







# Design Documentation

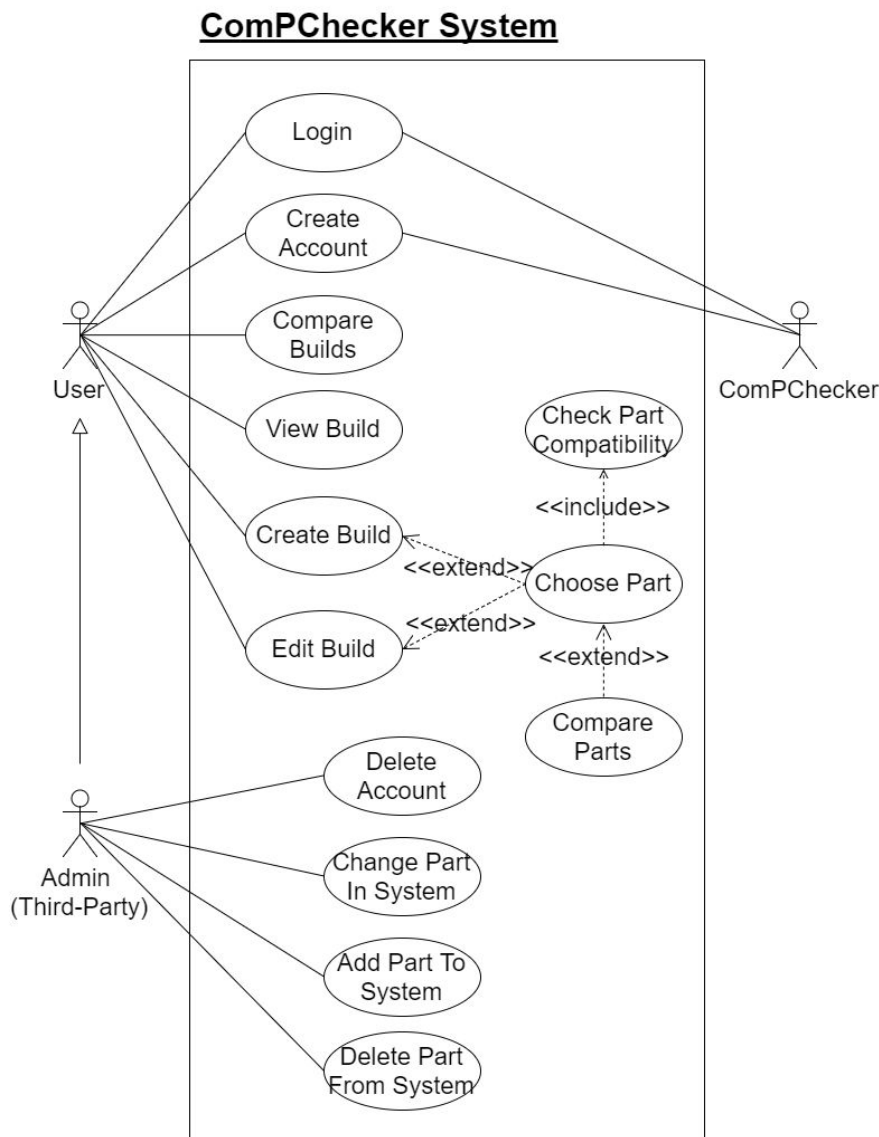
Team 7A  
ComPChecker

## Authors

<u>Name</u>	<u>Main Section and Contributions</u>	<u>Signature</u>
Thomas Chate	Architectural Pattern Design USE Case tables Mockup Write Up State Behaviour Diagram	
Lillie Hogg	Architectural Pattern Design Architectural Pattern Write Up State Behaviour Diagram	
Luke Jackson	Use Case Diagram and accompanied descriptions Contributed to GUI Descriptions	
Greg Lindert	Sequence Diagrams - description and helped with creation	
Tomasz Szelachowski	Sequence Diagrams - Creation	
Pawel Szymczyk	GUI Mockups	



## USE CASE



## **Group 7A**

The USE Case diagram illustrates the functions carried out by CompPChecker, both general users and admins can create accounts, but only an admin can delete an account. When it comes to logging in, there will only be one form/method to do this, however based on the credentials entered, the user will be taken to either the general user menu or the admin menu. Both the create build and edit build methods use the same method for choosing parts, except when editing a build, any pre-selected part is just overwritten. The admins can do everything in the system that the general users can do, mainly for testing purposes. The admins also have control over which parts are in the system database, so they can add/remove/edit as they see fit.



<u>Login</u>	
Actors	General Users, Admin Users
Description	The first process used by all users to gain authentication into the system.
Data	Username, password
Stimulus	To access the system.
Response	The user can either be authenticated (with their user type being returned as was as ID) or rejected.
Comment	For the Login method, there will only be one form/method to do this, however based on the credentials entered, the user will be taken to either the main menu or the admin menu.

<u>Create Build</u>	
Actors	General User
Description	This process creates a build and saves it.
Data	Motherboard, CPU, RAM, GPU, SPU, Case, Storage, and Accessory.
Stimulus	This is the main function of the system.
Response	The build is saved in the database
Comment	During this process the 'Check Compatibility' process is called to ensure all parts work with one another.

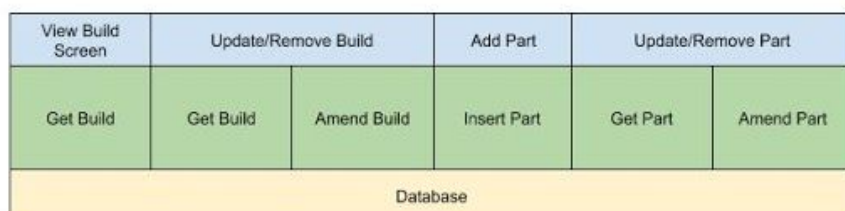
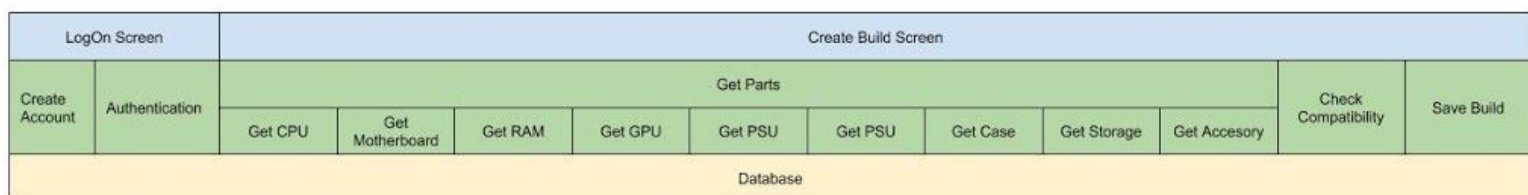
<u>Check Part Compatibility</u>	
Actors	System
Description	Checks that parts are compatible with one another by checking the database compatibility table.
Data	Motherboard, CPU, RAM, GPU, SPU, Case, Storage, and Accessory.
Stimulus	The system needs to run this before saving to ensure users can build their PC without any issues.
Response	Returns true or false depending on the check.
Comment	The compatibility table stores two part IDs and a boolean value which indicates its compatibility.



USE CASE4: Add Part	
Actors	Admin User
Description	This function adds a new part to the system (database).
Data	Data is dependant on part, as each part has different variables (a case does not have a clock speed value but a CPU would). All part types will store price and an amazon link.
Stimulus	New parts are frequently released released by manufacturers, these need to be added to ensure the system is up to date.
Response	Part is stored in database.
Comment	

View Build	
Actors	General User
Description	Allows a user to view a previously created build.
Data	Build ID
Stimulus	A user may want to view their previous built builds.
Response	Motherboard, CPU, RAM, GPU, SPU, Case, Storage, and Accessory.
Comment	The admin will use this function for testing purposes.

## System Architecture



\*Note diagram continues on a new line for clarity issues; the pattern should be one long rectangle.

Shown above is the system architecture design for the ComPChecker system. The architectural pattern chosen is a closed 3 layered architecture. The diagram above covers the main functionality of the system; creating a build and checking its compatibility. This pattern was chosen as it reflects our system, and each part of the system can be represented by a different layer, user interface (presentation), logic and database, using the layered architectural pattern clearly shows the process of the application systems. Additionally, it allows us to breakdown functions into different platforms of interacting modules. A problem which may occur when implementing the



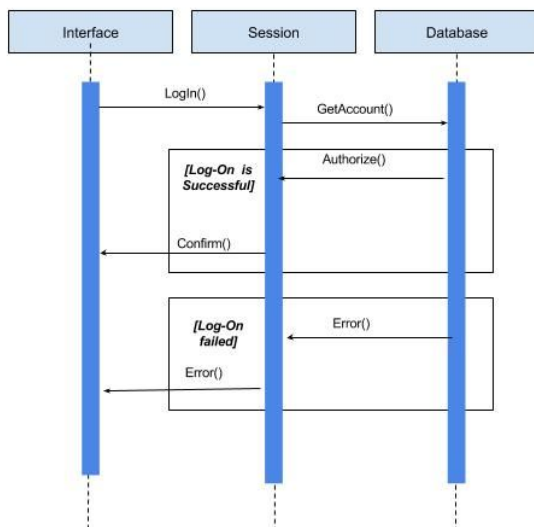


layered architectural pattern will be the performance, as the application could take longer to process data because the layers could initially prevent the user interface from trying to interact with the database, slowing the speed of the system.

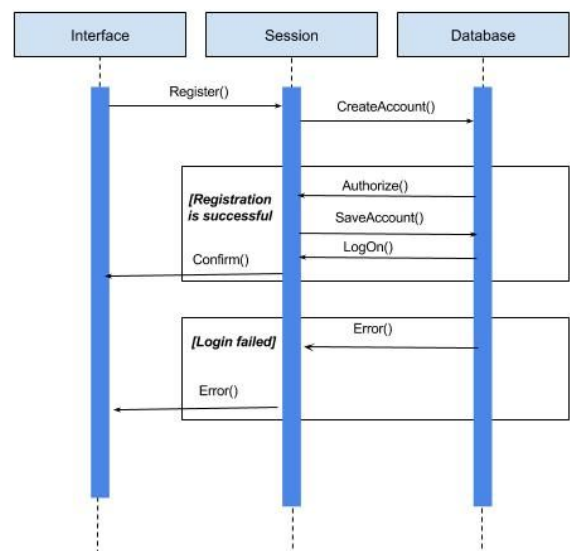
When we first started out designing our system architecture pattern we chose to create a pipe and filter system. We then found that this method did not clearly illustrate our system due to our system not having a set workflow. We later adapted our pipe and filter into a system behaviour state model which can be found at **appendix 1**. We then attempted an object oriented approach, with each 'part' representing a class; however we found this unsuitable. We then broke the system down and found that the 3 layered, and more specifically closed 3 layer, pattern matched our system

## Sequence Diagram

**Sequence Diagram 1: LogOn**



**Sequence Diagram 2: Create Account**



### **Sequence Diagram 1:**

A user is able to log in if they have an account. To access the system the user must provide a correct username and password combination.

### **Sequence Diagram 2:**

An account can be created by a user to access the system. The account details can either be saved (if that username isn't taken) or rejected.

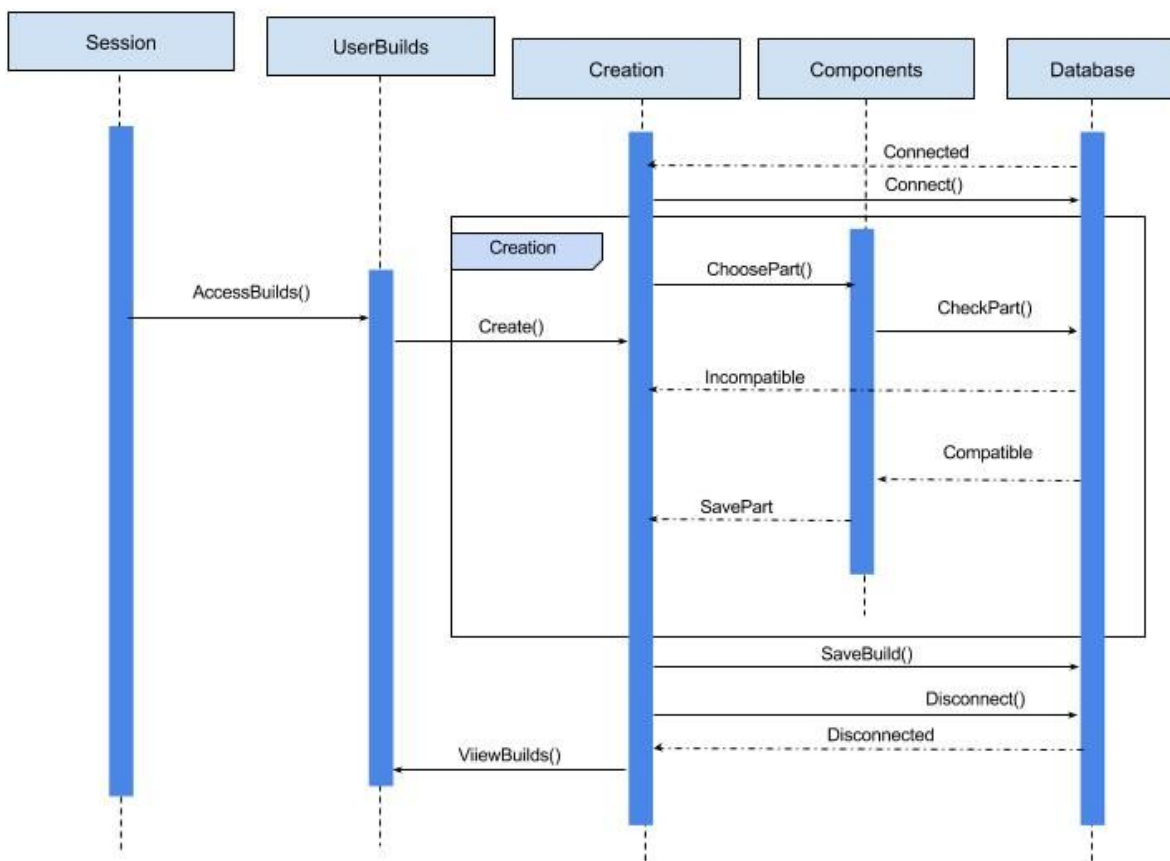


### Sequence Diagram 3: Create Build

To create a build user needs to pick several parts which *must* be compatible with each other, when a part is selected the system connects to database, the chosen part is 'checked' with the other selected parts to check compatibility.

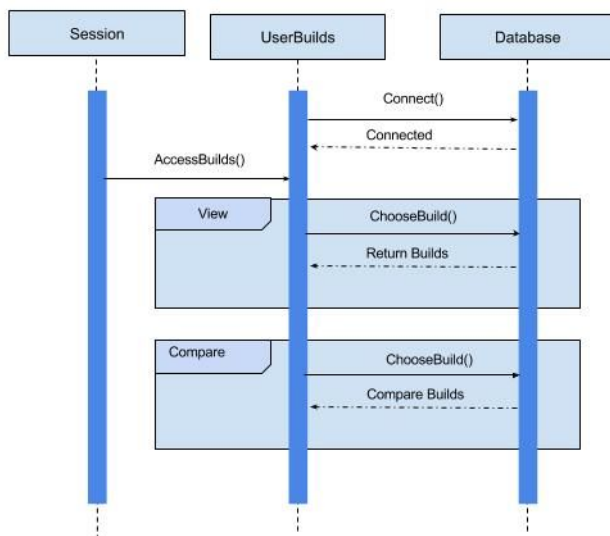
- if it is compatible it is saved and user is able to choose more parts
- if the part is incompatible user needs to chose a different part

When all parts are selected and they are all compatible with each other the build is saved in user's account .



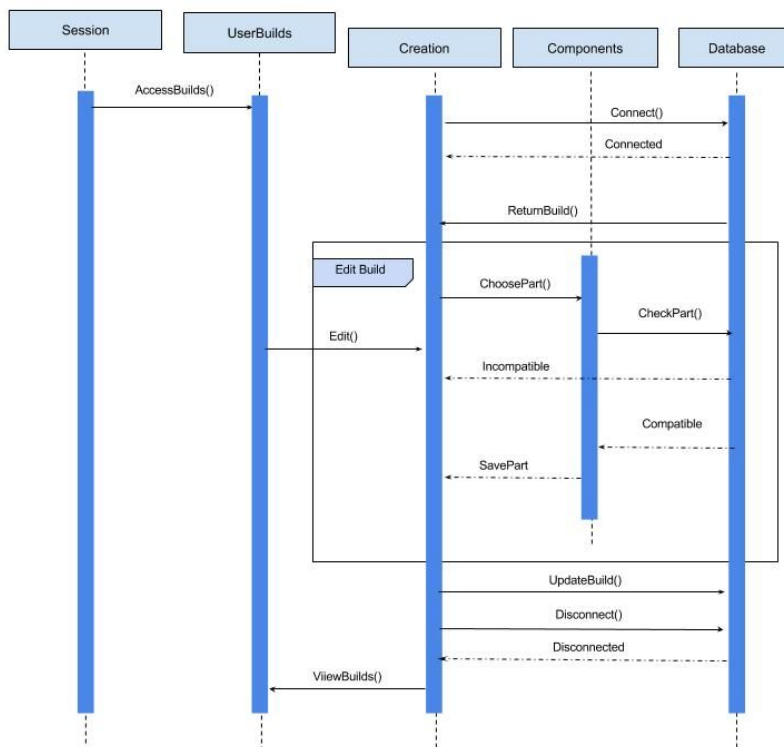


#### Sequence Diagram 4: View/Compare Builds



Once logged in user can view their old builds and compare them together, by connecting to the database the builds are loaded to be seen or to be compared with one another.

#### Sequence Diagram 5: Edit Build

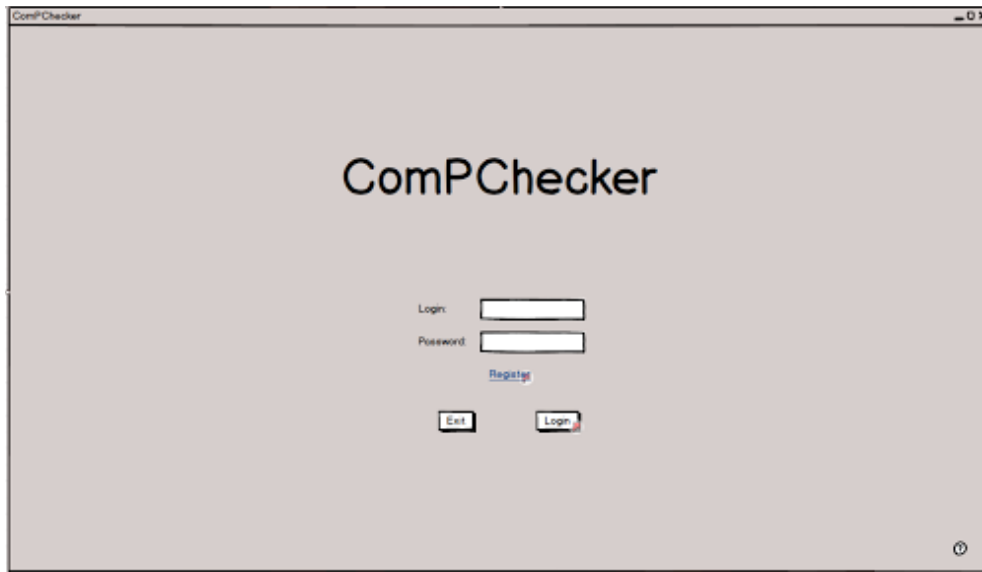


A user is able to edit a build by selecting an old part and choosing a new part; this change must be compatible with the rest of the build.



## UI Mock-ups

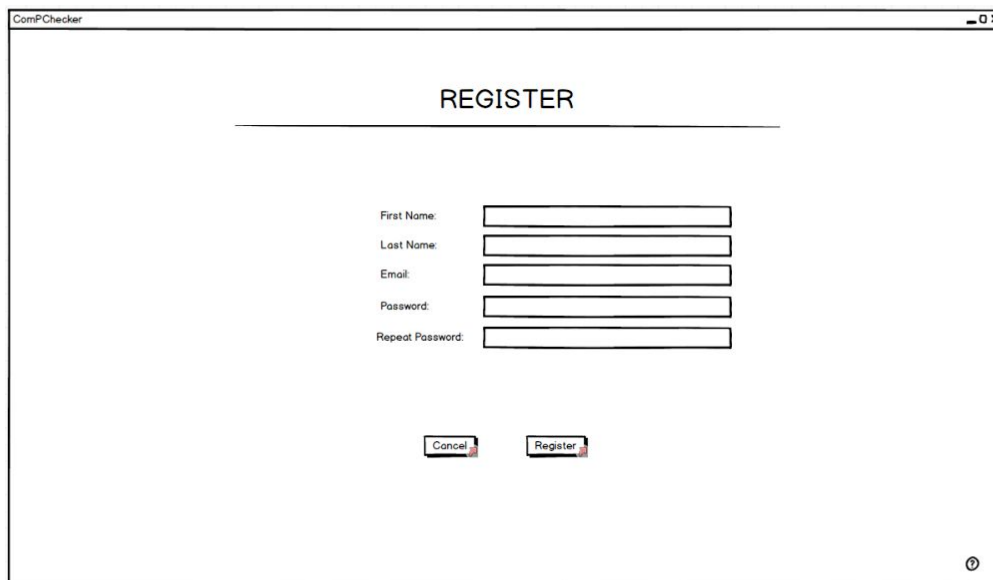
### *Mock-Up 1: Welcome Screen*

A screenshot of a web application window titled "ComPChecker". The window has a light gray background. In the center, the text "ComPChecker" is displayed in a large, bold, black font. Below the title, there are two input fields: "Login:" and "Password:". Below the "Password:" field, there is a blue link labeled "Register". At the bottom, there are two buttons: "Exit" and "Login". A small help icon (a question mark inside a circle) is located in the bottom right corner of the window.

This interface mockup shows the initial login screen. As the screen shows the UI is very simple and contains elements which users expect to see (exit, login field and password). The help button has also been included, this was highlighted by our users has as a core feature. There is also a link that the user can click to register/create a new account.

There is room here for us to customise and 'do-up' the screen a bit more, however as this is the initial login screen it needs to be clear and not cluttered so as to not confuse the user.

### *Mock-Up 2: Register Screen*

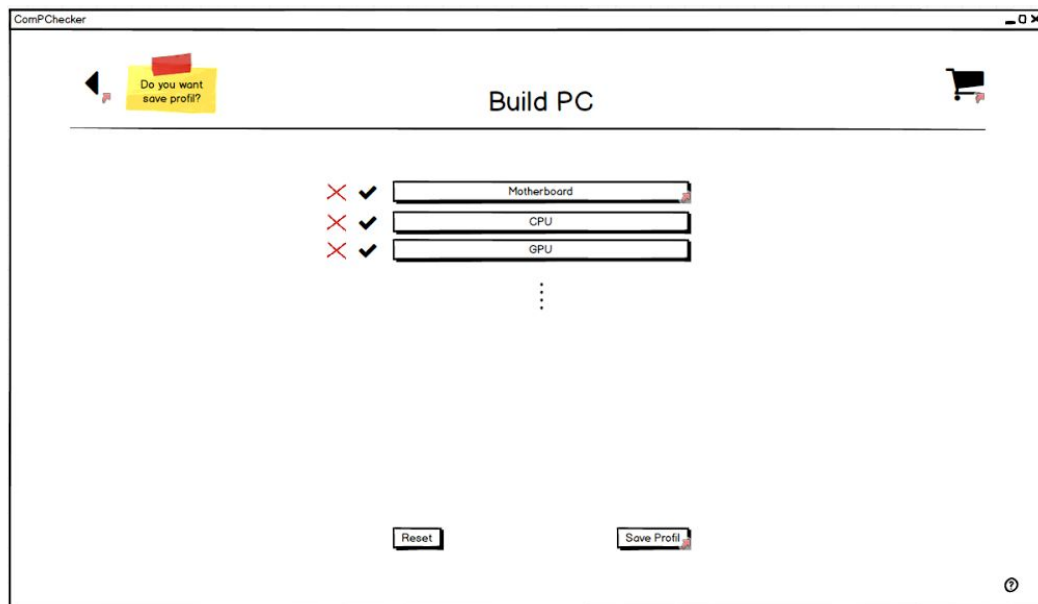
A screenshot of a web application window titled "ComPChecker". The window has a light gray background. In the center, the text "REGISTER" is displayed in a bold, black font, underlined. Below the title, there are five input fields: "First Name:", "Last Name:", "Email:", "Password:", and "Repeat Password:". At the bottom, there are two buttons: "Cancel" and "Register". A small help icon (a question mark inside a circle) is located in the bottom right corner of the window.

This interface is accessed through the login screen for those users who do not have an account. It is only used to create general users, admin users will be added to the system via the database (as there will be few admin users compared to general users).



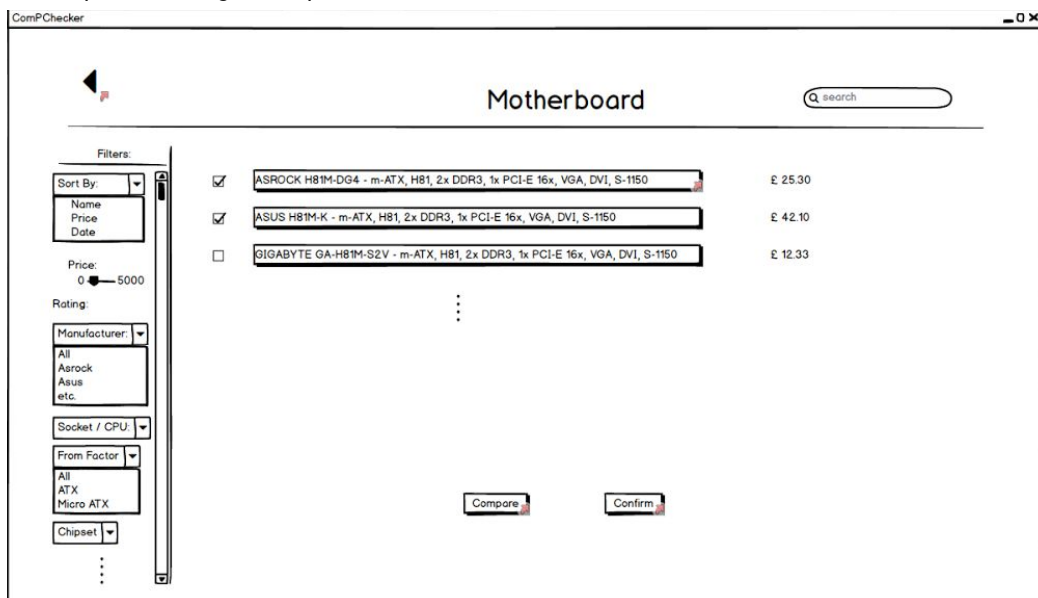


### Mock-Up 3: Building a PC



This is the main function screen of building a PC. Here the user has to click on each part type, in sequential order, once clicked on a part type it will direct the user to selecting a component screen (mockup 4).

### Mock-Up 4: Selecting a Component













Here users can select a part to add to their build. They are able to sort and filter parts by the panel on the lefthand side. Some detail is displayed here but more information can be seen by further inspecting the component or by comparing to another component.



### Mock-Up 5: Comparing Components

ComPChecker

COMPARISON

	ASROCK H81M-DG4	MOTHERBOARDS	ASUS H81M-K
Manufacturer:			
Form Factor:			
CPU Socket:			
Chipset:			
...			
			

Up to 5 components

This mockup shows the comparing components feature. It allows users to choose 2-5 parts (of the same type) and compare them with each other. They can add their preferred part to their build from this screen.



## Appendix

### Appendix 1: System Behaviour State Model for the function 'Build PC'

On this diagram the pump P2 represents the parts database.

