django admin site.

## Primary Actor

The primary actor is the admin of the website.

## Secondary Actors:

There are no secondary actors for this use case.

## Preconditions

The admin must be logged in with valid admin credentials.

## Main Success Scenario

1. The admin clicks on the products section on the admin site.
2. The admin clicks ‘new product’
3. The admin enters the details of the product.
4. The admin saves the product.
5. The product is now available to purchase on the store.

## Extensions

**Exception**: Admin is logged in without admin privileges.

1. The admin must logout.
2. The admin must log in with valid credentials.

### Success End Condition

A new item is added onto the store front.

### Security

Only the admins of the website can add new products or access the admin site.

# Use Case: Add to Cart

## Description

The customer has to ability to add any product that is available to the cart and that product to then be placed into their cart.

## Trigger

The customer clicks on the add to cart button on the product page that they are currently on.

## Primary Actor

The primary actor is the customer who is wanting to add an available product to their cart.

## Secondary Actors:

* The admin who will support the customer if an issue occurs when adding a product to their cart.

## Preconditions

The customer must be on the site and have an account.

## Main Success Scenario

1. Customer clicks on the Add to Cart button.
2. The product is successful added to the cart
3. Customer is redirected to their Cart.
4. The Customer has the option to continue shopping.
5. The Customer can also go to payment.

## Extensions

**Exception**: Submitted request fails:

1. A warning message is displayed.
2. Customer press's add to cart button again.
3. Continue from step 3.

**Exception:** Submitted request is unavailable:

1. A warning message is displayed.
2. Customer redirected to product page.
3. Continue from step 1.

### Success End Condition

The customer is redirected to the cart with other relevant details.

The product is visible to the customer in the cart.

### Security

There is no security risks.

# Use Case: Change privileges (Admin)

## Description

The either adds or removes privileges to a user or a group.

## Trigger

The admin clicks into the user or group they wish to amend privileges.

## Primary Actor

The primary actor is the admin of the website.

## Secondary Actors:

The customer and the group whose privileges are being changed.

## Preconditions

The admin must be logged in with valid admin credentials.

The customer or group that will have their privileges changed must already exist.

## Main Success Scenario

1. The admin clicks into the customer/group.
2. The admin now either adds or removes privileges to selected customer/group.
3. The admin saves those privileges.
4. The privileges are saved.

## Extensions

**Exception**: Admin is logged in without admin privileges.

1. The admin must logout.
2. The admin must log in with valid credentials.

**Exception**: The customer/group are not yet made.

1. The admin must wait for a customer to sign up or create a group from existing customers.
2. Continue from step 1.

### Success End Condition

Privileges are changed.

### Security

Only the admins of the website can change privileges or access the admin website.

# Use Case: Checkout cart

## Description

The customer checkouts their cart with existing items and proceeds to payment for the items.

## Trigger

The customer clicks on the checkout button located on the view cart page.

## Primary Actor

The primary actor is the customer who wishes to purchase items.

## Secondary Actors:

There are no secondary actors for this use case.

## Preconditions

The customer must be logged in with valid credentials.

At least one item must be present in the cart.

## Main Success Scenario

1. The customer clicks the checkout button.
2. The customer is the redirected to a page summarizing their order.
3. The customer clicks the pay now button.
4. The customer enters payment details.
5. The customer receives a message upon a successful payment.
6. The customer receives a confirmation email for their order.

## Extensions

**Exception**: Cart is empty.

1. The customer must add an item before checking-out.
2. Continue from step 1.

**Exception**: payment details incorrect.

1. The customer receives a message.
2. The customer re-enters the details.
3. Continue from step 5.

**Exception**: Insufficient funds.

1. Customer receives message of payment failure.
2. Continue from step 1.

### Success End Condition

The successfully updates checkouts and places order.

### Security

Only the customer has access to their cart.

# Use Case: Create Ticket

## Description

The customer creates a support ticket relating to the issue that they are experiencing. The support ticket has a title the reference number for their order and a description of the issue. An admin may also create a ticket themselves for a customer that they have been in contact with over the phone for example.

## Trigger

The customer clicks on the create ticket button on the customer support page on the site.

## Primary Actor

The primary actor is the customer who is experiencing an issue with the product or site.

## Secondary Actors:

* The admin who will support the customer with his/her issue and who will eventually close the ticket.

## Preconditions

The customer must be logged into the site with his/her own account. If the problem is relating to a specific order the customer must have the order reference number on hand.

## Main Success Scenario

1. Customer clicks on the create ticket button.
2. Customer is redirected to the create ticket form.
3. Customer fills in ticket title, order reference number if applicable and writes a description.
4. Customer clicks the submit button and is redirected to the ticket view.
5. An email is sent to the customer confirming the creation of the ticket containing details of the ticket and other relevant information.

## Extensions

**Exception**: Submitted data is incomplete:

1. A warning message is displayed.
2. Customer enters missing data.
3. Continue from step 4

**Exception:** Submitted incorrect order number:

1. A warning message is displayed.
2. Customer re-enters order number
3. Continue from step 4

### Success End Condition

The customer receives an email confirming the creation of the ticket with other relevant details.

The ticket is visible to the customer and admins within the customer support section.

### Security

Only the customer and the admins can see the customers ticket.

# Use Case: Customize Product

## Description

The customer may use this feature to customize the components that are to go into their server.

## Trigger

Customer clicks on configure server button on the homepage of the website.

## Primary Actor

The primary actor is the customer who configure a custom server to rent.

## Secondary Actors:

* There is no secondary actor in this case

## Preconditions

The user must have an internet connection and be on the website.

## Main Success Scenario

1. Customer clicks on configure server button.
2. Customer is redirected to the customize server form.
3. Customer selects parts via drop down menus and the price is totaled as parts are added.
4. Customer clicks add to cart button.
5. Custom server is added to the cart.

## Extensions

**Exception**: Server missing a vital component on submit:

1. An error message is displayed to the customer.

### Success End Condition

The custom server created by the user is added to his/her cart.

### Security

CSRF token is used on form submit for security

# Use Case: Delete Ticket

## Description

The customer may click on the delete ticket button on their previously created ticket which deletes the ticket from the database for example this may be used when the customer has solved their issue on their own. An admin might use this to delete a ticket that was created as a prank.

## Trigger

The customer clicks on the delete ticket button on their ticket.

## Primary Actor

The primary actor is the customer who wishes to close the ticket by him/herself as he/her figured out the issue on their own and does not want to waste the time of an admin.

## Secondary Actors:

* The admins may use this if the ticket was posted as a prank or derogatory.

## Preconditions

The customer must be logged into the site with his/her own account. The ticket must be a ticket that has been created by the user that is logged in the only exception to this is if the account is an admin account.

## Main Success Scenario

1. Customer/Admin clicks on the delete ticket button.
2. Customer/Admin the user is asked to confirm the delete.
3. Ticket is deleted from the database
4. Customer/Admin is redirected back to the customer support page .
5. Customer is emailed that the ticket has been deleted with all relevant details.

## Extensions

**Alternative**: User selects no in confirmation:

1. Confirmation appears.
2. Customer/Admin selects no.
3. Customer/Admin redirected back to customer support page

### Success End Condition

The customer receives an email confirming the deletion of the ticket with other relevant details.

The ticket is no longer visible to the customer and admins within the customer support section as it has been deleted from the database.

### Security

Only the customer and the admins can delete the customers ticket.

# Use Case: Edit Profile

## Description

The customer has the ability to edit their profile setting such as their password ,email address and username.

## Trigger

The customer clicks on the Settings button.

The customer clicks on edit profile.

## Primary Actor

The primary actor is the customer who is editing their profile for their account.

## Secondary Actors:

* The admin who will support the customer with any issues such as the profile not updating.

## Preconditions

The customer must have a valid account created on the site.

## Main Success Scenario

1. Customer clicks on the Settings button.
2. Customer must click on edit profile button.
3. Customer must refill in their details into the profile form .
4. Customer clicks the submit button and is redirected to the homepage.

## Extensions

**Exception**: Submitted data is incomplete:

1. A warning message is displayed.
2. Customer enters missing data.
3. Continue from step 4.

**Exception:** Submitted incorrect password:

1. A warning message is displayed.
2. Customer re-enters password.
3. Continue from step 4.

**Exception:** Submitted incorrect username:

1. A warning message is displayed.
2. Customer re-enters a valid Username.
3. Continue from step 4.

**Exception:** Submitted incorrect email:

1. A warning message is displayed.
2. Customer re-enters a valid email address.
3. Continue from step 4.

**Exception:** Submitted incorrect address:

1. A waring message is displayed.
2. Customer re-enters a valid address.
3. Continue from step 4

**Exception:** Submitted incorrect payment details:

1. A warning message is displayed.
2. Customer re-enters a valid payment details.
3. Continue from step 4.

### Success End Condition

The customer is signed into the correct account.

The customer can access their account.

### Security

Only the customer should know their own password.

The payments have encryption.

# Use Case: Filter/Search Items

## Description

The customer may use this feature to search keywords and apply filters to the search in a drop-down menu.

## Trigger

The customer types in a search keyword and filters price with slider beside search bar and clicks submit.

## Primary Actor

The primary actor is the customer who wishes to search/filter through the content on the site

## Secondary Actors:

* There is no secondary actor in this case

## Preconditions

The user must have an internet connection and be on the website.

## Main Success Scenario

1. Customer types in search keyword.
2. Slides slider to the price range best suited to the customer.
3. Customer clicks submit button.
4. Customer redirected to a list of all items matching the filter and search keyword.

## Extensions

**Alternative**: Admin no valid search item appears:

1. A no matching items message will be displayed where the products would have been.

### Success End Condition

The customer sees a list of all matching items for the search keyword and price filter.

### Security

There is no security risks in terms of this use case,

# Use Case: Login

## Description

The customer must enter their valid Login Credentials

## Trigger

The customer clicks on the Login button on the Sign in Page or any page with the sign in button.

## Primary Actor

The primary actor is the customer who is Login into their account.

## Secondary Actors:

* The admin who will support the customer with any Login or issues such as forgotten password.

## Preconditions

The customer must have a valid account created on the site.

## Main Success Scenario

1. Customer clicks on the Sign in button.
2. Customer is redirected to the Sign in form.
3. Customer fills in Sign in form with a valid username and password.
4. Customer clicks the submit button and is redirected to the homepage.

## Extensions

**Exception**: Submitted data is incomplete:

1. A warning message is displayed.
2. Customer enters missing data.
3. Continue from step 4.

**Exception:** Submitted incorrect password:

1. A warning message is displayed.
2. Customer re-enters password.
3. Customer has the option to reset password.
4. Continue from step 4.

**Exception:** Submitted incorrect username:

1. A warning message is displayed.
2. Customer re-enters Username.
3. Customer has option to reset username
4. Continue from step 4.

### Success End Condition

The customer is signed into the correct account.

The customer can access their account.

### Security

Only the customer should know their own password.

# Use Case: Refund Ticket

## Description

The admin uses refund ticket to resolve a ticket when the customer has requested a refund for the product and the issue qualifies them for a refund, the admin will receive a positive mark on his profile and the customer will be refunded within 7 days of the issue of the refund.

## Trigger

The admin clicks on the refund ticket button on the customers ticket.

## Primary Actor

The primary actor is the admin who has reviewed the customers refund request.

## Secondary Actors:

* The customer is the secondary actor as he/she will be in communication with the details of the issue and make a case for a refund.

## Preconditions

The admin must be logged into his/her own account. The admin must have carefully reviewed the details of the issue before issuing the refund.

## Main Success Scenario

1. Admin clicks refund ticket button.
2. Admin is met with are you sure option yes/no.
3. Admin selects yes.
4. Admin receives positive mark on his/her profile.
5. An email is sent to the customer confirming the refund of the order and other relevant information.
6. The order is refunded and will be sent back to the customer via the method he/she originally paid and will be retrieved within 7 working days.
7. Ticket is no longer visible to admins/customer in the customer support page and has been deleted from the database.

## Extensions

**Alternative**: Admin selects no in are you sure menu:

1. Admin selects no.
2. Are you sure menu disappears.
3. Page is back to state before button was clicked and nothing changes.

### Success End Condition

The customer receives an email confirming the refund of the order with other relevant details.

The ticket is no longer visible to the customer and admins within the customer support section.

### Security

Only an admin can refund a ticket.

# Use Case: Remove product (Admin)

## Description

The admin removes an existing item from the store front making it unavailable for purchase.

## Trigger

The admin clicks remove product on the Django admin site.

## Primary Actor

The primary actor is the admin of the website.

## Secondary Actors:

There are no secondary actors for this use case.

## Preconditions

The admin must be logged in with valid admin credentials.

## Main Success Scenario

1. The admin clicks on the products section on the admin site.
2. The admin checks the checkbox of the item he wishes to remove.
3. The admin clicks remove on the website.
4. The item is no deleted from the store front.

## Extensions

**Exception**: Admin is logged in without admin privileges.

1. The admin must logout.
2. The admin must log in with valid credentials.

### Success End Condition

A new item is removed from the store front.

# Use Case: Resolve Ticket

## Description

The admin uses resolve ticket to close a ticket when the customers problem has been solved by the admin and the admin will receive a positive mark on his profile.

## Trigger

The admin clicks on the resolve ticket button on the customers ticket.

## Primary Actor

The primary actor is the admin who has resolved the customers issue.

## Secondary Actors:

* The customer is the secondary actor as he/she will be in communication with the admin to try and resolve the issue.

## Preconditions

The admin must be logged into his/her own account. The customer should be satisfied with the resolution given by the admin before the ticket is resolved.

## Main Success Scenario

1. Admin clicks resolve ticket button.
2. Admin is met with are you sure option yes/no.
3. Admin selects yes.
4. Admin receives positive mark on his/her profile.
5. An email is sent to the customer confirming the resolution of the ticket and other relevant information.
6. Ticket is no longer visible to admins/customer in the customer support page and has been deleted from the database.

## Extensions

**Alternative**: Admin selects no in are you sure menu:

1. Admin selects no.
2. Are you sure menu disappears.
3. Page is back to state before button was clicked and nothing changes.

### Success End Condition

The customer receives an email confirming the resolution of the ticket with other relevant details.

The ticket is visible to the customer and admins within the customer support section.

### Security

Only an admin has the ability to resolve the ticket.

### Security

Only the admins of the website can add new products or access the admin site.

# Use Case: Sign Out

## Description

The customer can sign out from their account

## Trigger

The customer clicks on the sign out button on the home Page or any page with the sign out button.

## Primary Actor

The primary actor is the customer who is signing out of their account.

## Secondary Actors:

* The admin who will support the customer with any signing out issues.

## Preconditions

The customer must have a valid account created on the site.

## Main Success Scenario

1. Customer clicks on the Sign out button.
2. Customer has been logged out their account.
3. Customer is redirected to the home page.

## Extensions

**Exception**: Unsuccessful Sign Out:

1. A warning message is displayed.
2. Customer has not been signed out.
3. Continue from step 1.

### Success End Condition

The customer is signed out of the correct account.

The customer cant access their account without reentering their credentials .

### Security

Only the customer and admin can log tout of their own account.

# Use Case: Sign Up

## Description

The customer creates an account in which they select their own username and password.

## Trigger

The customer clicks on the Sign button on the home page on the site.

## Primary Actor

The primary actor is the customer who is wanting to create an account for the site.

## Secondary Actors:

* The admin who will support the customer if an issue occurs when signing up.

## Preconditions

The customer must be on the site.

## Main Success Scenario

1. Customer clicks on the Sign up button.
2. Customer is redirected to the sign up form.
3. Customer fills in sign up form.
4. Customer clicks the submit button and is redirected to the page they were on.
5. An email is sent to the customer confirming the creation of the account containing details of their account and other relevant information.

## Extensions

**Exception**: Submitted data is incomplete:

1. A warning message is displayed.
2. Customer enters missing data.
3. Continue from step 4

**Exception:** Submitted incorrect email:

1. A warning message is displayed.
2. Customer re-enters a valid email.
3. Continue from step 4

**Exception:** Submitted incorrect password:

1. A warning message is displayed.
2. Customer re-enters a useable password.
3. Continue from step 4.

**Exception:** Submitted incorrect username:

1. A warning message is displayed.
2. Customer re-enters a useable and non-offence username.
3. Continue from step 4.

**Exception:** Submitted incorrect address:

1. A warning message is displayed.
2. Customer re-enters a valid address.
3. Continue from step 4.

**Exception:** Submitted incorrect payment details:

1. A warning message is displayed.
2. Customer re-enters valid payment details.
3. Continue from step 4.

### Success End Condition

The customer receives an email confirming the creation of their account with other relevant details.

The username is visible to the customer section.

### Security

The customer picks a valid username and password. Admin have made password need to have contain a number and a capital letter.

# Use Case: Update Cart

## Description

The customer updates their cart. Either by removing or adding quantity of existing items, or by adding new items.

## Trigger

The customer adds an item from the store front.

The customer clicks either the ‘+’ or ‘-‘ button in the cart view page.

## Primary Actor

The primary actor is the customer who wishes to update their cart.

## Secondary Actors:

There are no secondary actors for this use case.

## Preconditions

The customer must be logged in with valid credentials.

## Main Success Scenario

1. The customer adds an item from the store front into their cart.
2. The customer clicks the ‘+’ to increase quantity or the ‘-‘ to decrease quantity.
3. The quantity is updated along with the price.

## Extensions

**Exception**: Not enough items in stock.

1. A warning message is displayed
2. The customer receives the option to be notified when said item is available.

### Success End Condition

The successfully updates the quantity of items in their cart.

### Security

Only the customer has access to their cart.

# Use Case: Empty Cart

## Description

The customer removes all existing items from their shopping cart.

## Trigger

The customer clicks on the empty cart button in the view cart page.

## Primary Actor

The primary actor is the customer who wants to remove all items from their shopping cart.

## Secondary Actors

There is no secondary actors for this use case.

## Preconditions

The customer must be logged in with valid credentials

The customer must have at least one item in the shopping cart.

## Main Success Scenario

1. The customer clicks the empty cart button located on the page.
2. The items are taken away from the cart.
3. The customer now sees an empty cart.

## Extensions

**Exception**: Shopping Cart is already empty.

1. The customer cannot delete any items.
2. The customer must add an item ( check ‘add cart’ use case).

**Exception**: Customer wants to delete only selected items.

1. The user must decrease the number of items in the shopping cart view page. This is dealt with in the ‘update cart’ use case.

### Success End Condition

All items are removed from the cart.

### Security

Only the customer has permissions to delete items from the cart.

# Use Case: Empty Cart

## Description

The customer removes all existing items from their shopping cart.

## Trigger

The customer clicks on the empty cart button in the view cart page.

## Primary Actor

The primary actor is the customer who wants to remove all items from their shopping cart.

## Secondary Actors

There is no secondary actors for this use case.

## Preconditions

The customer must be logged in with valid credentials

The customer must have at least one item in the shopping cart.

## Main Success Scenario

1. The customer clicks the empty cart button located on the page.
2. The items are taken away from the cart.
3. The customer now sees an empty cart.

## Extensions

**Exception**: Shopping Cart is already empty.

1. The customer cannot delete any items
2. The customer must add an item ( check ‘add cart’ use case)

**Exception**: Customer wants to delete only selected items.

1. The user must decrease the number of items in the shopping cart view page. This is dealt with in the ‘update cart’ use case.

### Success End Condition

All items are removed from the cart.

### Security

Only the customer has permissions to delete items from the cart.

Analysis of Requirements

The Server Rental website has a number of requirements for its systems that need to be addressed. Each system will require different methods and solutions to problems it might present. This document is aimed at addressing these systems and going through ways for these said systems to be introduced without causing issues that will halt the development cycle of the project.

# User

The user system will require various factors in order to work properly. Firstly, the user will sign up, providing an e-mail, username and password. These will be stored in the site’s database. These are required in order to tie a user to an account. They also ensure that only the user will have access to their account. Additionally, the user will need to provide a billing address and payment details in order to process purchases made on the site. These will also be stored on our database. Payment details will be encrypted.

# User profiles

Each user will have a unique profile. The profile will allow the user to change their password, e-mail address, billing address and payment details ( if the user wishes to save the details). These changes will be linked to the database and will update the existing tables. We will not keep records of these changes, hence there will be no ability to revert changes once they are made without re-entering the changed details. The user’s profile will also display the servers they have currently rented, their expiry date and any other details regarding it. This will also be stored on our database, mainly between the users table and the rented servers table.

# Admin

Similar to the user, the admin will provide an e-mail, username and password that will be stored on our database. The main difference between an admin account and a user account is that an admin has certain privileges that a user will not have. These privileges include the ability to add or remove products from the store front as well as update existing product descriptions, prices etc. This will be achieved through the use of the Django admin website that is by default provided by Django upon the creation of a project. The admin website might be edited in order to provided functionality that is necessary for our website to run.

# Shop

The shop will require many factors for it to work. Firstly, a table of products from the database will be linked to the shop, allowing us to put products on sale. The display of the products will include the component name, price and specifications, which will all come from the component table in the database. The shop is also going to have a selection of pre-built server available to rent, these pre-built servers will be stored in a server table. Aside from pre-built servers, the shop will allow users to build their own customised servers from a list of components. These components will have their own table in the database. To build your own server, the user will fill the components from drop down menus linked to specific components. Once submitted, the custom-build will be added to the server table. The user will also have the option to share their custom-build server with the store. This will take the custom server from the server table and place it into the product table allowing it to show up on the store front. Component availability doesn’t really matter in this situation as the servers are not yet rented. This will be dealt with in the cart system.

# Cart

The function of the cart is to take products from the shop and create an order for the user. This cart will be session based. The cart will require several functions to operate. Details on these can be found within the use case descriptions. The cart will also require the Stripe API in order to process payments. Additionally, a PayPal API could be introduced to offer an alternative way to pay. Once an item is added to the cart, and the user wishes to proceed with checkout, the site will look to the database to see if the components within the server are in stock. IF not a message will appear and alternatives will be offered. However, once the check is successful the cart will proceed to the payment page. This page will require the user to enter payment details if not saved within their profile. Once the order is complete, the user will receive a uniquely generated order number. This order number will tie the server to the user. The server is then moved from the server table into the rented server table. An expiry date is added as well as the name of the user currently renting the server. However since a user can rent more than one server, and many users can rent a specific configuration, the order number is going to be the primary key linking the user to the specific server.

# Support Tickets

This system will offer support to the users from admins. This will require a support ticket table in the database. Each ticket will have a unique ID and an e-mail associated with it. Additionally a short description of the issue will be attached. These tickets will be stored in the database for the admin to pull out. Once an admin pulls the ticket, they will have the e-mail of the person with the issue, allowing for easy contact. The admin can then deal with the ticket, be it technical support within the server, refunds, or general queries. For refunds, The admin would have to manually expire the server which would move it from rented servers table into the server table. The admin would also then issue a refund manually. A log of tickets and interactions between the admin and user will be kept for documentation. Once the ticket is resolved, it will be moved from the support tickets table to the completed tickets table.

# 