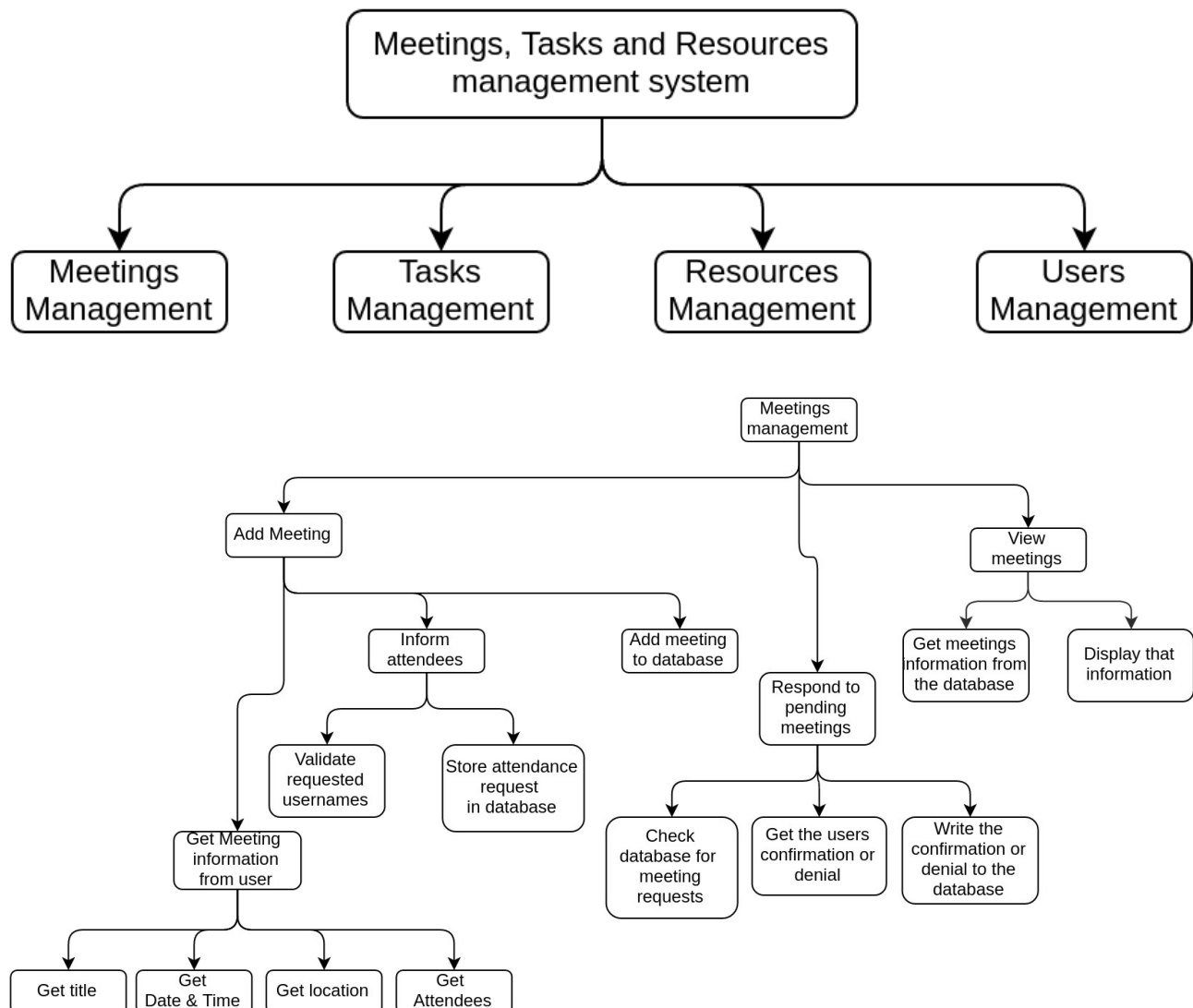
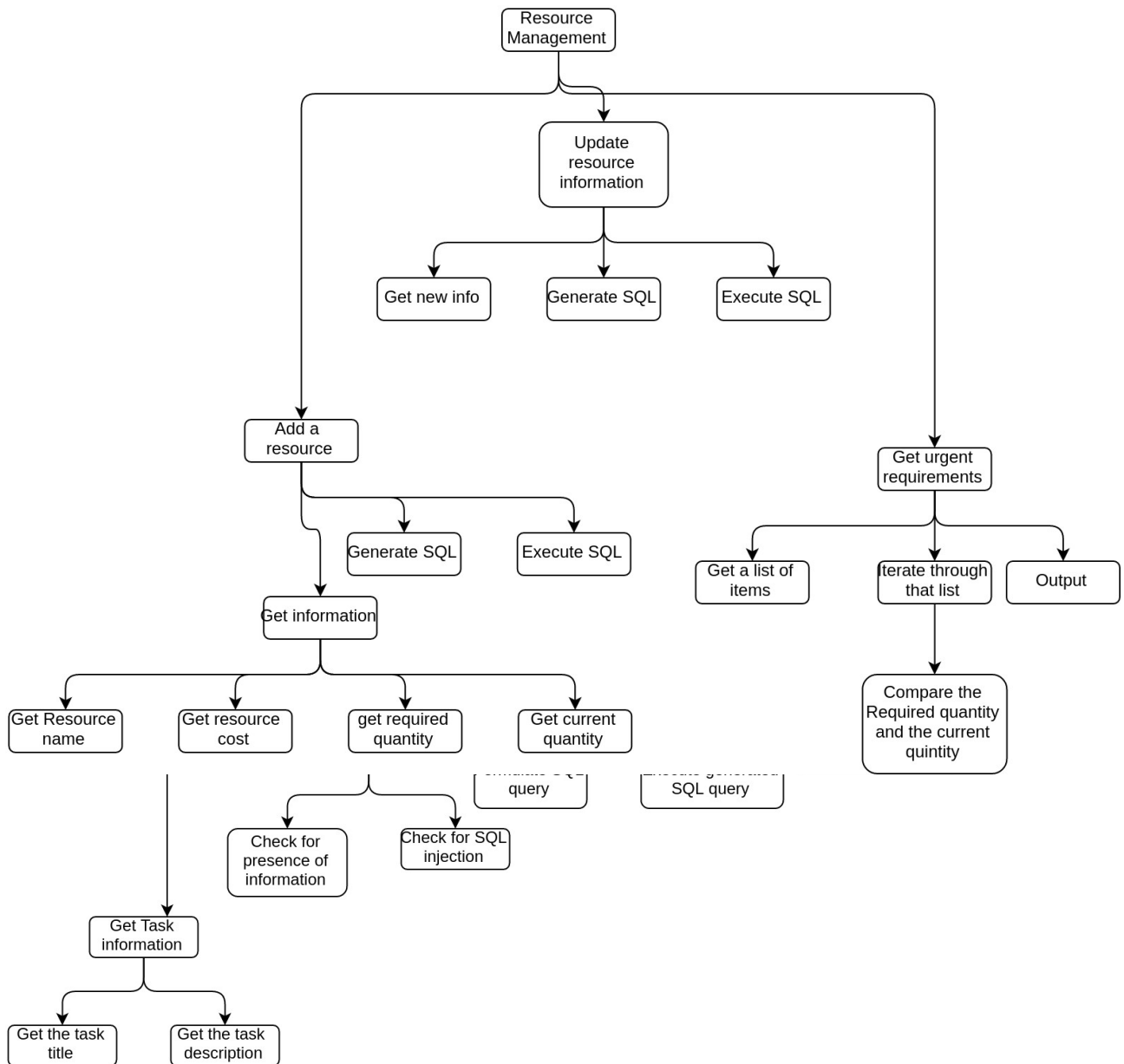


Hardware Specification

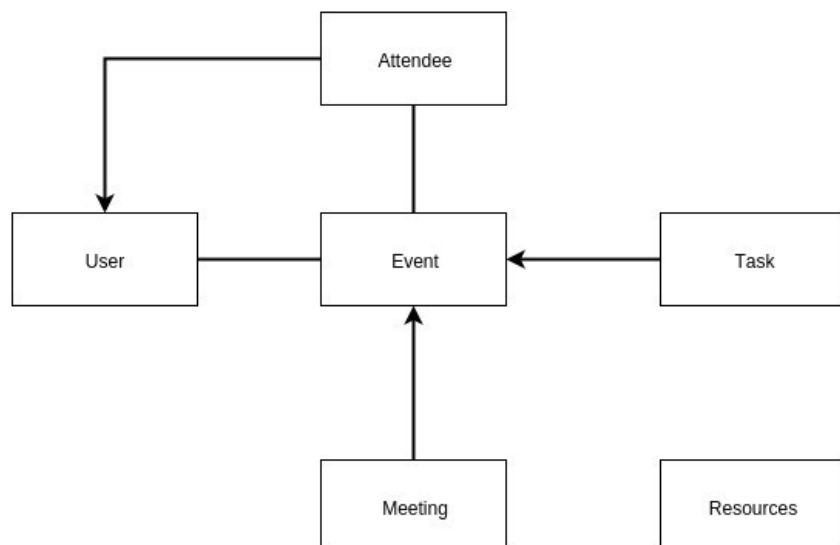
The system will run on both desktops and laptops, with a minimum screen resolution of 1024x768 (4:3 aspect ratio) and all of the PCs are running Windows Vista or later. This is important as the graphic user interface must fit within this area otherwise the users may not be able to operate it. A combination of a keyboard and a mouse will be used for inputting information into the forms and the majority of the navigation within the program will be done using a mouse. All of the storage will be on a remote NAT (network attached storage) and the output will be directly to a monitor.

Program Structure

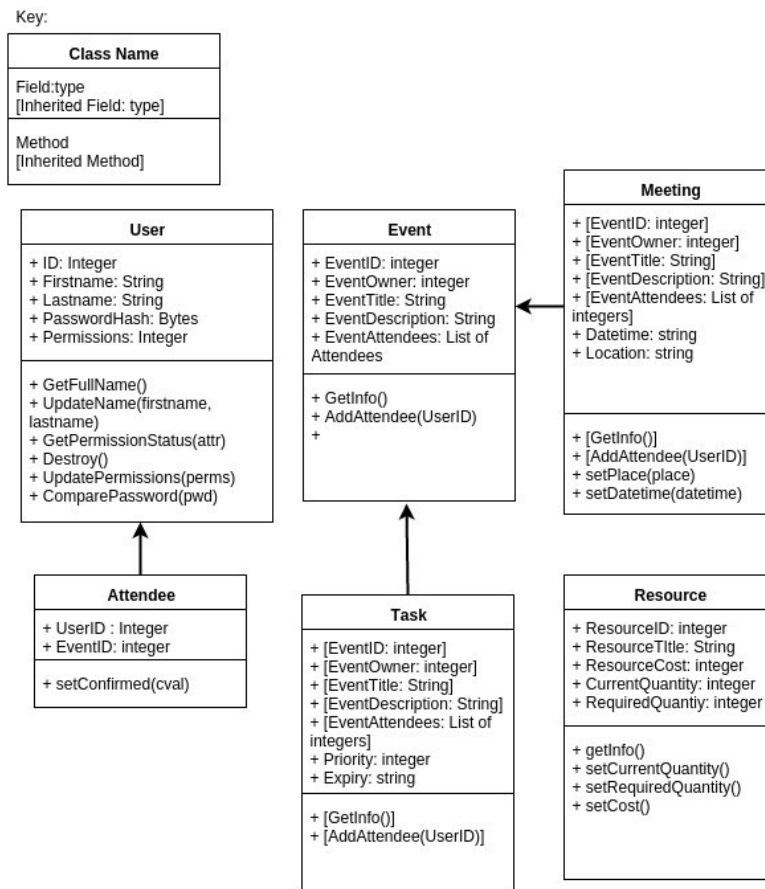




Object Diagrams



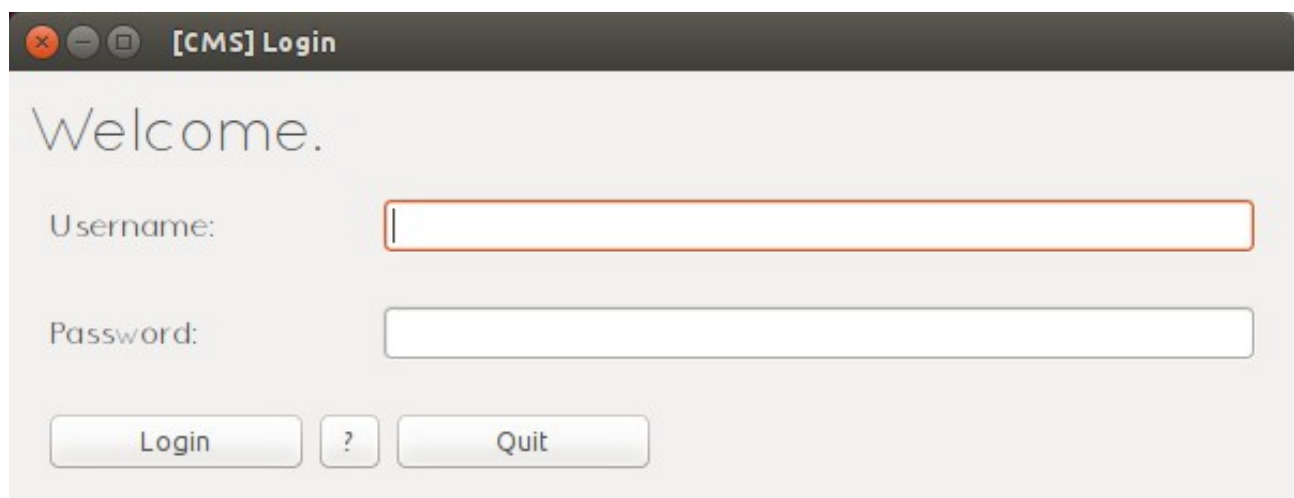
Class Definitions



Prototyping

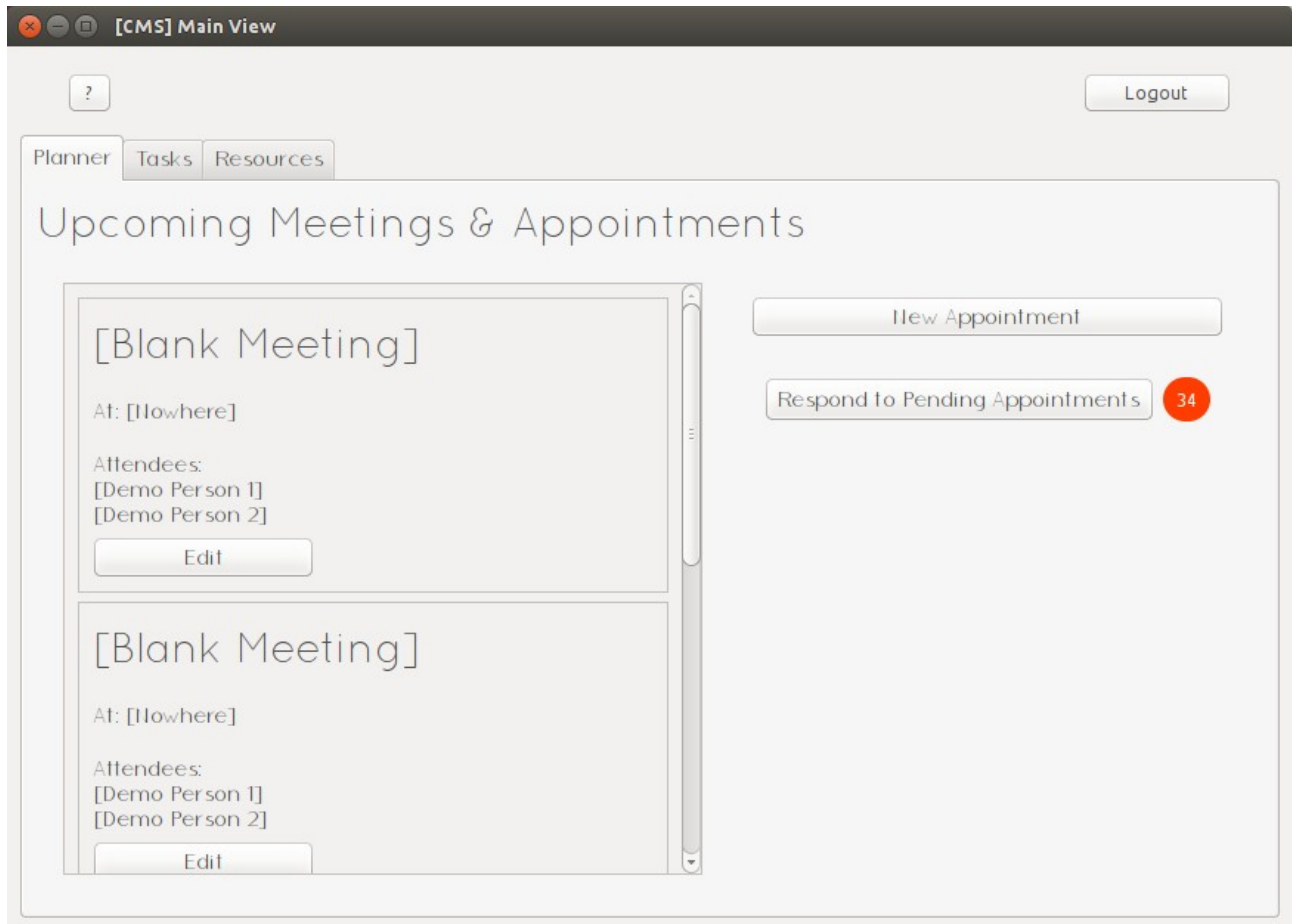
Prototyping allows me to test out different aspects of the system, allowing me to evaluate how easily they can be made using the tools available. The main part of this system is the graphical user interface (GUI), and is the most experimental part, so to make sure the various elements work and work together.

The first part of the system is the login window, this needs to be clear and simple to use so that the users can enter their information with little or no guidance.

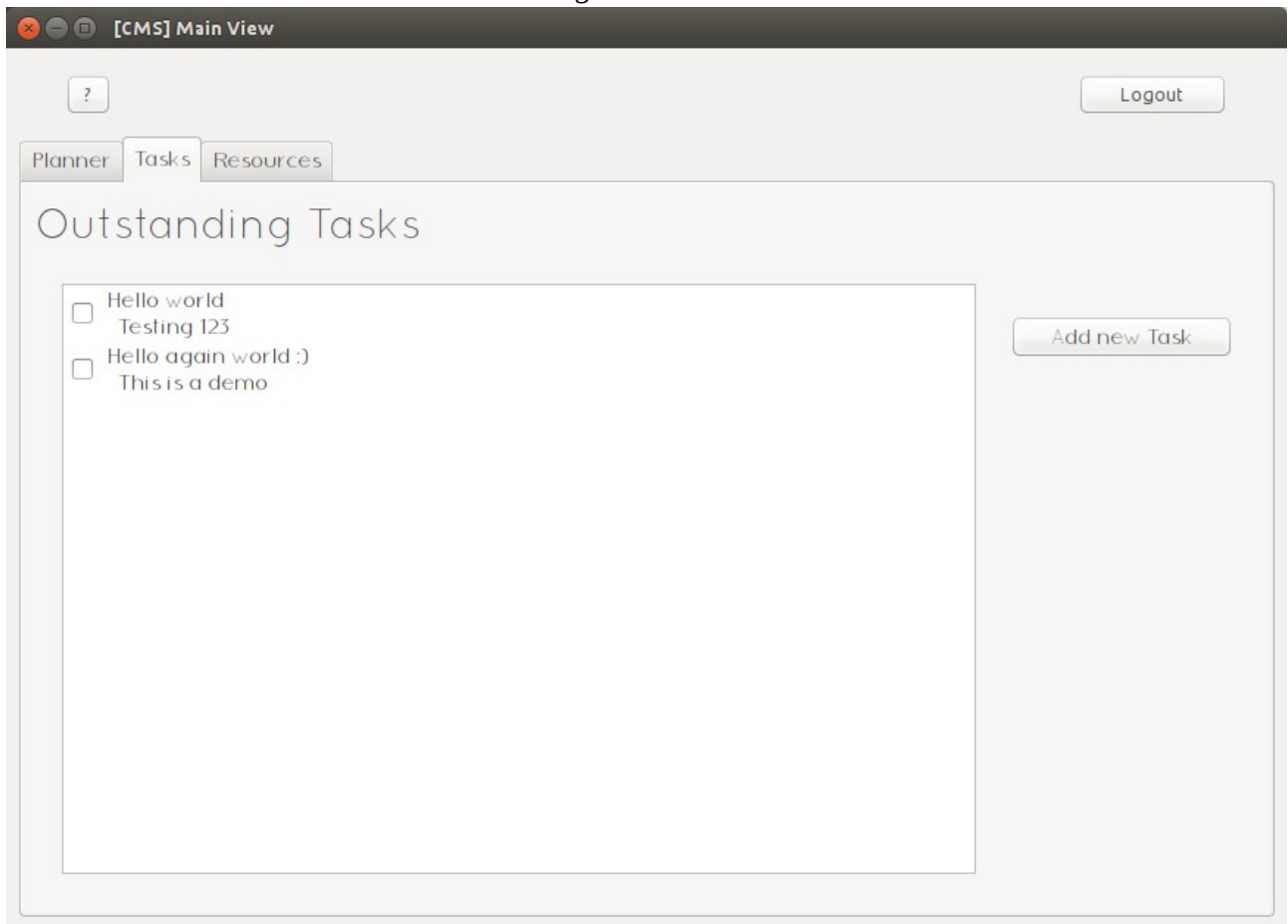


The next part of the system is the meetings list view, which contains a list of all the meetings belonging to a user. The '34' in the orange circle is just to test the indicator widget I designed, in the actual software, it will

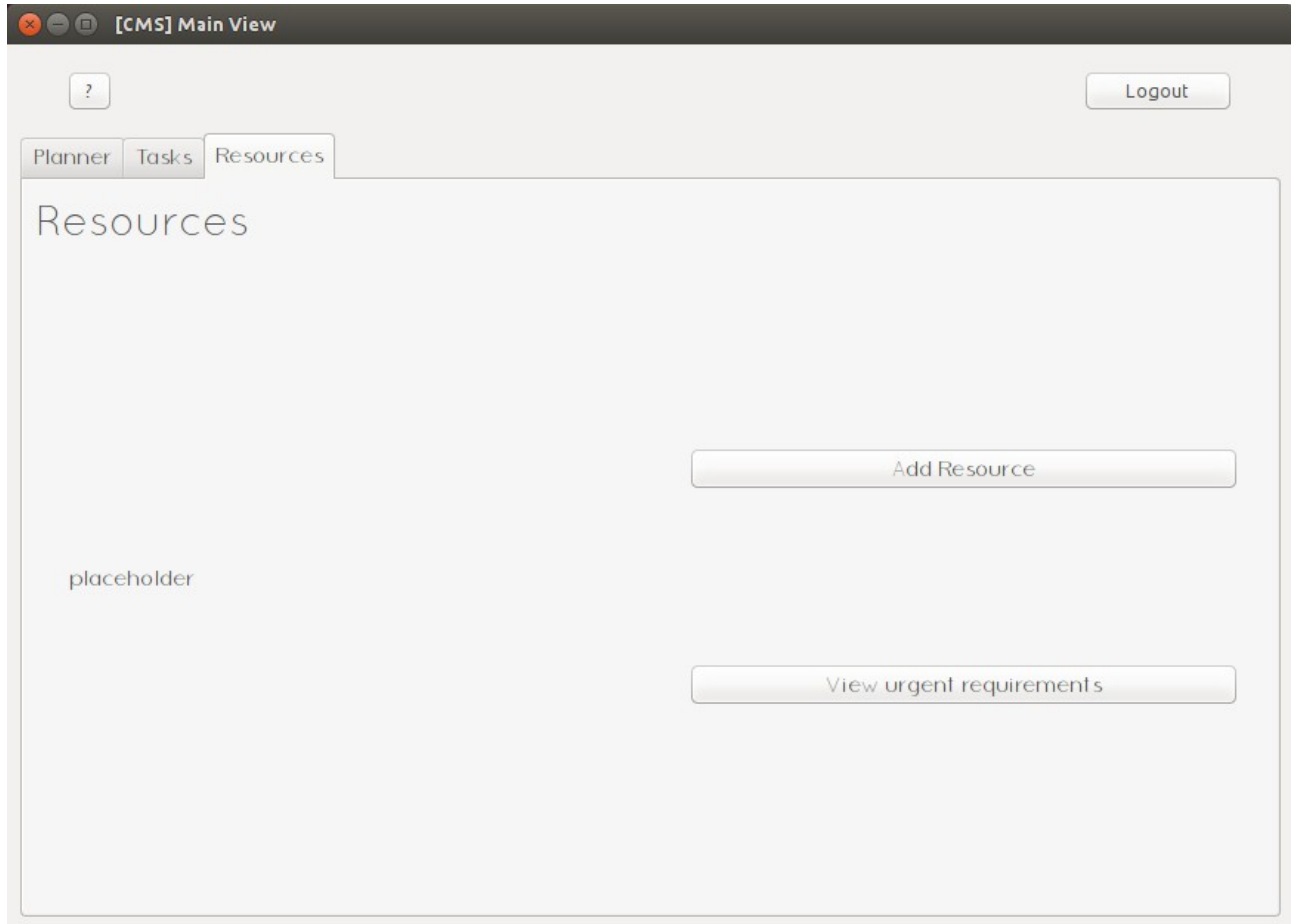
show the number of appointments that the user has been invited to attend but has not responded to yet. The list on the left contains a list of blank placeholder meeting widgets to show what each meeting could look like.



The next part of the system is the tasks management screen, which contains a tick-able list of tasks on the left and various controls to interact with this on the right.



For the resources management, getting a table of data required more time and resources than I initially expected so, the actual table is not included in the prototype, I spoke to my liaison with the client about this and the conclusion of that meeting was that support for the resource management section was of a lower priority than the meetings and tasks subsystems. In this prototype, the resources table is replaced with the

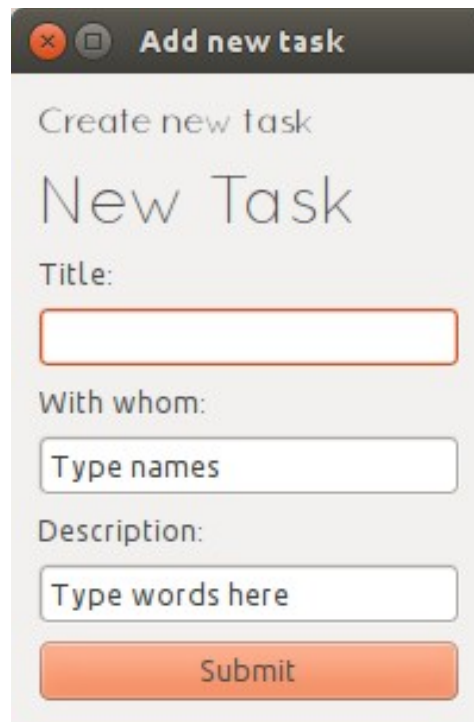


word "placeholder".

For the input form for adding new meetings, ease of use is of paramount importance, so I styled the form in a way that is similar to standardized input forms found in mobile applications, which is something most members of the team are familiar with so it follows a logical order of organisation of information.

A screenshot of a mobile application form titled "Add New Meeting". The form has a dark header bar with window controls and the title "Add New Meeting". The main content area is titled "Add New Meeting" and contains four input fields: "Meeting Title:", "Attendees:", "Where", and "When". Each input field is a simple white rectangle with a thin border. At the bottom of the form, there are two buttons: "Save" (orange) and "Cancel" (gray).

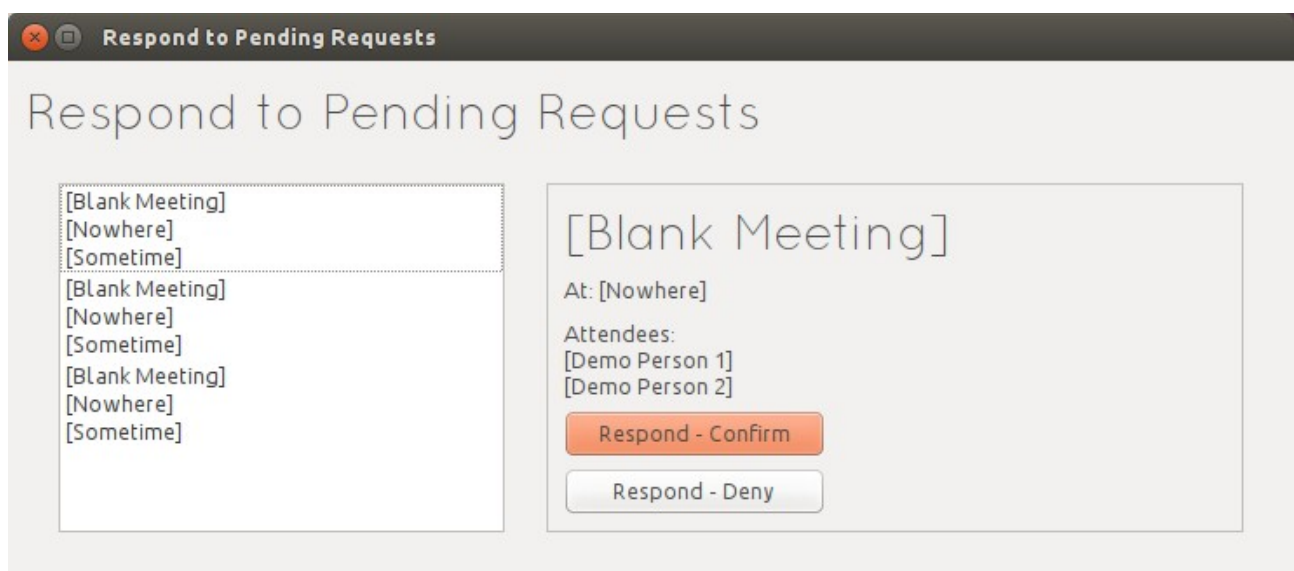
I used a similar set of design principles when designing and prototyping the form to add tasks



A screenshot of a web form titled "Add new task". The form has a dark header bar with a close button (red circle with an 'x') and a maximize button (grey square). Below the header, the text "Create new task" is followed by a large heading "New Task". The form contains three input fields: "Title:" with a red border, "With whom:" with a placeholder "Type names", and "Description:" with a placeholder "Type words here". At the bottom is an orange "Submit" button.

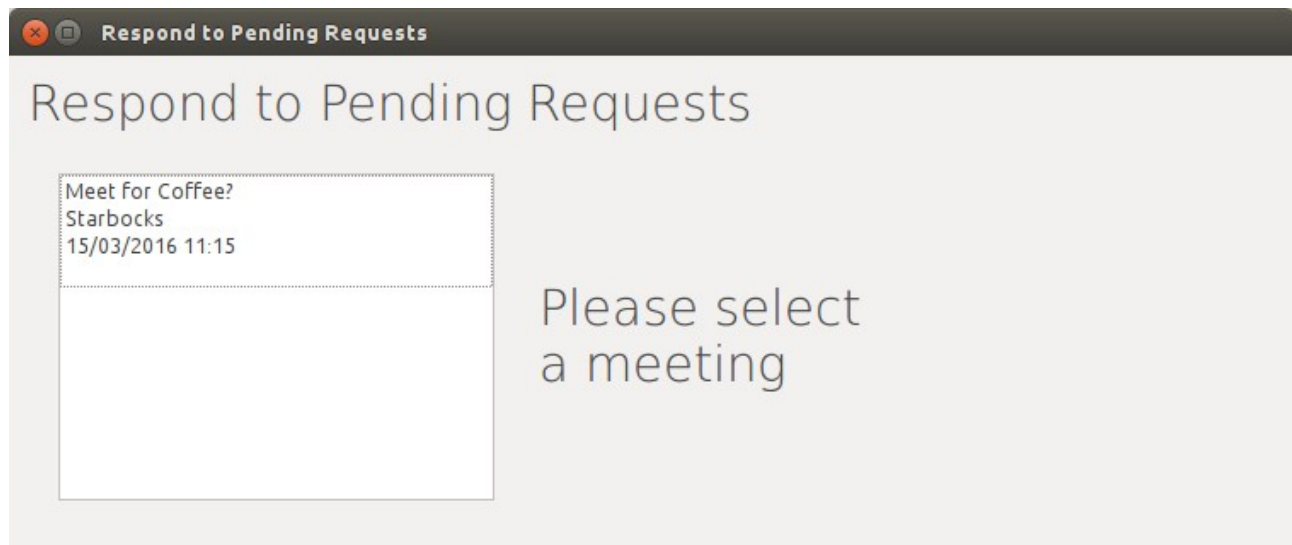
I'm still unsure if tasks need to have the option to add attendees, so I included it just in case.

For responding to meetings, I used the same system of a list of options and information on the left and more detailed information and controls on the right.



A screenshot of a web form titled "Respond to Pending Requests". The form has a dark header bar with a close button (red circle with an 'x') and a maximize button (grey square). Below the header, the text "Respond to Pending Requests" is followed by a large heading "Respond to Pending Requests". The form is divided into two main sections. The left section contains a list of meeting options: "[Blank Meeting]", "[Nowhere]", "[Sometime]", "[Blank Meeting]", "[Nowhere]", "[Sometime]", "[Blank Meeting]", "[Nowhere]", and "[Sometime]". The right section contains a large text area with the heading "[Blank Meeting]", followed by "At: [Nowhere]", "Attendees: [Demo Person 1]", and "[Demo Person 2]". At the bottom of the right section are two buttons: "Respond - Confirm" (orange) and "Respond - Deny" (grey).

I realised when I created this prototype that it might not be obvious that a user should select a meeting from the list on the left, so I changed it so that if the user has not selected a meeting, a message is shown as below:



Definition of Data Requirements

Identification of all Data Input Items

- User name
- Password
- Meeting title
- Meeting location
- Meeting date & time
- Meeting attendees
- Task title
- Task description
- Task completed?
- Meeting request accepted or rejected
- Resource title
- Resource cost
- Resource current amount
- Resource required amount
- Updated password

Identification of all data output items

- Meeting title
- Meeting date & time
- Meeting location
- Meeting attendees
- List of confirmed attendees
- Task title
- Task description
- Resource name
- Resource cost
- Resource quantity
- Resource required quantity
- List of urgently required resources

Output to database

- Meeting title
- Meeting date & time
- Meeting location
- Meeting attendees
- Meeting Owner
- Task title
- Task description
- Task owner
- Resource name
- Resource cost
- Resource quantity
- Resource required quantity

Explanation of how Data Output is generated

Context	Output	How it's generated
Viewing meetings in the meeting list found in the 'Planner' tab	Meeting Title	Fetched from the table 'Meetings' in the database. All of the data in this table is generated when a user adds a meeting.
	Meeting Date & Time	
	Meeting Location	
	Meeting Attendees	
Viewing Requested meetings	Meeting Title	
	Meeting Date & Time	
	Meeting Location	
	Meeting Attendees	
	Meeting Owner	Referenced from the table 'Users' in the database, is associated with a meeting when it's created.
Viewing a list of tasks	Task Title	Taken directly from the 'Tasks' table of the database, which contains data the user input into the 'add task' form.
	Task Description	
Updating a task	Task title	
	Task Description	
Viewing Resources	Resource name	Taken directly from the 'Resources' table in the database, which contains data that comes directly from the user.
	Resource cost	
	Resource quantity	
	Resource required quantity	
Viewing a list of urgently required resources	Resource name	Selected from the database table 'Resources' on the condition that the current quantity is lower than the required quantity.
	Resource cost	
	Resource quantity	
	Resource required quantity	

Data Dictionary

Name	Data Type	Length	Validation	Example Data	Comment
UserID	Integer	2 Bytes	Number, unique	35	Is used to reference individual users across the system, must be unique.
UserName	String	45 chars	No more than 45 chars, must not be empty	Steve Johnson	Is used to identify users in a human readable way.
UserPassword	String	32 chars	Md5 checksum	35ff99022aa9a2c6	The system never

Name	Data Type	Length	Validation	Example Data	Comment
			– only in hexadecimal characters	8d11b2b820579532	stores the user's passwords in plaintext, so the password is stored as a md5 checksum
UserAccessRights	Integer	2 Bytes	Must be present, between 0 and 31.	30	A 5-bit integer used to represent the user's access to various areas of the system.
UserUsername	String	60 chars	Must be present & unique	SteveJohnson	Used for logging in and referring to other users when requesting attendance.
MeetingID	Integer	2 bytes	Must be present and unique	478	Used to reference individual meetings across the system
MeetingOwnerID	Integer	2 bytes	Must be present, unique and a valid UserID	35	A foreign key referring to the Users table
MeetingTitle	String	60 chars	Must be present	Coffee with James	Holds the title for the meeting
MeetingDateandTime	String	45 chars	Must be present and follow the format: YYYY/MM/DD hh:mm	2016/02/15 14:23	Holds the date and time for the meeting in ISO standard formatting, can easily be used with Python's <i>datetime</i> library.
MeetingLocation	String	45 chars	None	Costa Coffee, Grand arcade	Contains the location of the meeting
TaskID	Integer	2 bytes	Must be present and unique	348	Is a unique identifier for each task
TaskTitle	String	45 chars	Must not be longer than 45 chars and must be present	Get lunch for the family	Contains a title for each task, used by the user to identify tasks
TaskDescription	String	255 chars	None	Go to Sainsco's and get some vegetables and oven chips	Contains a short description about the task for the user.
ResourceID	Integer	2 bytes	Must be present and unique	75	A unique identifier used to reference

Name	Data Type	Length	Validation	Example Data	Comment
					resources throughout the system
ResourceName	String	45 chars	Must be present	Fairtrade instant coffee	A human readable identifier for various resources.
ResourceCost	Integer	2 bytes	Must be present, defaults to zero if not present	129	The cost of an item, in pence.
ResourceQuantity	Integer	2 bytes	Must be present, defaults to zero if not present	29	The amount of an item currently in stock
ResourceRequiredQuantity	Integer	2 bytes	Must be present, defaults to zero if not present	23	The minimum amount of the resource that can be in stock without raising warnings.

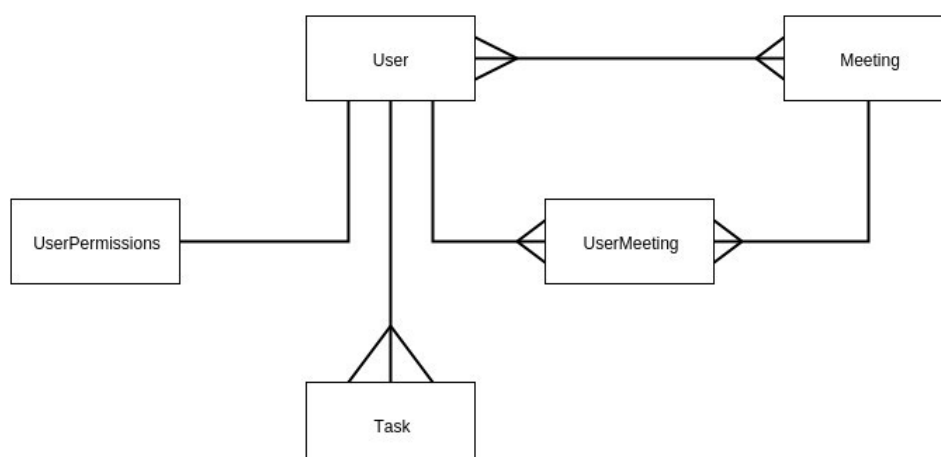
Identification of Appropriate Storage Media

Because there is potential for there to be a large amount of data, which is likely to be changed and updated frequently, a single non RAID spinning HDD would be ample to store the data, because the system is only going to be accessed by a maximum of 10 users simultaneously, speed increasing RAID is not necessary, however a redundancy drive would be useful to provide a continuous backup of the data. Solid state drives would not be appropriate because of the continuous rewriting of storage locations which would relatively quickly wear out a solid state disk which would not be cost effective to replace.

Database Design

Normalisation

ER-Diagrams



UNF to 3NF

Un-normalised
UserUsername UserName UserID UserPermissions UserPassword MeetingID MeetingOwner MeetingTitle MeetingLocation MeetingDateandTime MeetingAttendees MeetingConfirmed? TaskID TaskOwner TaskTitle TaskDescription ResourceTitle ResourceCost ResourceID ResourceQuantity ResourceRequiredQuantity

1NF	
Repeating	Non-Repeating
MeetingID	UserUsername
MeetingOwner	UserName
MeetingTitle	UserID
MeetingLocation	UserPermissions
MeetingDateandTime	UserPassword
MeetingAttendees	ResourceTitle
TaskID	ResourceCost
TaskOwner	ResourceID
TaskTitle	ResourceQuantity
TaskDescription	ResourceRequiredQuantity
MeetingConfirmed?	

2NF	
Repeating	Non-repeating
MeetingID MeetingOwner MeetingTitle MeetingLocation MeetingDateandTime MeetingID UserID Confirmed?	UserUsername UserName UserID UserPermissions UserPassword ResourceTitle ResourceCost ResourceID ResourceQuantity ResourceRequiredQuantity

TaskID UserID TaskTitle TaskDescription	
--	--

3NF
Meeting MeetingID OwnerID MeetingTitle MeetingLocation MeetingDateAndTime
User UserID UserUsername UserName UserPassword UserPermissions
MeetingAttendee UserID MeetingID Confirmed?
Task TaskID OwnerID TaskTitle TaskDescription
Resource ResourceID ResourceTitle ResourceCost ResourceQuantity ResourceRequiredQuantity

Entity Relationships

Meeting(MeetingID, OwnerID, MeetingTitle, MeetingLocation, MeetingDateAndTime)

User(UserID, Username, Password, Name, Permissions)

MeetingAttendee(UserID, MeetingID, confirmed?)

Task(TaskID, OwnerID, TaskTitle, TaskDescription)

Resource(ResourceID, ResourceTitle, ResourceCost, ResourceQuantity, ResourceRequiredQuantity)

SQL Queries

These are the frameworks for generating SQL queries that I have used throughout this project. The reason I say 'frameworks' is because the SQL queries will be passed through python's string formatter with the appropriate information included.

SQL	Description
CREATE TABLE IF NOT EXISTS Users (UserID INTEGER PRIMARY KEY AUTOINCREMENT ,Name TEXT , Username TEXT , Password TEXT , Permissions INTEGER);	This is an example of an SQL statement to create a table within a database, note the use of the AUTOINCREMENT keyword, this ensures that all of the values in that field are unique, hence it is useful for defining primary keys.
INSERT INTO Users(Name, Username, Password, Permissions) VALUES ({0});	This is an example of a query to add a user to the table Users, the payload represented by the "{0}" is a comma seperated list of values for the Name, Username, Password and Permissions
SELECT * FROM Users {0} ;	This is a simple sql query for getting all of the information about a particular user, the part of the query denoted with "{0}" will be replaced by something similar to "WHERE UserID = 20" to select a specific user.
SELECT UserID FROM Users WHERE (Username = '{0}');	This is a similar query, designed to get the UserID based on username. This is useful for the login process, where a 'user' object is selected by username.
UPDATE MeetingAttendee SET Confirmed = 1 WHERE {0}	This statement will update the value of if a meeting request is confirmed or not to 1 using a SQL update statement. The placeholder denoted by "{0}" will be replaced with something similar to "MeetingID = 34 and UserID = 2"
DELETE FROM MeetingAttendee WHERE {0}	This statement deletes a meeting attendance request, the placeholder "{0}" will be replaced with something similar to "MeetingID = 39 AND UserID = 3" or just "MeetingID = 23" to delete all of the requests for a specific meeting.

(Note: A complete listing of SQL constructs will be found in the code listing later on in the project)

Security & Integrity

This system will store sensitive information about potentially vulnerable people, which means the storage of this data is encompassed under the Data Protection Act 1998 (DPA). This means the personal information must be kept up to date, and kept securely. The data stored shouldn't be stored any longer than necessary, in the case of the Church, they currently have a policy of renewing data annually (when they get new dairies), so the database has to be implemented in a way that makes it easy for the system to comply with this policy. The Church currently has a secure file server located on site which they currently use for sharing and storing other private information in accordance to the DPA, each member of staff has a username and password which they can use to access this server and all attempts to access it are logged.

When the data is input, it needs to be validated to ensure it is correct (also necessary for compliance with the DPA), part of the validation can be ensure by using drop down menus wherever possible. I will also need to ensure the database is not left with missing dependencies when data is added, removed or changed, I will do this with software checks as part of the system that run when data is transferred in and out of the database.

System Security

Because this system is critical to the operation of the Church's administrative and pastoral aspects, it is important that the data is not interfered with, lost or stolen. Access to the system is controlled by username-password pairs and there will be no support for 'guest' accounts. If there is an invalid username, password or combination of the two, no access to the system will be granted. The system has support for user permissions, so that each user can be given access to specific areas of the system, as required.

The conditions of complying with the Data Protection Act are as follows:

- The data will remain in the UK
- The data will be held securely, and is inaccessible to unauthorised people
- The data stored will only be used by the Church and will not be disseminated under any circumstances
- The data will be destroyed as soon as it is no longer required
- The data is kept up-to date and accurate
- Only necessary data will be collected and stored.

Validation

Item	Example	Validation	Comments
Username	username1	Presence Check, Verified against the database	The entry form immediately cancels it's verification if left empty, obviously the username is checked against the database to determine if it's a valid user.
Password	Pa\\$\w0rd	resence Check, checksum	The entry form

Item	Example	Validation	Comments
		verified against the database.	immediately cancels if this is left blank, if not left blank, the system will generate a hash of the user's input and then check that against the hash stored in the database.
Meeting Title	Dinner With the Smiths	Presence Check	To ensure that a title is entered, this is how the meetings are identified for the user, so it's essential that they have a name. Because often there are recurring meetings, it is not necessary to ensure that the meeting title is unique.
Meeting Attendees	username1, username2	Validated against the database	Usernames which are not recognised are not used when adding the MeetingAttendee database entries. The user should be informed when an invalid username is entered.
Meeting Location	Norwich Forum	Presence Check	The user is informed if the location field is left empty and the form will refuse to submit.
Meeting Start Date & Time	1/28/16 7:54PM	Discrete set of values	This will be a spin box, which is a widget that only allows the user to select input from a set list of values.
Task Title	File A Tax Return	Presence Check and enforced truncation	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required. If the entered data is too long, it will be truncated and an ellipsis will be appended to the text.
Task Description	Make sure you fill out every single little field	Presence Check	If the field is left empty when the user attempts to submit the form, the form

Item	Example	Validation	Comments
			will not submit and the user will be reminded gently that that field is required.
Resource Name	Teabags	Presence Check	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required.
Resource Quantity	100	Presence Check, Integer Validation	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required. The system will attempt to change the value into an integer, if it fails it will notify the user and the form will not submit.
Resource Cost	£10.40	Presence check, regex check	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required. A regular expression will be used to check if the entered price is in a valid format (either £{int}.{int} or {int}p)
Resource Required Quantity	200	Presence Check, Integer Validation	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required. The system will attempt to change the value into an integer, if it fails it will notify the user and the form will not submit.
New Password	P4\$w0Rd	Presence Check	If the field is left empty when the user attempts to submit the form, the form will not submit and the user will be reminded

Item	Example	Validation	Comments
			gently that that field is required.
Confirm New Password	P4\\$\$w0Rd	Presence Check, verified against field "New Password"	If the field is left empty or does not match the field "New Password" when the user attempts to submit the form, the form will not submit and the user will be reminded gently that that field is required.