PC Building App: Senior Capstone Project

Keith Holcomb

CISS:451 Senior Projects 001

Spring 2025

Table of Contents

Introduction

Goals & Purpose

Computers have become fundamental tools in modern society, impacting global industries to personal communication to daily routines. They have become widespread and are essential in today's technological environment. The process of building a computer, however, can be seen as a very complex task. There are a variety of parts that go into a single computer, such as motherboards, memory and graphic cards. Understanding each of these components and how they interact with each other adds another layer of complexity to building a computer from parts.

The PC Building App is an application project that attempts to reduce the complexity and confusion of building a computer from parts. This project is a hypothetical business solution in that it introduces a wide audience of users to specific computer parts, which the user could then seek to purchase.

Significance

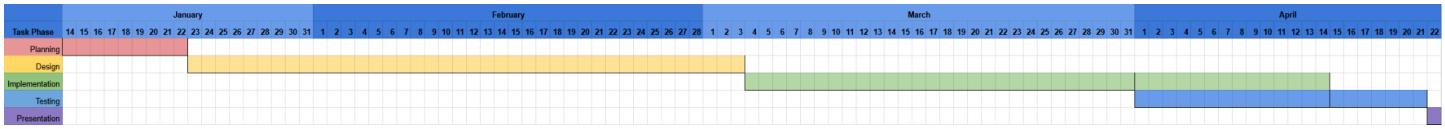
This project has significance in a few different ways. One main point of significance for this project is that it attempts to solve the complexity of building a personal computer or PC. Another major point of significance comes from my background of building computers for others. I have built my own computer, as well as several others for friends. One of the main issues I have noticed when someone comes to me about computer building is that some don't know where to start, what parts they need, or where they can get those parts. The PC Building App is something I created as a tool or guide for someone interested in having a computer built; I want this project to be something that people can use to get started with building a computer.

The PC Building App attempts to solve the problem of confusion/complexity in computer building by introducing a list making system, as well as a way to view other user created lists.

These lists are referred to as ‘PC Build Lists’ within the application, and are filled with slots for every required computer part that a single computer would require to function. A user can create any number of these PC Build Lists and add computer parts to them directly within the application. These computer parts are contained within a separate table that the user can access to add to ‘PC Build Lists’ or compare with other parts.

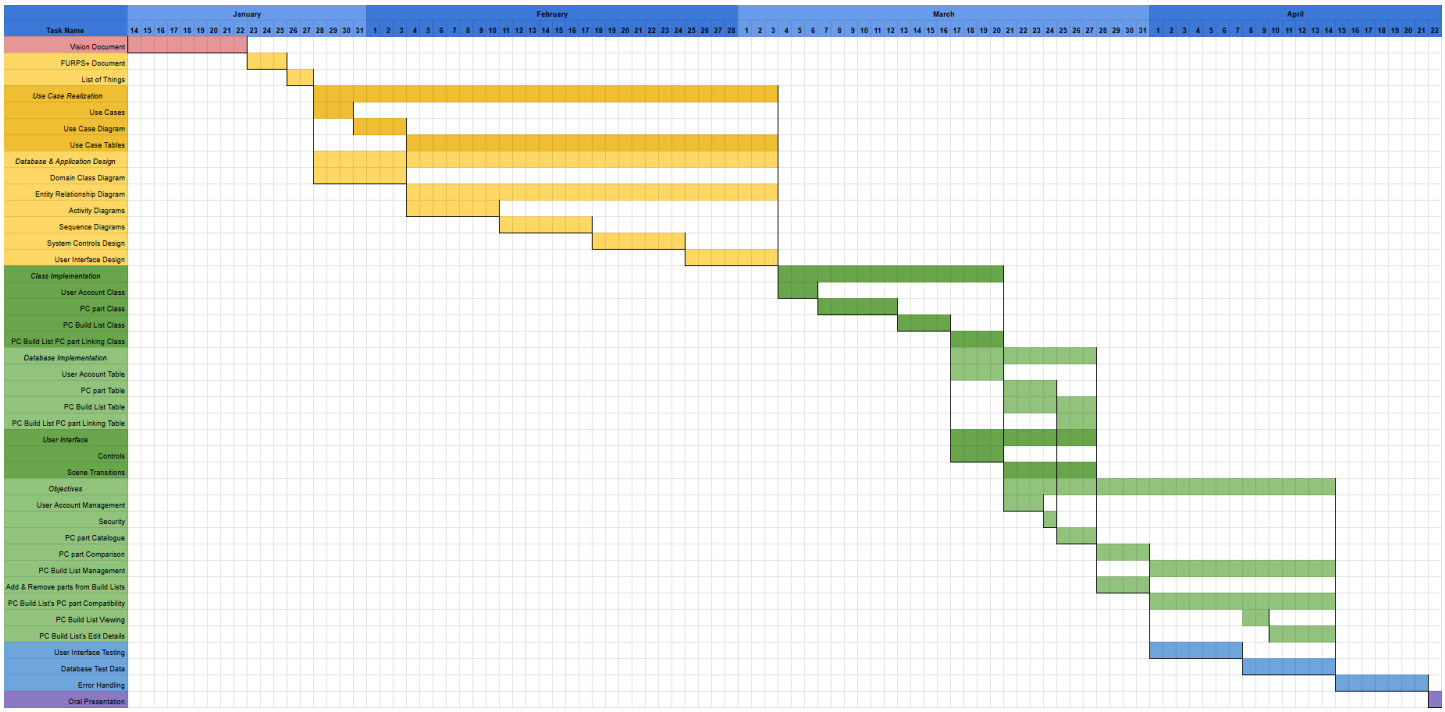
Project Timeline

In this section, the project timeline is presented using Gantt diagrams that outline the critical path of the PC Building App’s creation. The Gantt diagrams presented will show the timeline overview of core project phases, as well as a detailed view of timeline project phases. First, the PC Building App’s timeline of core project phases using a Gantt chart:



This diagram highlights the five main phases of the PC Building Apps’s creation, including the planning, design, implementation, testing, and presentation phases.

Next, is the PC Building App's detailed timeline using a Gantt chart:



This diagram highlights each of the fundamental phases of the PC Building App in further detail, with each core phase being color-coded. The color-coding is red/pink for planning, yellow/gold for design, green for implementation, blue for testing, and purple for presentation. Each of these phases is shown in more detail, which will be briefly highlighted. The planning phase consists entirely of the Vision document, which is included in Appendix A. The design phase consists of the FURPS document (Appendix B), Use Case Analysis diagrams (Appendix C), domain class & entity-relationship diagrams (Appendix D), activity diagrams (Appendix E), and sequence diagrams with detailed design (Appendix F), and user interface design (Appendix G). The implementation phase consists of class implementation, database implementation, user interface implementation and objectives. The testing phase consists of testing user flow, database test data, as well as ensuring error handling works as intended. The presentation phase consists entirely of the oral presentation of the PC Building App.

Objectives

To address the problem and its significance, the PC Building App attempts to achieve the following objectives:

1. PC Build List creation and management.
2. Adding and removal of PC parts to PC Build Lists through the PC part catalogue which can be searched through and filtered.
3. PC part compatibility checks within a PC Build List.
4. User account creation, login and management.
5. PC Build List publishing and viewing access.

Narrative for Analysis & Design

Analysis & Design

The analysis and design of the PC Building App project had several phases/parts. These included the planning, PC Building App's requirements analysis, and detailed design. Some of these phases were conducted one after the other, while some were done simultaneously.

First, there is the planning phase of the PC Building App. The planning phase was the first step in the analysis of the PC Building App. In this phase, the problem of computer building complexity was first defined, and an application solution was proposed. At this phase of analysis, some defining features of the application were planned, such as the system capabilities and the potential business benefits of creating such an application.

The planning phase was done by creating a vision document, which encapsulated the problem description, system capabilities and potential business benefits of the PC Building App project (see Appendix A).

Next, there is the PC Building App's requirements analysis phase of the PC Building App. System requirements analysis was the second step in the analysis of the project. In this phase, the functional and non-functional requirements of the application were defined using a functional-usability-reliability-performance-supportability (FURPS) document (see Appendix B). These functional requirements would later become the objectives for the PC Building App project.

Third, is the detailed design phase of the PC Building App, which began alongside the PC Building App's requirements analysis phase. The detailed design phase is the third phase of analysis and design for this project, but is also one of the longest. This phase consisted of the majority of the project's design, including the applications inputs and outputs, database,

processes, controls and the user interface. The detailed design for the PC Building App project was created using several diagrams and documents. For use case analysis, a use case diagram, as well as a use case table with detailed descriptions was created (see Appendix C).

System Requirements

The initial functional requirements of the PC Building App included PC part selection, PC build planning, and PC build accounts/guides. The initial non-functional requirements for the PC Building App included the usability, reliability, performance, security, design constraints, implementation, interface and supportability requirements. These non-functional requirements of the PC Building App are covered in the FURPS document in Appendix B.

For the functional requirement of the PC Building App's PC part selection, it can be divided into three parts. The first part is the table of PC parts. The second part is the PC part search filtering, as well as sorting. The third part of the PC part selection requirement is the literal selection and comparison of PC parts. These functional requirements would later become the objectives for the PC Building App project. The objectives for the PC Building App are covered in more detail in the 'Objectives' section of the report.

Input, Output, Database & System Processes

This section details the design and implementation of the application's data handling, system architecture, and process flow. It describes how the application manages data, interacts with the database, and executes the necessary logic to support the user interface and functionality.

For database tables and application processes, a table was created to list the various things of the application. This list of things was then used to create a domain class diagram, and

later an entity-relationship diagram (see Appendix D). These diagrams define the structure of the data and the relationships between different entities within the application.

For the design level of inputs and outputs in the application, activity diagrams were created. These activity diagrams detailed several what-if scenarios that would specify the flow between user interactions and the application (See Appendix E). Detailed sequence diagrams would then be designed to show how these activities would actually be implemented into the application, as well as specify how the user, the application and the database would all interact with each other in these processes (See Appendix F). This illustrates the flow of data and control within the system, showing how different components interact to process user requests and update the database.

The system capabilities of the PC Building App project as an application would be a JAVA programming language based program that would make use of a SQL-based database to read and write information necessary for the application to run. In the original envisioning phase, the SQL-based database to be used was MySQL, but this would later change to another SQL-based database, called PostgreSQL.

Implementation of the PC Building App was done in parts, the first being to establish the classes and their associated database tables. The classes that were created were the ‘UserAccount’, ‘PCpart’, ‘PCBuildList’, and ‘PCBuildListPCpart’ classes. The four classes were associated directly with four different database tables, ‘useraccount’, ‘pcpart’, ‘pcbuildlist’, and ‘pcbuildlist_pcpart’. These classes and tables were created based on the list of things as well as the domain class diagram and entity-relationship diagram shown in Appendix D. Each of the classes, as well as each of the associated tables hold important data that the PC Building App uses to function.

The PC Building App makes use of several classes to perform its processes. Some of these classes are directly related to tables within the database. These classes are used to represent portions of the database tables to perform logical computations within the PC Building App during run-time. Each of the following classes were designed using the list of things, as well as the domain class diagram included in Appendix D.

The first class is the ‘UserAccount’ class, which has two data fields, an instance variable, and a boolean status. The first data field is the ‘username’ field. This field is meant to hold user input to be used during user account management. The second data field is the ‘accountid’ field. This field is meant to hold the associated ‘accountid’ table column of a ‘useraccount’ entry. The ‘UserAccount’ class makes use of an instance variable, which the PC Building App accesses to determine whether the user is currently logged into a user account or not. The ‘UserAccount’ class instance is used alongside the boolean status, ‘isLoggedIn’ for the PC Building App to determine whether the user is currently logged into an account or not, as well as run other logic.

The second class is the ‘PCpart’ class, which has seven data fields and a boolean status. The seven data fields are ‘partID’, ‘partName’, ‘partType’, ‘manufacturer’, ‘model’, ‘price’ and ‘specifications’. Each of these data fields represent values retrieved from the database table, ‘pcpart’. The boolean status is a BooleanProperty that determines whether a PC part within the PC part catalogue interface has its CheckBox column selected.

The third class is the ‘PCBuildList’ class, which has twelve data fields. The twelve data fields are ‘pcBuildListId’, ‘accountId’, ‘pcBuildListName’, ‘description’, ‘creationDate’, ‘modifiedDate’, ‘totalPrice’, ‘listType’, ‘content’, ‘category’, ‘thumbnailURL’, and ‘publish’. Each of these data fields represent values written and retrieved from the database table,

‘pcbuildlist’. This class is used by the PC Building App to perform logic related to the specific data to be written or retrieved from the database.

The fourth class is the ‘PCBuildListPCpart’ class, which has four data fields. The four data fields are ‘listPartId’, ‘listId’, ‘partId’, and ‘quantity’. Each of these data fields are associated with values retrieved from the database table, ‘pcbuildlist_pcpart’. This class is used by the PC Building App to perform logic related to the specific data to be retrieved from the database.

The PC Building App interacts with the database using SQL queries, which are generated based on user actions and the PC Building App’s programming logic. These queries interact with each of the database’s tables based on their column data. Each of these tables has several important columns, each of which are important for the processes of the PC Building App. These tables were designed using the list of things, as well as the entity-relationship diagram shown in Appendix D.

The first table is the ‘useraccount’ table, which has three different columns; The first being an ‘accountid’ column. This column is unique in that it iterates a serial number whenever a new entry to the table is added. The ‘username’ column is the second column in the ‘useraccount’ table. The PC Building App uses this column for user account management, having the user input a unique value for every user account entry. Lastly, there is a ‘password’ column. The ‘password’ column holds an encrypted string up to 128 characters long. The ‘password’ column holds an encrypted value to ensure the security of user-provided passwords. The encryption and decryption of the ‘password’ column of the ‘useraccount’ table is managed within the PC Building App itself.

The second table in the database is the ‘pcpart’ table. This table has 7 different columns, the first of which is the ‘partid’ column. Similar to the ‘useraccount’ table, the first column of this table iterates a serial number whenever a new entry is added to the table. The ‘partname’ column is the second column of the ‘pcpart’ table. This column represents the name of the computer part associated with the table entries, and similarly to the ‘username’ column of the ‘useraccount’ table, it also has to be unique. The third column is the ‘parttype’ column. This column is responsible for identifying what part of a computer a certain computer part is. This column is particularly important, as it determines what portion of a PC Build List a ‘pcpart’ entry belongs to. The possible entries for the ‘parttype’ of the ‘pcpart’ table are ‘CPU’, ‘Motherboard’, ‘CPU Cooler’, ‘RAM’, ‘Storage’, ‘GPU’, ‘Power Supply’, ‘Case’, and ‘Case Fan’. The fourth column of the ‘pcpart’ table is the ‘manufacturer’ column. This column represents what brand or company a computer part entry belongs to. The fifth column is the ‘model’ column. This column represents the exact model name given to the computer part by its manufacturer. The sixth column is the ‘price’ column. This column holds the USD price of a given computer part entry. This price would vary based on the vendor the computer part is sold from. The seventh column is the ‘specifications’ column. This column is extremely important, as it holds the data necessary for computer parts to be compared to each other, as well as holds the data needed to check the computer parts compatibility with other parts. This column is lengthy for every ‘pcpart’ entry, and is stored as a JSONB. The PC Building App is responsible for converting the JSONB into a usable string or list. The PC Building App reads the JSONB to process comparing parts as well as process compatibility checks.

The third table is the ‘pcbuildlist’ database table. The ‘pcbuildlist’ table has twelve columns for data entry. The first column is the ‘pcbuildlistid’, and similar to the previous tables,

is also a serial number that iterates when new entries are added. The second column is an ‘accountid’ column. This column is a reference to a specific entry of the ‘useraccount’ table and represents what user created the ‘pcbuildlist’ entry. The third column is the ‘pcbuildlistname’. This column, similar to other name-based columns, is required to be unique by the PC Building App. The fourth column is the ‘description’ column. The ‘description’ column is used by the PC Building App to allow the user to specify any details of a PC Build List they wish to share. The fifth column for the ‘pcbuildlsit’ table is the ‘creationdate’ column. This column holds the date of the day the ‘pcbuildlist’ entry was created and is not meant to change. The sixth column is the ‘modifieddate’ column and holds the date of the latest day the ‘pcbuildlist’ entry was edited. The seventh column is the ‘totalprice’ column. This column is set by the PC Building App and holds a numeric value equal to all of the ‘price’ columns of ‘pcpart’ entries that are to be associated with the ‘pcbuildlist’ list. The eighth column is the ‘listtype’ column. This column has two possible values of ‘Normal’ and ‘Guide’. The PC Building App uses this column to determine whether a ‘pcbuildlist’ entry should use the remaining columns. If a PC Build List has a ‘listtype’ of type ‘Normal’, the remaining columns are to be left empty. If a PC Build List has a ‘listtype’ of type ‘Guide’, then the remaining four columns are used within the PC Building App. The ninth column is the ‘content’ column. This column is used in the PC Building App when a ‘pcbuildlist’ is of type ‘Guide’. The tenth column is the ‘category’ column. This column is used in the PC Building App when a ‘pcbuildlist’ is of type ‘Guide’. Both the ninth and tenth column are meant to be filters for published PC Build Lists to help users viewing published PC Build Lists to filter between entries. The eleventh column is the ‘thumbnailurl’ and is used in the PC Building App to display an image when the ‘pcbuildist’ is of ‘listtype’ ‘Guide’ and is published. The twelfth column is the ‘publish’ column. This column is a boolean column that can either be

true or false. The PC Building App uses this column to determine if a ‘pcbuildlist’ with the ‘listtype’ of type ‘Guide’ should be visible when viewing published PC Build Lists.

The fourth table of the database is the ‘pcbuildlist_pcpart’ table. This table is responsible for connecting a ‘pcbuildlist’ table entry with any number of different ‘pcpart’ table entries. The ‘pcbuildlist_pcpart’ table has four columns. The first is a serial value that iterates for every entry of the table, called ‘listpartid’. The second column is a ‘listid’ column. This column identifies a single ‘pcbuildlist’ entry’s ‘listid’ value. The third column is a ‘partid’ column. This column identifies a single ‘pcpart’ entry to attach to a ‘pcbuildlist’. The last column is the quantity column. This column has a default value of 1, and represents how many of a single ‘pcpart’ entry are present in one ‘pcbuildlist’ entry. The PC Building App uses this table to identify what ‘pcpart’ entries should be loaded into a PC Build List.

The PC Building App uses these database tables and classes to implement several key processes to achieve the project’s objectives:

- User Account Management: The PC Building App processes user registration, login, and logout requests. This involves validating user input, interacting with the 'useraccount' table in the database, and managing user sessions.
- PC Part Catalogue Management: The PC Building App retrieves and displays PC part information from the 'pcpart' table. It also supports filtering, sorting, and comparison of PC parts.
- PC Build List Management: The PC Building App allows users to create, view, edit, and delete PC build lists. This involves managing data in the 'pcbuildlist' and 'pcbuildlist_pcpart' tables, including adding, removing, and checking compatibility of PC parts.

- PC Part Compatibility Checking: The PC Building App implements a series of rules to ensure that the selected PC parts in a build list are compatible with each other. This process involves retrieving PC part specifications from the database and comparing them based on predefined criteria. Any compatibility issues are then displayed to the user.

Application Controls

The PC Building App uses JavaFX, a JAVA-based client application platform, to implement controls used in the application. Several of the controls used within the application were designed alongside the user interfaces, in the form of JavaFX scenes (See Appendix G). The other portion of controls used within the PC Building App were created when implementing the objectives of the project. All controls referenced in this section are from the JavaFX 24 *javafx.scene.control* package (*javafx.scene.control* 2024).

The following are all of the controls that were introduced when designing the user interfaces of the PC Building App, what they are, and how they were used in the project.

- AnchorPane. A layout pane that allows nodes to be anchored to the top, bottom, left, or right edges (*javafx.scene.layout.AnchorPane* 2024). AnchorPane controls were used in the PC Building App to anchor content to different sides of some user interfaces, as well as anchor content to the center of user interfaces.
- BorderPane. A layout pane with top, bottom, left, right, and center regions (*javafx.scene.layout.BorderPane*). A BorderPane control was used as the parent, or main control of every user interface. Everything that was visible within the application was wrapped inside of a single parent BorderPane control.

- Button. A standard button control (*javafx.scene.control.Button* 2024). Button controls were used within the PC Building App to transition between different user interfaces, as well as submission of user input.
- ButtonBar. A pane that lays out buttons in a platform-appropriate manner (*javafx.scene.control.ButtonBar* 2024). A ButtonBar control was used in several user interfaces to manage a set of Button controls related to user account management. The ButtonBar control was also used as a visual header within the PC Building App's user interfaces.
- ColumnConstraints. Defines constraints for columns within a GridPane layout (*javafx.scene.layout.ColumnConstraints* 2024). ColumnConstraints controls were used to adjust the size of published PC Build Lists in the viewing user interface. This user interface uses a GridPane control to show all published PC Build Lists within the database.
- ComboBox. A selection control that displays a drop-down list of options (*javafx.scene.control.ComboBox* 2024). ComboBox controls were used to show a dropdown selection to the user of all of the PC Build Lists they owned. Selecting a PC Build List from the ComboBox control allowed the user to load the PC Build List as well as add PC parts to the PC Build List.
- GridPane. A layout pane that arranges nodes in a grid of rows and columns (*javafx.scene.layout.GridPane* 2024). A GridPane control is used within the PC Building App to display published PC Build Lists (or guides) to the user.

- ImageView. A node that displays an Image (*javafx.scene.control.ImageView* 2024). ImageView controls are used within the PC Building App to display various images, such as the application icon, as well as PC Build List thumbnails.
- Label. A non-editable text control (*javafx.scene.control.Label* 2024). Label controls were used to display text to the user within the PC Building App's user interfaces. This text would be titles, instructions or information on how certain controls within the application worked.
- PasswordField. A text field control that obscures entered text, used for secure input (*javafx.scene.control.PasswordField* 2024). PasswordField controls were used in the PC Building App for the user to securely enter passwords and password confirmations during user account management.
- RadioButton. A selectable button that is part of a toggle group, allowing only one selection within the group (*javafx.scene.control.RadioButton* 2024). RadioButton controls were used in the PC Building App to accept specific user input. Some of the specific user input decided via RadioButton controls includes whether a PC Build List is of type 'Normal' or type 'Guide', and whether a PC Build List of type 'guide' was published or not.
- ScrollPane. A container that provides scrollbars for viewing content that exceeds the available display area (*javafx.scene.control.ScrollPane* 2024). ScrollPane controls were used within the PC Building App to hold database related data. One of the most prevalent uses of a ScrollPane control is to hold the PC part catalogue, which is a rather large table of data. This catalogue was stored within a TableView control, and this TableView control was stored within the ScrollPane control.

- TextField. A text input control that allows the user to enter a single line of text (*javafx.scene.control.TextField* 2024). TextField controls were used within the PC Building App to gather data during user account management, as well as PC Build List creation.

The following are all of the controls that were introduced when implementing the objectives of the PC Building App, as well as what they are, and how they were used in the project.

- Alert. A dialog box used to display notifications, warnings, or errors to the user (*javafx.scene.control.Alert* 2024). An Alert control was used within the PC Building App for critical error handling. The errors Alert controls were used were critical errors that would never occur under normal user flow, and would be used to resolve these errors.
- CheckBox. A selectable control that allows the user to toggle an option on or off (*javafx.scene.control.CheckBox* 2024). CheckBox controls were used within the PC Building App to add functionality to the PC part catalogue, in which a CheckBox control was added to every single PC part catalogue entry. These CheckBox controls could be selected by the user to either add the PC parts to a PC Build List or to compare the PC parts specifications with one another.
- TableView. A control that displays data in a tabular format, with columns and rows (*javafx.scene.control.TableView* 2024). TableView controls were used within the PC Building App to hold database table data, such as PC parts.
- TableColumn. Represents a column within a TableView (*javafx.scene.control.TableColumn* 2024). TableColumn controls were used within the PC Building App to hold the columns of database tables, such as PC part columns.

- TableRow. Represents a row within a TableView (*javafx.scene.control.TableRow* 2024). TableRow controls were used within the PC Building App to specify specific entries within a database table, such as a single PC part entry.

User Interface

This section describes the design and functionality of the PC Building App's user interface, focusing on the specific controls and their role in facilitating user interaction.

The user interface designs were detailed to show various controls and ensure proper user-application flow (See Appendix G). Alongside the user interface design portion of the detailed design phase, the implementation phase of the PC Building App would also begin.

The application's user interface is composed of several distinct screens or views, each designed to provide specific functionality:

The initial user interface gives the user three buttons, all related to user account management. One of the buttons is to switch to creating a user account within the database. Another button is to log in to an existing user account within the database. The third button is to skip logging in to a user account and continue through the PC Building App as a guest for now. If the user chooses to create an account or log in, they are taken to the associated login or create account user interfaces, and afterward are taken to the main menu user interface. If the user chooses to continue as a guest, they are immediately taken to the main menu user interface.

- *Controls:* Three Button controls (Create Account, Login, Continue As Guest).

The user account creation interface presents the user with a TextField control and two PasswordField controls. The user is to enter their desired username, and a password twice. If the username isn't taken, and the two password entries match, then the PC Building App adds the

user account entry to the database, and logs the user into it. The user is then taken to the main menu interface.

- *Controls:* One TextField control (Username), two PasswordField controls (Password, Password Confirmation).

The login interface presents the user with a TextField control and a PasswordField control. The user is to enter their username and associated password. The PC Building App will check with the database to confirm there is a username matching the user's entry, and then check if the password matches the one stored within the database. If successful, the user is logged in and taken to the main menu interface.

- *Controls:* One TextField control (Username), one PasswordField control (Password).

The main menu interface has many controls, including a ButtonBar control, and several Button controls. The ButtonBar holds three Button controls, all of which are for user account management, such as creating a user account, logging into a user account, or logging out of the current user account. This same ButtonBar is used in many user interfaces within the PC Building App. The main menu interface also has four other Button controls as well as a ComboBox control. One button switches the user interface to the published PC Build List view interface. Another button switches the user interface to the PC part catalog interface. A third button switches the user interface to a PC Build List creation interface. The last button loads a PC Build List that the user selects using the ComboBox control which has a dropdown view of all the PC Build Lists the user owns. This ComboBox control is only filled with entries when the user is logged in and has PC Build Lists associated with their account. This ComboBox control is also reused in other user interfaces whenever the user has to select from PC Build Lists they own.

- *Controls:* ButtonBar containing three Button controls (Create Account, Login, Logout), four separate Button controls (View PC Build Guides, View PC Parts, Create a PC Build List, Load a PC Build List), one ComboBox control (PC Build List selection).

The published PC Build List view uses a GridPane control to show any published PC Build Lists available to view. Each published PC Build List entry also uses an ImageView control to show the thumbnail URL's image, or a default image if it cannot be loaded. Each of these published PC Build Lists are held within ColumnConstraints controls that limit their view size to being half of the GridPane control's width. This interface also uses the ButtonBar from the main menu interface, as well as an additional button to return to the main menu interface.

- *Controls:* GridPane control (PC Build Guides), ImageView controls (For Thumbnail Images), ButtonBar containing three Button controls (Create Account, Login, Logout), one Button control (Main Menu).

The PC part catalog interface uses a ScrollPane to hold the TableView the PC Building App creates to show PC part entries from the database table. In this TableView, the PC Building App adds several columns for each table entry, including one that adds a CheckBox control for every table entry. This CheckBox control is used to either compare selected PC parts or add selected PC parts to a PC Build List. This interface reuses the ButtonBar from the main menu for user account management, and also reuses the ComboBox control from the main menu interface. The ComboBox control holds the user's owned PC Build Lists and is used with two buttons in the PC part catalog interface. One of the buttons is to load the PC Build List selected from the ComboBox control. The other button adds any PC part entries that have their CheckBox control selected to the PC Build List selected in the ComboBox control. There is also one more button within the PC part catalog interface to return to the main menu interface. There is a fourth Button

control that uses the CheckBox controls selected within the TableView control. This Button control is to compare any selected PC parts with each other in the PC part comparison interface.

- *Controls:* ScrollPane (Holds TableView), TableView (PC Part Entries), CheckBox controls (Select PC Parts), ButtonBar containing three Button controls (Create Account, Login, Logout), ComboBox (PC Build List selection), four Button controls (Load PC Build List, Add to Build List, Main Menu, Compare Parts).

The PC part comparison interface takes any selected PC parts from the PC part catalog interface and displays them side-by-side along with their specifications. This display makes use of several Label controls that each display a portion of the PC part's information. The only other controls present in this user interface are the ButtonBar control from the main menu interface, which contains the three Button controls related to user account management discussed within the main menu interface.

- *Controls:* Label controls (PC Part Information), ButtonBar containing three Button controls (Create Account, Login, Logout).

The PC Build List load interface takes a given PC Build List, loads it from the database table and displays it, any associated 'pcbuildlist_pcpart' table entries, and several controls to manage the PC Build List. This interface reuses the ButtonBar control which holds three Button controls related to user account management, as discussed in the main menu interface. The PC Build List database table is loaded into a ScrollPane control that uses several Label controls and Button controls to display the PC Build List data. This ScrollPane is filled with Label controls that show the PC Build List details, total price, and any associated PC parts, split between the PC part types. This ScrollPane is also filled with a Button control to change the interface to the PC Build List edit interface. There is also a Button control to refresh the total price displayed within

the ScrollPane. There are also Button controls for any empty PC part type. These Button controls will direct the user to the PC part catalog and filter the catalog to only display the PC part type associated with the Button control pressed in the PC Build List load interface. There are also Button controls for any PC part entries to remove the PC part entries and reload the PC Build List load interface. This interface also reuses the ComboBox control from the main menu interface to select a PC Build List the user owns for a Button control to load into the PC Build List load interface. This ComboBox is also used for another Button control which deletes the PC Build List selected within the ComboBox control. Lastly, there is also a Button control to return the user to the main menu interface.

- *Controls:* ButtonBar containing three Button controls (Create Account, Login, Logout), ScrollPane (PC Build List Data), Label controls (PC Build List Details, Total Price, PC Parts), Button controls (Edit PC Build List, Refresh, Add PC Part, Remove PC Part, Main Menu, Delete PC Build List), and a ComboBox control (PC Build List selection).

The PC Build List edit interface takes a given PC Build List, and loads its details into several TextField controls and RadioButton controls. These TextField controls are editable by the user to change details such as the PC Build List's name, description, and type. RadioButton controls determine whether the PC Build List is of type 'Normal' or type 'Guide,' and if the PC Build List is of type 'Guide,' there are more TextFields available to the user to edit the PC Build List's content, category, and thumbnail URL. If the PC Build List is of type 'Guide,' there are also two RadioButton controls to determine whether the PC Build List is published or not. Once the user fills in any necessary changes, they can press a Button control which will have the PC Building App write any changes to the PC Build List database table entry. There is also a Button control to cancel performing any change to the PC Build List. If either of these Button controls

are pressed, the user is returned to the PC Build List load interface with the PC Build List. The PC Build List edit interface also features the ButtonBar control from the main menu interface, as well as its three Button controls for user account management.

- *Controls:* TextField controls (PC Build List Name, Description, Content, Category, Thumbnail URL), RadioButton controls (List/Guide, Published/Not Published), Button controls (Submit, Cancel), ButtonBar containing three Button controls (Create Account, Login, Logout).

Testing Procedures

Testing was done for the design and creation of the PC Building App. The tests conducted for the PC Building App include testing application-level functionality, application-to-database data entry, database data retrieval, and user flow testing.

First, the PC Building App was tested for application level functionality, such as the application start, as well as user interface transitioning. These tests were conducted by running the program and attempting to ‘mock’ user flow. An example of this is a test where logging in to an account was tested. This test would run the program right to the login interface. User input would be collected using a TextField control and a PasswordField control. If the values entered in these fields matched the test-values, then the program would print to the console that the user ‘logged in successfully’.

Second, the PC Building App was tested for application-to-database data entry. These tests were conducted to ensure that writing ‘useraccount’ table entries, as well as ‘pcbuildist’ table entries worked as expected. An example of one of these tests would run the program right to the create account interface. The TextField control, and two PasswordField controls would be pre-filled with test-values, then the PC Building App would attempt to query to the ‘useraccount’

table a new entry. The database would then be observed to check the latest table entry to see if the ‘username’ column value matched the test-value used in the TextField control.

Third, the PC Building App was tested for database retrieval. These tests were conducted to ensure that all the tables, especially the ‘pcpart’ table would correctly display info within the PC Building App. An example of this test continues off the previous example test. After test-values were confirmed to be written to the database. Another test of the PC Building App would begin, where the login interface would have test-values prefilled into them as the create account interface did. The PC Building App would then print to the console whether data in the database matched the pre-filled values.

Lastly, the PC Building App was extensively tested for user flow. User flow testing was conducted to determine if all functionality of the PC Building App worked as expected. These tests would test starting the application from the beginning, logging in or creating an account, then navigating all available menus, such as creating a PC Build List, loading a PC Build List, viewing published PC Build Lists and viewing the PC part catalogue. These tests would be conducted the most throughout the PC Building App’s development.

Results

Goals & Program Output

The goals set for the PC Building App include PC Build List management with PC part compatibility checks, PC part catalogue searching/filtering, PC Build List ‘Guide’ publishing/viewing, and user account management. These goals align with the primary objectives of the PC Building App project and are met when the application and database are run with minimal issues.

The first goal of the PC Building App is PC Build List management with PC part compatibility checks. This goal encompasses the core functionality of a PC Build List, including PC Build List creation, deletion, as well as the addition/removal of PC parts and PC part compatibility checks within a PC Build List. This goal can be achieved within the project by creating a user account, then beginning the process of creating a PC Build List. After filling out the core details, the PC Building App will bring the user to the PC Build List load interface, it is here where the user can add and remove PC parts, as well as see any compatibility issues with the PC parts in their PC Build List. In this menu, the user is also able to load different PC Build Lists and even delete the PC Build List that is currently loaded.

The second goal of the PC Building App is the PC part catalogue with its search/filter features. This goal covers the PC part catalogue and its core functionality, which includes being searched through using keywords, filtered & sorted by columns, as well as used to add PC parts to PC Build Lists and used to compare the specifications of PC parts. This goal can be achieved by directly going to the PC part catalogue, which will fill the interface with all ‘pcpart’ database table entries. These entries can be selected using CheckBox columns to either be compared with each other or added to a PC Build List. In this interface, the user can also use the TextField

control called ‘searchBar’ to type keywords and filter the catalogue. When within a PC Build List, if the user clicks one of the Button controls to add a specific PC part type, the PC part catalogue is loaded and filtered to only show entries of that ‘parttype’ column.

The third goal of the PC Building App is the PC Build List’s ability to be published as a ‘listtype’ of ‘Guide’, and being able to view all published ‘Guide’ PC Build Lists. This goal covers the PC Build List being of two types as discussed previously, ‘Normal’ and ‘Guide’, as well as the type ‘Guide’ being able to be published for all users to see in the PC Build List view interface. This goal is achieved by any user, logged into an account or not, accessing the PC Build List view interface. This interface will show the details of a PC Build List, including its name, description, creation date, modified date, type (which is always ‘Guide’), content, category, and a visual image of the PC Build List’s thumbnail URL. Any PC Build List of type ‘Guide’ that has its ‘publish’ column as true can be seen within this view. For a user to achieve this, they must load a PC Build List and edit its details in the PC Build List edit interface. In this interface, the user must specify that the PC Build List is of type ‘Guide’ and set its published status to ‘true’ using the RadioButton controls provided in the edit interface.

The fourth goal of the PC Building App is user account management. This goal encompasses the user’s ability to create and login to a user account. This user account is used to manage content created by the user, which is any PC Build List the user creates. This goal can be achieved when the user starts the application and chooses to either create an account or login to an existing account. Either of these actions will take the user to an appropriate interface. The create account interface, as discussed previously, will have a TextField control for the username, and two PasswordField controls for the password and password confirmation. The login interface, also discussed previously, will have a TextField control for the username, and a

PasswordField control for the password. Once the user gets past either of these interfaces, their user account status is logged in. The user can then proceed to load any PC Build Lists they have created before, or create an entirely new PC Build List.

Exception Handling

Although the goals of the PC Building App are fairly straightforward, there are many possible errors that could occur when the project is run. These errors include errors in input validation, database connection, and PC Build List compatibility errors. Each of these errors can be identified by the user during run-time within a Label control named ‘errorLabel’ which is located towards the bottom of every user interface. This ‘errorLabel’ control details error messages in red to the user to let them know their action failed, and the reason why.

The first type of error that could occur is in input validation. Input validation errors occur within the PC Building App when the user’s input is incorrect. These errors can occur during account creation, login, PC Build List creation/editing, PC part comparison, as well as when the user tries to use the heading ButtonBar controls when they don’t need to.

The error messages that the user could see related to input validation in the PC Building App during account creation in the ‘errorLabel’ control are as follows:

- “Account creation failed. Username already exists.” Username has already been taken within the ‘useraccount’ database table.
- “Account creation failed. Username must be between 8 and 32 characters long.” Username was either too long or too short in length.
- “Password must be between 8 and 100 characters in length.” Password was either too long or too short in length.

- “Password must contain at least one uppercase, lowercase and digit character.” Password didn’t contain at least 1 uppercase, lowercase and numeric character.
- “Password and password confirmation mismatch.” PasswordField controls did not match.

The error messages that the user could see related to input validation in the PC Building App during login within the ‘errorLabel’ control are as follows:

- “Please enter your login credentials.” One or both of the field controls were left empty.
- “Login failed. Wrong username or password.” Occurs when either the username wasn’t found or the username was found but the password was wrong.

The error messages that the user could see related to input validation in the PC Building App during PC Build List creation, loading, as well as PC Build List editing and viewing within the ‘errorLabel’ control are identical and are as follows:

- “To create a PC Build List, the user needs to be logged into an account.” Occurs when trying to press the ‘Create PC Build List’ Button control on the main menu interface without being logged into a user account.
- “Build List Name Already Exists: Choose a unique name.” The PC Build List name has already been taken within the ‘pcbuildlist’ database table.
- “Invalid Build List Name: Must be 8 to 64 characters.” The PC Build List name was either too short or too long.
- “Invalid Description: Must be 255 characters or less.” The PC Build List description was too long.
- “Invalid Content: Must be 64 characters or less.” The PC Build List was of type ‘Guide’ and the content was too long.

- “Invalid Category: Must be 64 characters or less.” The PC Build List was of type ‘Guide’ and the category was too long.
- “Invalid Thumbnail URL: Must be 255 characters or less.” The PC Build List was of type ‘Guide’ and the thumbnail URL was too long.
- “Error: No PC Build List selected.” Occurs when the user attempts to load a PC Build List without selecting one from the ComboBox control on the main menu or PC part catalogue interfaces.
- “Error loading image from URL: ‘thumbnailURL’ - using default.” Occurs when a user input an invalid thumbnail URL and the PC Building App tries to load it within the published PC Build List view interface.

There is a single error message that the user could receive related to input validation in the PC Building App when trying to compare PC parts within the PC part catalogue interface. This error will show up in the ‘errorLabel’ control and is as follows:

- “Select at least two parts of the same type.” Occurs when the user does not select two PC parts of the same type when attempting to hit the ‘Compare Parts’ Button control.

The last error messages that the user could receive related to input validation in the PC Building App are whenever the ButtonBar Button controls are accessed when they aren’t needed. These errors will show up in the ‘errorLabel’ control as follows:

- “User is already logged into an account.” Occurs when the user presses the ‘Create Account’ or ‘Login’ Button controls within the heading ButtonBar control, despite already being logged into an account.

- “There is no account to logout of; User is already a guest.” Occurs when the user presses the ‘Logout’ Button control within the heading ButtonBar control, despite already not being logged into any user account.

The second type of error that could occur is related to database connection. Database connection errors occur when the PC Building App fails to reach the database needed to read or write data to. These errors can occur throughout the entire project, whenever the database is meant to be accessed, such as user account management, PC Build List management, PC part catalogue access, and published PC Build List viewing.

The error messages that the user could receive related to database connection in the PC Building App for user account management shown within the ‘errorLabel’ control are as follows:

- “Account creation failed. Unable to connect to the database.” Occurs when the user attempts to create a user account, but database connection fails.
- “Login failed. Connection to the database couldn’t be established.” Occurs when the user attempts to login to a user account, but database connection fails.
- “Failed to create PC Build List.” Occurs when the user tries to submit the creation of a new PC Build List, but database connection fails.
- “Failed to update PC Build List.” Occurs when the user tries to submit changes made to a PC Build List within the edit interface, but database connection fails.
- “Error: Failed to load selected PC Build List.” Occurs when the user tries to load a PC Build List to the load interface, but database connection fails.
- “Error: Connection failed while trying to get published PC Build Guides” Occurs when the user tries to load the published PC Build List view, but database connection fails.

The last type of error that could occur is related to PC Build Lists and their PC part compatibility checks. These error messages do not use the ‘errorLabel’ control, as they are meant to be present within a loaded PC Build List. Instead, the error messages related to compatibility issues create their own Label controls to display within the PC Build List load interface. The amount of compatibility checks within the PC Building App for PC parts are extensive, causing a lot of potential error messages to be displayed.

All of the potential compatibility check error messages that the user could be shown within the PC Building App’s PC Build List load interface are as follows:

- “CPU socket does not match Motherboard socket.” & “Motherboard socket does not match CPU socket.” Both of these error messages are displayed when the PC parts of type ‘CPU’ and ‘Motherboard’ are incompatible within the PC Build List due to having a socket mismatch.
- “CPU max memory speed exceeds Motherboard support.” & “Motherboard does not support CPU max memory speed.” Both of these error messages are displayed when the PC parts of type ‘CPU’ and ‘Motherboard’ are incompatible within the PC Build List due the CPU having a higher max memory than the Motherboard can support.
- “CPU socket not compatible with CPU cooler.” & “CPU Cooler not compatible with CPU socket.” Both of these error messages are displayed when the PC parts of type ‘CPU’ and ‘CPU Cooler’ are incompatible within the PC Build List due to having a socket mismatch.
- “RAM type is not compatible with Motherboard” & “Motherboard does not support this RAM type” Both of these error messages are displayed when the PC parts of type

‘Motherboard’ and ‘RAM’ are incompatible within the PC Build List due to the RAM being of a different type than the Motherboard supports.

- “RAM speed is not supported by Motherboard.” & “Motherboard does not support this RAM speed.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘RAM’ are incompatible within the PC Build List due to the RAM having a faster speed than the Motherboard supports.
- “Number of RAM modules exceeds Motherboard RAM slots.” & “Motherboard has insufficient RAM slots.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘RAM’ are incompatible within the PC Build List due to there being more RAM cartridges than the Motherboard supports.
- “Total RAM capacity exceeds Motherboard maximum memory.” & “Motherboard has insufficient maximum memory.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘RAM’ are incompatible within the PC Build List due to the amount of memory from the RAM cartridges exceeding the max amount of memory the Motherboard supports.
- “GPU interface is not compatible with Motherboard PCIe slots.” & “Motherboard PCIe slots do not support GPU interface.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘GPU’ are incompatible within the PC Build List due to the Motherboard not having an available PCIe slot that supports the GPU.
- “Motherboard form factor is not supported by the case.” & “Case does not support Motherboard form factor.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘Case’ are incompatible within the PC Build List due to the Motherboard being a different form factor size than the Case supports.

- “CPU Cooler height exceeds Case maximum CPU Cooler height.” & “Case cannot fit selected CPU Cooler due to height.” Both of these error messages are displayed when the PC parts of type ‘Case’ and ‘CPU Cooler’ are incompatible within the PC Build List due to the CPU Cooler not being able to fit within the Case’s height.
- “CPU Cooler radiator size not supported by selected Case.” & “Case does not support CPU Cooler radiator size.” Both of these error messages are displayed when the PC parts of type ‘Case’ and ‘CPU Cooler’ are incompatible within the PC Build List due to the CPU Cooler havinga radiator size that does not fit within the Case.
- “GPU length exceeds Case maximum GPU length.” & “Case cannot fit selected GPU due to length.” Both of these error messages are displayed when the PC parts of type ‘Case’ and ‘GPU’ are incompatible within the PC Build List due to the GPU being longer than the Case supports.
- “GPU width may not fit within the selected Case.” & “Case may not fit the selected GPU due to width.” Both of these error messages are displayed when the PC parts of type ‘Case’ and ‘GPU’ are incompatible within the PC Build List due to the GPU width exceeding the Case’s supported GPU width.
- “PSU does not have enough power connectors for the GPU.” & “PSU lacks necessary GPU power connectors.” Both of these error messages are displayed when the PC parts of type ‘Power Supply’ and ‘GPU’ are incompatible within the PC Build List due to the Power Supply not having enough power connectors to support the selected GPU.
- “Motherboard power connectors are not compatible with PSU.” & “PSU does not have the required power connectors for the Motherboard.” Both of these error messages are displayed when the PC parts of type ‘Motherboard’ and ‘Power Supply’ are incompatible

within the PC Build List due to the Power Supply not having the power connectors that the Motherboard requires.

- “PSU wattage is insufficient for the system.” This error message is displayed when the PC part of type ‘Power Supply’ is incompatible with the rest of the PC Build List due to not having enough wattage to support the entire computer.
- “Too many 3.5 inch drives for this Case.”, “Case has insufficient 3.5 inch drive bays.”, “Too many 2.5 inch drives for this Case.” & “Case has insufficient 2.5 inch drive bays.” All of these error messages are displayed when the PC parts of type ‘Case’ and ‘Storage’ are incompatible within the PC Build List due to not having enough HDD or SSD storage slots in the Case for the amount of Storage within the PC Build List.
- “Fan size (‘fanSize’) is not supported by the selected Case.” & “Selected Case does not support fan size (‘fanSize’).” Both of these error messages are displayed when the PC parts of type ‘Case’ and ‘Case Fan’ are incompatible within the PC Build List due to a Case Fan within the PC Build List being too big or small to fit within any of the Case’s fan slots.

Ethical Practices

With the PC Building App, it is important that the handling of user data is done securely. The PC Building App ensures that secure inputs, such as passwords, are not directly sent and received between the application and the database. The PC Building App instead encrypts password fields to ensure that user input passwords are never directly sent or received from the database. The database instead stores an encryption key that the PC Building App can check between user input and stored data. It is important that user data is handled in a safe and secure

way, which the PC Building App achieves by encrypting important user input, such as password fields.

Conclusion

Design Versus Implementation

The PC Building App project had many differences between its initial design and the actual implementation. Most of these were additions made to achieve the initial design objectives, but some phases of the PC Building App's design had to be adjusted as implementation was done. From the original design to the actual implementation the platforms used to implement the project changed, including the IDE the project was compiled in, as well as the database architecture used. An early part of the design phase that did not end up in actual implementation of the PC Building App was the addition of affiliate links being provided to the user for each PC part. These links would open in the user's browser of a store page to purchase the associated PC part.

First, the implementation of the PC Building App changed what IDE the project was compiled and programmed in. According to Amazon Web Services, an "An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently" (What is an IDE? - AWS). At first, in the initial design phase, it was listed that the PC Building App would be programmed and compiled in the Eclipse IDE. According to the Eclipse Foundation, "Eclipse is an Integrated Development Environment (IDE) for Java" (What is Eclipse?: The Eclipse Foundation). During implementation, however, the Eclipse IDE kept having issues. One of the most prominent of these issues was that the Eclipse IDE would become corrupted whenever the computer itself was turned off. This caused a lot of work to be lost due to the installation of the Eclipse IDE becoming corrupted upon starting the computer the next day. As a fix to this issue, a switch in IDEs was made and the IntelliJ IDEA was used in place of the Eclipse IDE. The IntelliJ IDEA is "an Integrated Development

Environment (IDE) for professional development in Java and Kotlin" (IntelliJ IDEA overview: Intellij idea). Similar to Eclipse, IntelliJ was an IDE platform that worked well with Java and JavaFX, the programming languages used for the user interface and programming logic of the PC Building App.

Second, the implementation of the PC Building App changed what database architecture the project used to read and write data tables to. At the initial design phase, it was listed that the PC Building App would use a MySQL database server to read and write info between the PC Building App and database tables. According to an article from Oracle.com, "MySQL is an open source relational database management system (RDBMS) that's used to store and manage data" (Erickson, 2024). Between the design and implementation phases of the PC Building App, along with the IDE being changed, there would also be a change in database architecture. This change was made at the same time as the switch to a different IDE, in that the MySQL database had given some setup issues, and it was easier to switch to a different database server than to adjust several phases of the implementation around those setup issues. The MySQL database server would be replaced by a PostgreSQL database server, which had similar compatibility with the PC Building App. According to PostgreSQL's website, "PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads" (PostgreSQL: About). Using these, testing and implementation of the PC Building App was a lot easier to get working on.

Third, one of the only design phase plans that did not make it to the final PC Building App project was the use of affiliate links. Affiliate links were meant to be a part of the PC part database table, 'pcpart', as a column of data. This column would be accessible within the PC part

catalogue, as well as user created PC Build Lists that had PC part entries. These affiliate links were meant to fulfill a potential business goal of allowing manufacturers or vendors to provide possible links for users to purchase the associated PC parts from. This idea was ultimately removed from implementation due to being outside the scope of the project. Put simply, the potential business benefit of the affiliate links was not feasible in this version of the PC Building App due to PC part entries being manually entered. This issue is covered in the 'Future Project Extensions' section in further detail.

Remaining Issues

Although the PC Building App fulfills its primary objectives, there are still some potential issues with the current implementation of the project. The remaining problems foreseen with the current PC Building App project are that the PC Building App relies on a locally created database, some of the PC part compatibility checks are done multiple times, and some user flow quality-of-life features are missing.

First, the PC Building App relies heavily on a database, consisting of four tables of data. If this database is inaccessible, the PC Building App will hardly function. This creates an issue where the database needs to be accessible at all times, but the scope of the project only covers the creation of a locally hosted database. This means that if a user on a different device than that of the database tries to run the PC Building App, they will ultimately fail to connect to the database, resulting in extremely limited functionality to their version of the PC Building App project. Within the PC Building App, there is a class called the 'PostgreSQLController', this class handles all actions related to interacting with the database. For the PC Building App to function on other clients, the 'PostgreSQLController' may have to be modified to alter the address the PostgreSQL database server is running on. This is not something that the PC

Building App has within its user interfaces, which means a normal user client would not have a means to adjust this. This is a major issue that lies in the PC Building App's scope of being a class project, and is discussed further in the 'Future Project Extensions' section of this report.

Second, one of the major objectives of the PC Building App's PC Build Lists are its compatibility checks. These compatibility checks run through every PC part associated with a PC Build List and ensure they are compatible, and, if otherwise, provide helpful messages to the user on why there are compatibility issues and what PC parts are causing incompatibility. Due to the nature of computer parts, their compatibility is very intricate, and implementing automatic checks was a large task. When creating the compatibility checks, it is important that all potential PC part compatibilities are accounted for. Some of these include CPU-to-Motherboard compatibility, CPU-to-CPU-Cooler compatibility, Motherboard-to-CPU compatibility, Motherboard-to-RAM compatibility, and so on. As all of these compatibility checks were created, some end up checking the same types of compatibility. In the example given, there are two compatibility types, CPU-to-Motherboard and Motherboard-to-CPU, that would essentially check and display the same error messages. This was taken into consideration late into implementation, and most duplicate compatibility checks were removed, but some remain in the current PC Building App. One of the remaining duplicate compatibility checks includes the Case-to-Motherboard compatibility check and the Motherboard-to-Case compatibility check. This issue isn't major, but in the case that a PC Build List has this compatibility issue, there would be two duplicate error messages generated and displayed to the user under the 'Motherboard' and 'Case' sections of the user's PC Build List.

Third, the PC Building App is missing some important quality-of-life user flow features. Although this isn't a major concern, it is something that user's would notice as they attempt to

navigate and use the PC Building App. One of the main user flow quality-of-life features that is not included is the ability to interact with Button controls using the keyboard. An example of this being an issue is when a user attempts to login to a user account. When logging in, the user is prompted to enter a username and password. After typing both of these entries, there is no keyboard shortcut to enter the information they provided. Instead, the user is required to use their mouse and click the Button control to enter their login credentials. This is a minor issue that was not noticed until after the implementation of the PC Building App.

Learning Experiences

There were many new things that I learned while working on the PC Building App project. Some of these are specific learning experiences such as learning the differences in database architectures of MySQL and PostgreSQL. There are some more bigger experiences such as connecting a database to an application. Most importantly, there is the experience of working on a larger project.

The PC Building App was a large project, and one of the first larger projects I've worked on. What I learned from this was my own potential to work on something much larger than what I've done for other courses. I feel that learning to work on a large project that had deadlines and general expectations made it so that I am better prepared for actual work on application projects. I feel that some of the main skills that I can apply to other applications from this PC Building App include my time management, ability to adjust to different programming environments, as well as design applications and implement those designs in my own way.

Another learning experience from the PC Building App is applying a database to an actual real-world application. In courses that cover database management, I had never actually worked on a project that would make use of a database. It was simply learning how to interact

with a database server directly. The PC Building App showed me how to take what I learned from database management courses and apply it to an actual application, the PC Building App. Instead of directly interacting with the database, many of the database interactions were between the PC building App itself and the database server. I feel that I can apply this to any future projects that require data management because I now believe I have the skills to make use of a database server to manage data for a real-world application.

Finally, some smaller learning experiences from the PC Building App include technical and syntax experiences, such as switching between database architectures. This was one of the more prevalent of the minor learning experiences, in that a lot of the database management calls, such as query functions, had to be called slightly differently. Although query statements did not change between these architectures, learning how to call upon a PostgreSQL server versus a MySQL server was certainly a learning experience. I had many smaller learning experiences with the PC Building App that were largely around re-learning Java-based programming, as well as re-learning database management. I feel that these experiences can be applied to other projects in that I am better prepared to learn things, whether those are new or things to re-learn. I feel that it is important that I am able to adapt to these subtle changes between any future projects I work on.

Future Project Extensions

The PC Building App has fulfilled its primary objectives, and is overall a successful project. The PC Building App serves as a learning experience, as well as my first real project. The PC Building App may not be perfect, but it has plenty of room for improvement. Continuing to work on the project, some extensions made to it would be to solve any remaining issues, expand the project scope, and expand the objectives of the PC Building App to meet that scope.

As for remaining problems, I would like to continue the PC Building App by reviewing the PC part compatibility checks, as well as introducing user flow quality-of-life that is missing from the current iteration of the PC Building App. Both of these issues were covered in the 'Remaining Issues' section of the report. To summarize, the PC part compatibility checks have some compatibility checks that repeat themselves. To fix this, I would design a compatibility check structure that ensures duplicate checks are removed from the PC Building App's processes. User flow quality-of-life that is missing mainly refers to the lack of keyboard control flow for Button controls. This is a minor issue, and can easily be fixed within an initial PC Building App extension.

Next, expansion of the project's scope. This would take the PC Building App, which was previously designed around being a class project, and making it a potential real-world product. Some changes to be made to achieve this would be introducing version control, converting the application to use an actual database server rather than a local database, and expand upon the PC part catalogue to include affiliate links. Version control would allow the PC Building App to be edited in versions, allowing users to use previous versions as a backup. The PC Building App currently relies on a locally hosted server, one of the changes to making this a project a real product would involve changing the database to one hosted on a server that clients can connect to over the internet.

Lastly, the PC Building App could have much of its primary processes be updated to match a larger, real-world product scope. Some of these include the PC part catalogue, PC Build Lists, and user account management. The PC part catalogue was created for the scope of a class project, this would have quite a few potential changes, one of which being the ability for PC parts to include an affiliate link for users to purchase the PC parts that are shown within the PC

Building App. The PC Build Lists are created and viewed from within the application. A potential change to the PC Building App could be to introduce ways of viewing published PC Build Lists on the web, instead of solely within the PC Building App's application. This would also mean that PC Build Lists need to become shareable outside the PC Building App, such as creation of shareable links. User account management within the current PC Building App is very rudimentary, only requiring a username and password. For the PC Building App to become a real-world product, the PC Building App's user account management would have to include further credentials, such as an email or phone number, to allow the PC Building App to further secure user accounts, via the use of two-factor-authentication.

Bibliography

Amazon Web Services. (n.d.). *What is an IDE? - integrated development environment explained - AWS*. Amazon Web Services. https://aws.amazon.com/what-is/ide/

Eclipse Foundation. (n.d.). *What is Eclipse?: The Eclipse Foundation*. Eclipse. https://www.eclipse.org/home/whatis/

Erickson, J. (2024, August 29). *MySQL: Understanding What It Is and How It's Used*. Oracle. https://www.oracle.com/mysql/what-is-mysql/

JetBrains. (2025, April 7). *IntelliJ IDEA overview: Intellij idea*. IntelliJ IDEA. https://www.jetbrains.com/help/idea/discover-intellij-idea.html

Oracle. (n.d.). *javafx.scene.control (JavaFX 24)*. openjx.io. https://openjfx.io/javadoc/24/javafx.controls/javafx/scene/control/package-summary.html

The PostgreSQL Global Development Group. (n.d.). *PostgreSQL: About*. PostgreSQL. https://www.postgresql.org/about/

Appendix

Appendix A: Vision Document

System Vision Document

Keith Holcomb

PC Building App

Problem Description

PCs are an essential to productivity and staying connected on the web. As time has gone on, PCs have been getting better and better, but they've also become more complicated for most people to understand. There are a lot of different parts that go into a PC, and it can be confusing for someone that wants to make their own PC. It is important that the PC Building App is able to have users create an account and make a PC Build list, which would show what parts the user has picked for the PC. The PC Building App will need to provide info about specific PC parts from popular brands that user's may plan to purchase. The PC Building App will also need to check compatibility between the parts the user has selected to ensure they will work when built into the PC. The PC Building app should also have a way to link each part to its associated product page, so the user can purchase the part(s).

System Capabilities

The PC Building App should be capable of:

- Creating and storing user accounts as a table in the database.
- Collecting and storing information about different PC parts as a 2nd table in the database. This is not meant to be an extensive list of products, but a small general/example list of popular or affordable parts.
- Creating and storing PC Build lists created by the users as a 3rd table in the database.
- Use of a linking table, which will hold the id of a PC part and the id of the PC Build list it belongs to, as well as additional info (such as quantity).
- Providing guidance to users who are creating lists, such as checking PC part compatibility, or recommending parts to add to the list.

System Vision Document

Keith Holcomb

PC Building App

The PC Building App's specifications:

- Use PostgreSQL to create and manage a minimum of 4 tables; 1 for user accounts, 1 for user created lists, 1 for PC Parts catalogue, 1 for linking PC parts from the catalogue to user created lists.
- Use Java & JavaFX to handle the PostgreSQL database for the front-end.
  Ex. A class for user created lists would be used to create a list object that displays the data of a user created list within the application.
- Use Java & JavaFX to create a GUI capable of having the user create/login-to an account, create and view lists, view the catalogue of PC parts within the application.
- A user registered in the account database should be able to create a PC Build List. This List will be created empty, and display each part that the user needs to add to the list.

Business Benefits

The deployment of this PC Building app will provide the following business benefits:

- Simplify the process of interested users in finding PC parts for their PC build.
- Educate users on the basics of building a PC, such as the necessary parts and the compatibility between different parts.
- Project database can be expanded to maintain correct and current information about PC parts and their manufacturers, such as relative price.
- Project can be expanded to provide affiliate links to product parts' store pages for users to potentially purchase from.

Appendix B: Functionality, Usability, Reliability, Performance, Supportability Document

FURPS+ Document

Keith Holcomb

PC Building App

System Requirements:

Functional Requirements:

- PC Part selection.
  - Database of PC parts/components (CPU, GPU, RAM, Motherboard, etc.)
  - Filtering and sorting options (price, brand, compatibility, etc.)
  - Ability to compare multiple parts (parts are comparable if they are of the same type (CPUs, GPUs, etc).
- PC Build planning (via list)
  - List-like interface to add PC components.
  - Compatibility checks between parts added to the list.
  - Estimated build cost calculation (PC Build List/Guide total price).
- PC Build account/guides
  - Ability to create and login to an account
  - Ability to save and share custom PC Builds from user accounts.
  - Access to recommended PC Build guides (Budget, Gaming, etc.)

Nonfunctional Requirements:

- **Usability Requirements:**
  User Interface/Experience (JavaFX application)
  - Easy to navigate interface
  - Responsive design for various screen sizes (1080p, Ultrawide, windowed, fullscreen etc.)
  - Clear and helpful instructions/tooltips
- **Reliability Requirements:**
  Data accuracy/security
  - Accurate and up-to-date PC part information
  - Reliable compatibility checks
- **Performance Requirements:**
  Responsiveness and Resource Usage
  - Fast loading times and smooth navigation
  - Fast build price calculation updates
  - Low memory and power consumption
- **Security Requirements:**
  Security and protection
  - Secure user data storage and transmission
  - Encrypted user account information (password encryption)
  - Protection against unauthorized access of data (PostgreSQL built-in database security)

FURPS+ Document

Keith Holcomb

PC Building App

- **Design Constraints:**
  PC parts list will be simplified:
  - PC parts list in the database will not be modifiable via the application.
  - PC parts list will not be updated in real-time (via methods like data skimming), but use an example list of parts, created by myself.
  - PC parts list will be a list of a handful of popular parts from different manufacturers and brands (Ex. Intel CPU and AMD CPU, NVIDIA GPU and AMD GPU, Corsair PSU and Thermalright PSU, etc).

  Account creation will be simplified:
  - Account creation will not use an email or any secondary authentication upfront.
  - Account creation will simply require a username, password and a password confirmation.
  - Account login will only require the username and password.

  PC Build Guides will be simplified:
  - Guide viewing will simply show that a guide exists, and some of its details.
  - Guides won't actually be able to have their parts viewed.
- **Implementation Requirements:**
  Programming languages
  - Latest version of Java with JavaFX to program the application and the interface.
  - IntelliJ will be used to compile the Java application.

  Development platform
  - The primary target platform is desktop.

  Database
  - Latest version of PostgreSQL will be used for the database.
  - Database will store user info, component list, as well as PC Build lists, all in separate tables.
- **Interface Requirements:**
  User interface (UI) Design
  - The visual UI should be simple and easy to navigate (Simple font, and contrasting colors (shades of blue and gold), etc.)
  - The layout of the UI should consist of

    Options in the top-left to create an account or login/log-out of an account.

    On the left side of the application there will be recommended Build lists and a 'Getting Started' section.

FURPS+ Document

Keith Holcomb

PC Building App

In the center of the screen, there should be an option to browse the PC parts list/catalogue.

On the right side of the application there will be an option to create or load a PC Build list.

- Windows for the application will be locked to specific resolutions that fit within 720p definition. This ensures the viewing window can be seen on modern screen sizes, such as 720p, 1080p, etc.

User experience (UX) Design

- Simple user flow designs

  Ex. When entering account information, the user can enter their username and press the 'tab' key to go to the password box without having to click on it.

- **Physical Requirements:**

  The PC Building App is a software application and doesn't have any direct physical requirements.

- **Supportability Requirements:**

  Maintenance

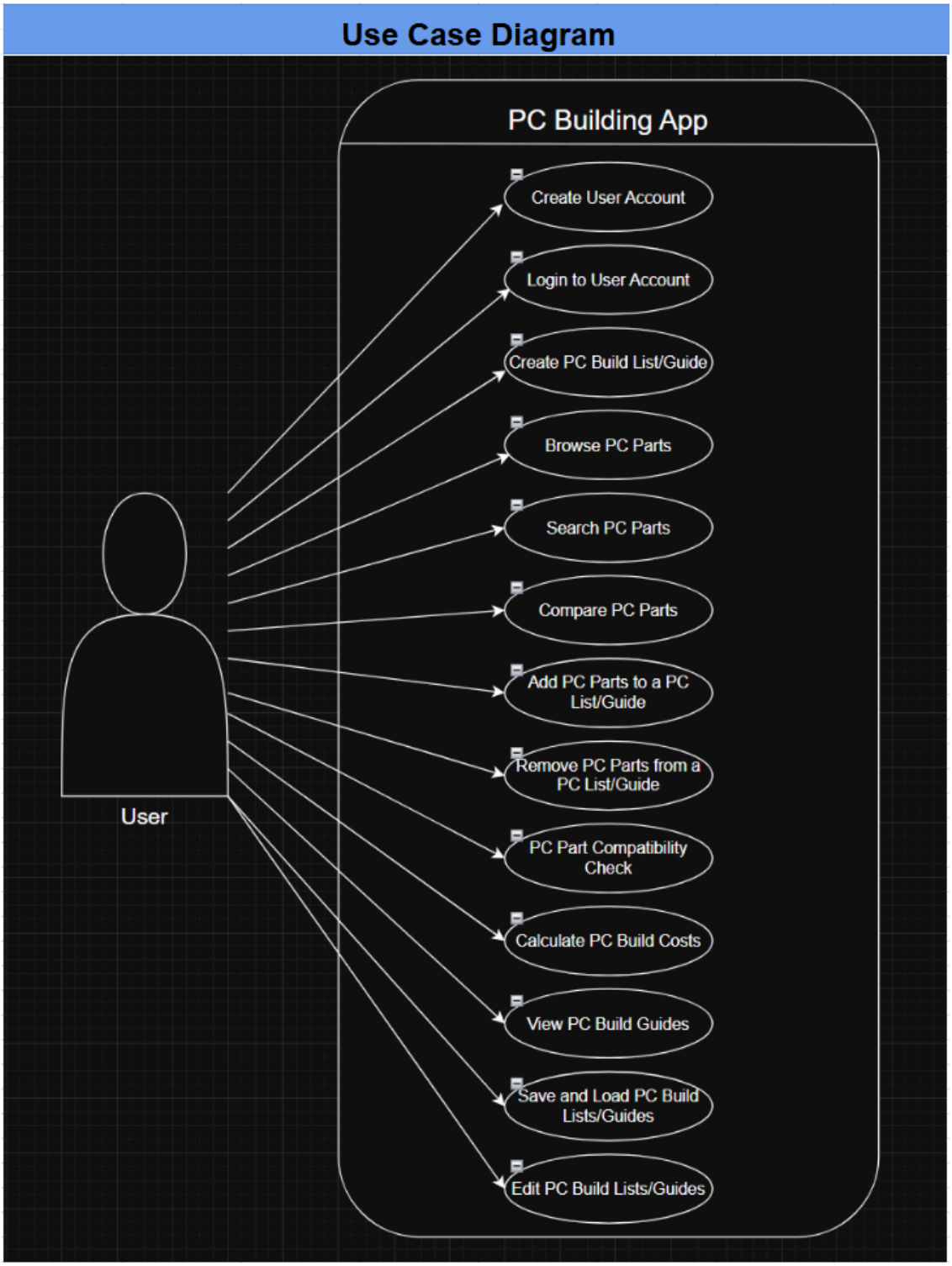
  - PostgreSQL database will be able to backup the database.

  Documentation

  - Developer documentation in the form of code comments.
  - Simple user documentation will be within the application to help the user get started.

Appendix C: Use Case Realization (Use Case Diagram & Detailed Descriptions Table)

Use Case Diagram:



Use Case Detailed Descriptions Table:

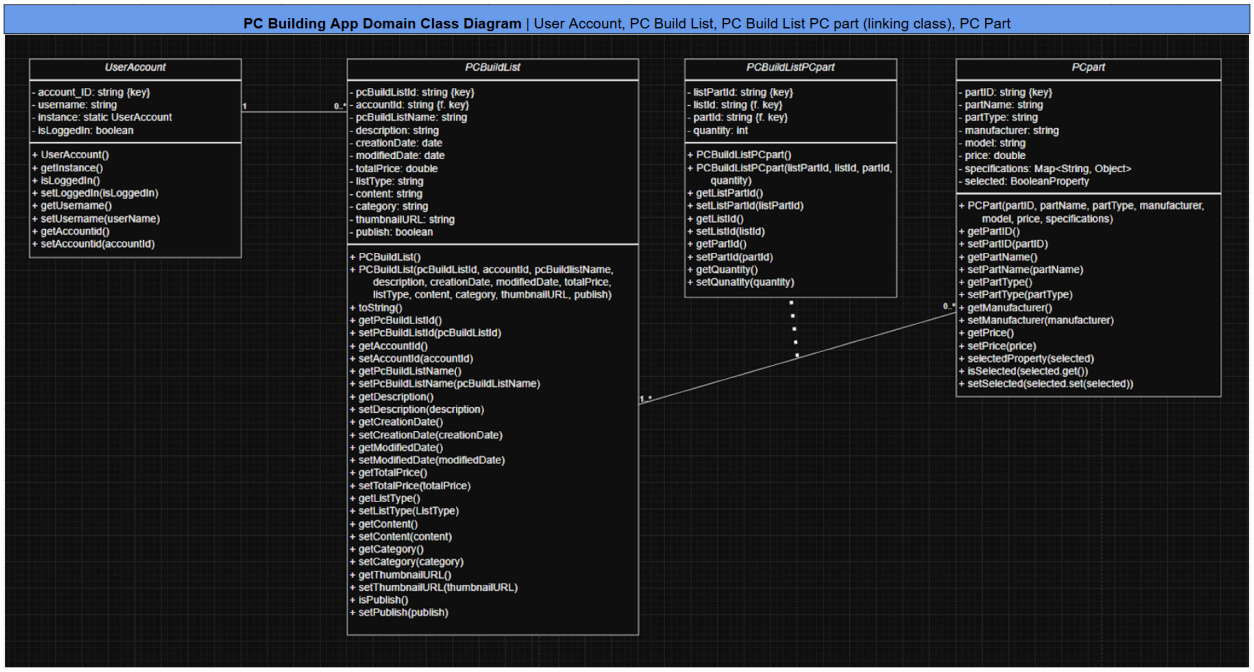
PC Building App | Use Case Table										
Use Case Name/Scenario	Triggering Event	Description	Actor	Related Use Case(s)	Stakeholder(s)	Precondition	Postconditiion	Flow of Activities		Exception Conditions
								Actor (User)	System	
Create User Account	User indicates to create a user account.	The user can enter a username and a password, and the system will assign an account number which will tie into creating a PC Build list	User	Maybe related to 'Login to user account'	N/A	The user doesn't have a pre-existing account	A new user account is created and the user is logged in.	1. User indicates they want to make an account. 2. User enters a username and password.	1.1. System prompts user to enter a username and password. 2.1. The system creates an account object using the provided credentials from the user.	The user is already logged in to an account.
Login to User Account	User indicates to login to a user account.	The user can login to their existing account using the username and password they used for account creation	User	Maybe related to 'Create user account'	N/A	The user has a pre-existing account	The user is successfully logged into their account.	1. User indicates they want to login to an account. 2. User enters their username and password	1.1 System prompts the user to enter login credentials. 2.1 System checks that the username and password match. 2.2. System logs the user into their account.	The user is already logged in to an account.
Browse PC Parts	User indicates to browse the PC parts catalogue.	The user can browse available PC parts (CPU, GPU, RAM, etc.)	User	Related to 'Search PC Parts' and 'Compare PC Parts'	N/A	The application has launched successfully	The user views the list of available PC parts.	1. User indicates they want to view the catalogue of parts. 2. User is shown the catalogue of PC parts.	1.1. System brings up the catalogue of PC parts that the user can browse.	System fails to retrieve any data of PC Parts.
Search PC Parts	User defines keywords in the search box of the PC parts catalogue.	The user can search for a specific part by typing keywords (ex. RTX 4090, Corsair, etc.) in a search bar	User	Related to 'Browse PC Parts'	N/A	The application has launched successfully	The user views a list of PC parts that match the keyword search criteria.	1. User clicks on the search bar in the PC parts catalogue, and enters a value. 2. User sees an updated version of the catalogue only containing things related to their search.	1.1. System queries the catalogue of PC parts based on the keyword(s) the user input in the search box.	System fails to retrieve any data of PC Parts.
Compare PC Parts	User selects more than one PC part within the PC parts catalogue.	The user can compare the compatibility and price of different PC parts side-by-side	User	Related to 'Browse PC Parts'	N/A	The user has selected more than one PC part in the PC Part components list.	The user views a table comparing each PC part selected by price, brand, and compatibility.	1. User selects more than one PC part in the catalogue and clicks to compare the selected parts. 2. The user sees the side-by-side comparison.	1.1 System displays compact side-by-side details of each PC part the user selected.	System fails to retrieve any data of PC Parts. The user attempts to compare a single part to nothing.
Create PC Build List/Guide	User indicates to create a PC Build List/Guide.	The user can create a PC Build List/Guide, which will allow them to add different PC parts to a list	User	Related to 'Save and Load PC Build List/Guide'	N/A	The user is logged into an account and the user clicks to create a new PC Build list	An empty PC Build list is created, and is shown to the user.	1. User indicates they want to create a new Build List. 2. User indicates what type of list (Regular or Guide). 3. User enters the name of the list.	1.1 System prompts user to indicate type of list (Regular/Guide). 2.1. System prompts to enter a name for the list. 3.1. System creates the list object, assigning the list type and name to it, leaving all part options empty.	The user is not logged into an account.
Add PC Parts to a PC Build List/Guide	User indicates to add a PC part in the catalogue to a PC Build List/Guide.	The user can add a component they found while browsing the PC part catalogue to their own list	User	Related to 'Create PC Build List/Guide'	N/A	The user has created a PC Build list	The PC part the user selects will be added to the user's PC Build list.	1. User clicks to add a PC part to a PC list/guide. 2. User indicates which list to add the PC part to.	1.1 System prompts the user to specify which list of theirs to add the PC part to. 2.1. System updates the list to include the given PC part	The user is not logged into an account. The user does not have a list currently created/loaded.
Remove PC Parts from a PC Build List/Guide	User indicates to remove a PC part from the PC Build List/Guide.	The user can remove a component they no longer want in their PC build list	User	Related to 'Create PC Build List/Guide'	N/A	The user has a PC Build list that isn't empty	The PC part the user selects to remove will be removed from the user's PC Build list.	1. User clicks to remove a specific PC part from a PC list/guide. 2. User sees the list has updated to not include the part they wanted removed.	1.1. System updates the list, removing the specified PC part from the list.	The user does not have a list currently created/loaded.
PC Part Combatibility Check	User indicates to add a PC part in the catalogue to a PC Build List/Guide.	The application detects compatibility issues of selected parts in a list automatically, and gives the user visible warnings on incompatible parts in their list.	User	Related to 'Add PC Parts to a PC Build List' and Remove PC Parts from a PC Build list.	N/A	The user has added PC parts in their PC Build list	Compatibility warnings are displayed if any compatibility issues are detected.	1. User added a PC part to their list and is now viewing the list. 2. User can see whether or not there is a compatbility issue with the PC parts in the list.	1.1 System checks compatibility between all parts within the list. If there is a compatbility issue, System prompts the user of any parts that have an issue, and what the issue is.	The user does not have a list currently created/loaded.
Calculate PC Build List/Guide total price	User indicates to add a PC part in the catalogue to a PC Build List/Guide.	The application calculates the estimated total cost of all PC parts in the user's PC Build list	User	Related to 'Add PC Parts to a PC Build List' and Remove PC Parts from a PC Build list.	N/A	The user has added PC parts in their PC Build list	The total estimated cost of the PC Build list is displayed.	1. User added a PC part to their list and is now viewing the list. 2. User can now see the total price of all PC parts in the list at the bottom of the list.	1.1 System checks each part in the list for their price. 1.2. System adds all prices together as a total price. 1.3. System displays the total price at the bottom of the user's list.	The user does not have a list currently created/loaded.
View PC Build Guides	User indicates to view PC Build Guides.	The user can access pre-built guides for popular PC Builds (Budget, Gaming, etc.)	User	Related to 'Edit Details for PC Build List/Guide'	N/A	The application has launched successfully	The user is shown a pre-built PC Build list based on what they choose.	1. User clicks to view PC Build Guides. 2. User is shown a list of PC Build Guides.	1.1. System displays all uploaded PC Build Guides.	System fails to retrieve any data of PC Build Guides.
Save, Load and Delete PC Build List/Guide	User indicates to save/load a PC Build List/Guide	The user can save their list in the database under their account and load it	User	Related to 'Create PC Build List/Guide'	N/A	The user is logged in and has created a PC Build list that is not empty	The user's PC Build list is saved in the database under the user's account ID and can be loaded so long as the user is logged in.	1. User clicks to save/load a PC Build List/Guide. 2. The user specifies what PC Build List/Guide to load. 3. The user is shown the loaded PC Build List/Guide.	1.1. The current PC Build List/Guide is written and saved to the database under the user's account. 1.2. The user is shown all available PC Build List/Guides that are under their account to load. 2.1. System loads the PC Build List/Guide in place of the previous one.	The user is not logged into an account. The user does not have any list/guide associated with their account.
Edit Details for PC Build List/Guide	User indicates to edit info of a PC Build List/Guide	The user can modify info used in displaying their PC Build List/Guide, including to publish a Guide.	User	Related to 'Save, Load and Delete PC Build List/Guide' and 'View PC Build Guides'	N/A	The user is logged in and is currently viewing the PC Build List to add PC Parts.	The user is taken to a menu that preloads the lists data, and the user can edit each individual section. This includes the Publish option for Guides.	1. User clicks to edit a PC Build List/Guide. 2. The user changes specifics of their PC Build List/Guide. 3. The user clicks to save the changes and is returned to the list with its changes applied.	1.1. The current PC Build List/Guide is passed via id to the new scene. 1.2. The user is shown all the fields of their PC Build List/Guide and can make changes to them. 2.1. System saves any changes and loads the previous scene.	The user is not logged into an account. The user does not have any list/guide associated with their account.

Appendix D: List of Things, Domain Class Diagram, & Entity-Relationship Diagram

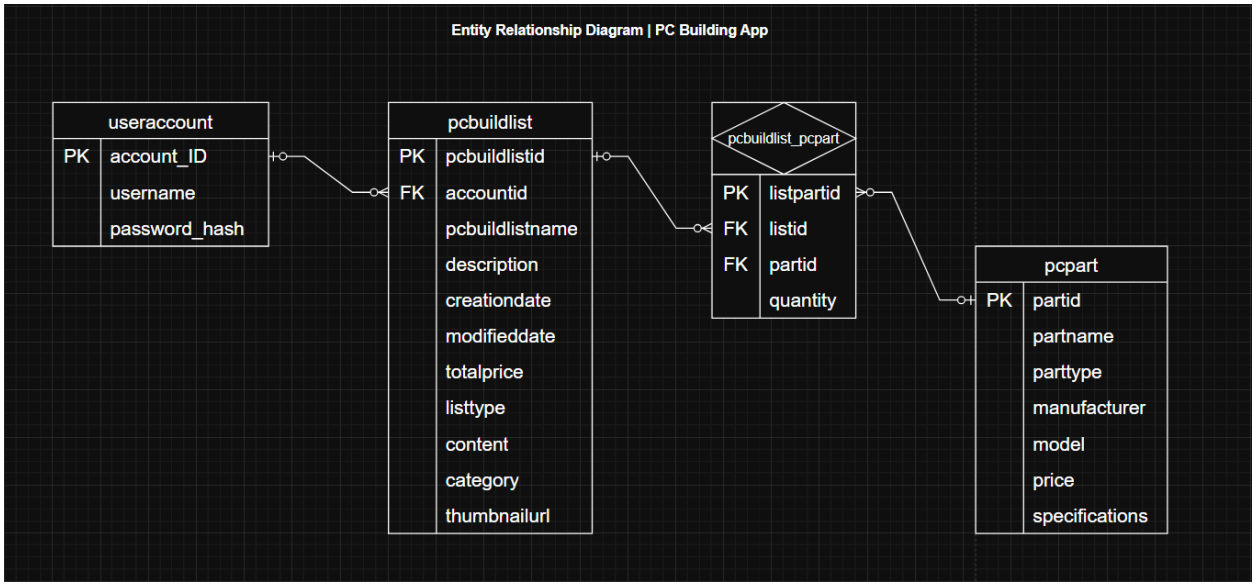
List of Things:

List of Things in the PC Building App	
Thing	**Description**
User Accounts	Individuals that use the app, represented by the account they create/login-to.
PC Parts	GPUs, CPUs, RAM etc. Each PC part has its own attributes to be stored.
PC Build Lists	A collection of PC parts created by the user.
PC Build Guides	A collection of PC parts pre-created in the system. These would essentially be the same as a build list, but contain extra attributes specifically for a guide.

Domain Class Diagram:

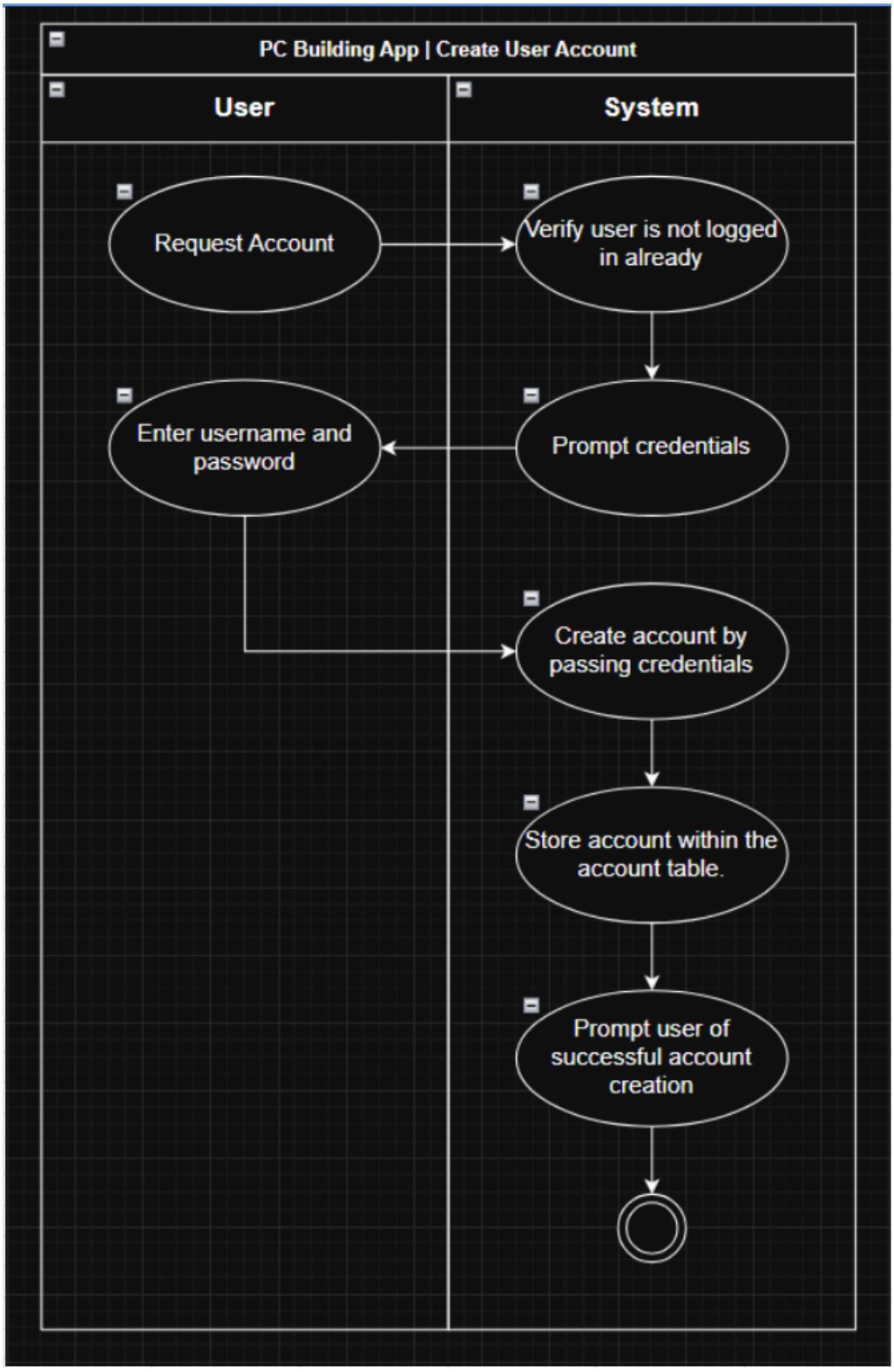


Entity-Relationship Diagram:

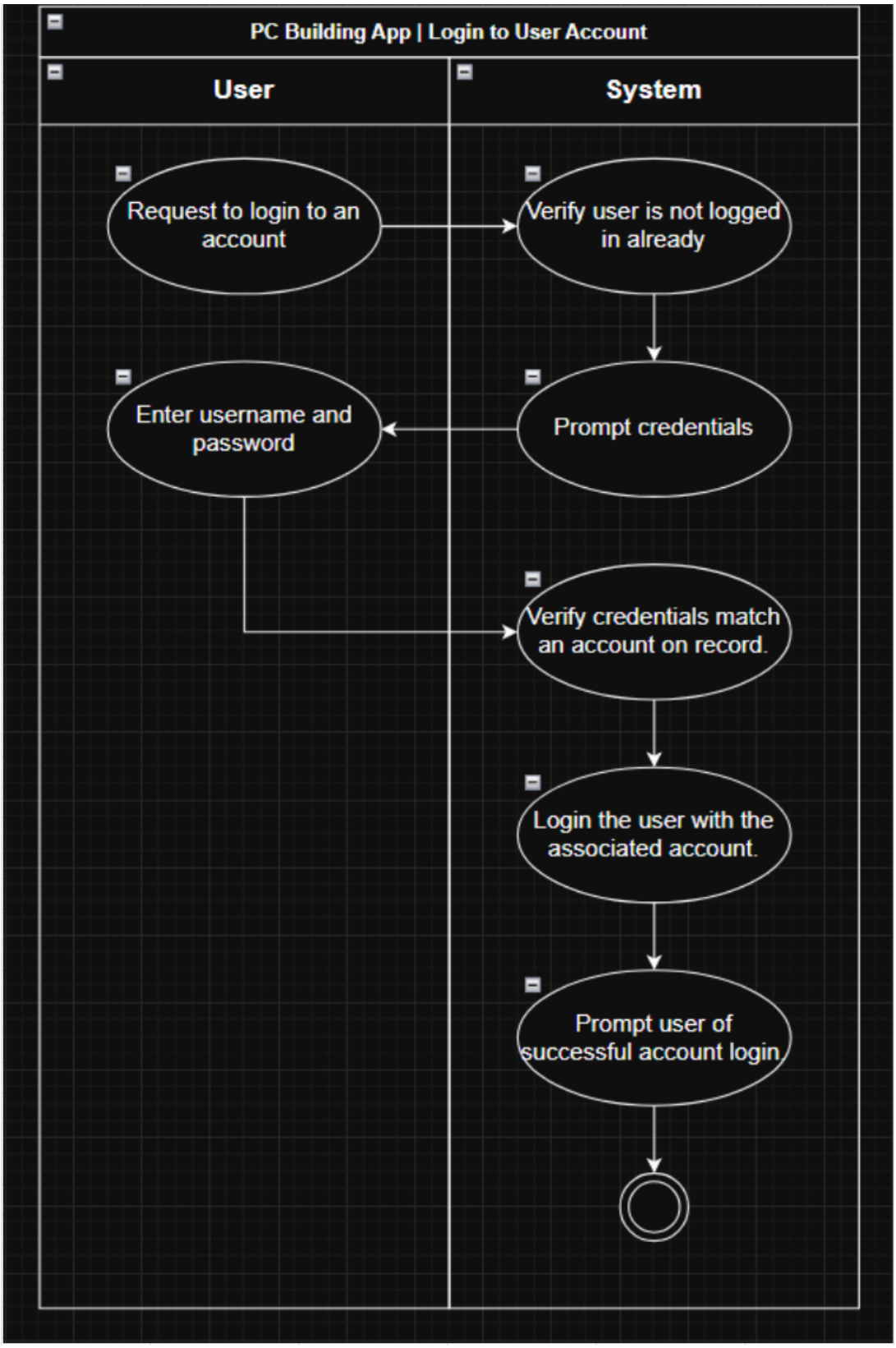

Entity Relationship Diagram | PC Building App
useraccount
PK
account_ID
username
password_hash
pcbuildlist
PK
pcbuildlistid
FK
accountid
pcbuildlistname
description
creationdate
modifieddate
totalprice
listtype
content
category
thumbnailurl
pcbuildlist_pcpart
PK
listpartid
FK
listid
FK
partid
quantity
pcpart
PK
partid
partname
parttype
manufacturer
model
price
specifications

Appendix E: Activity Diagrams

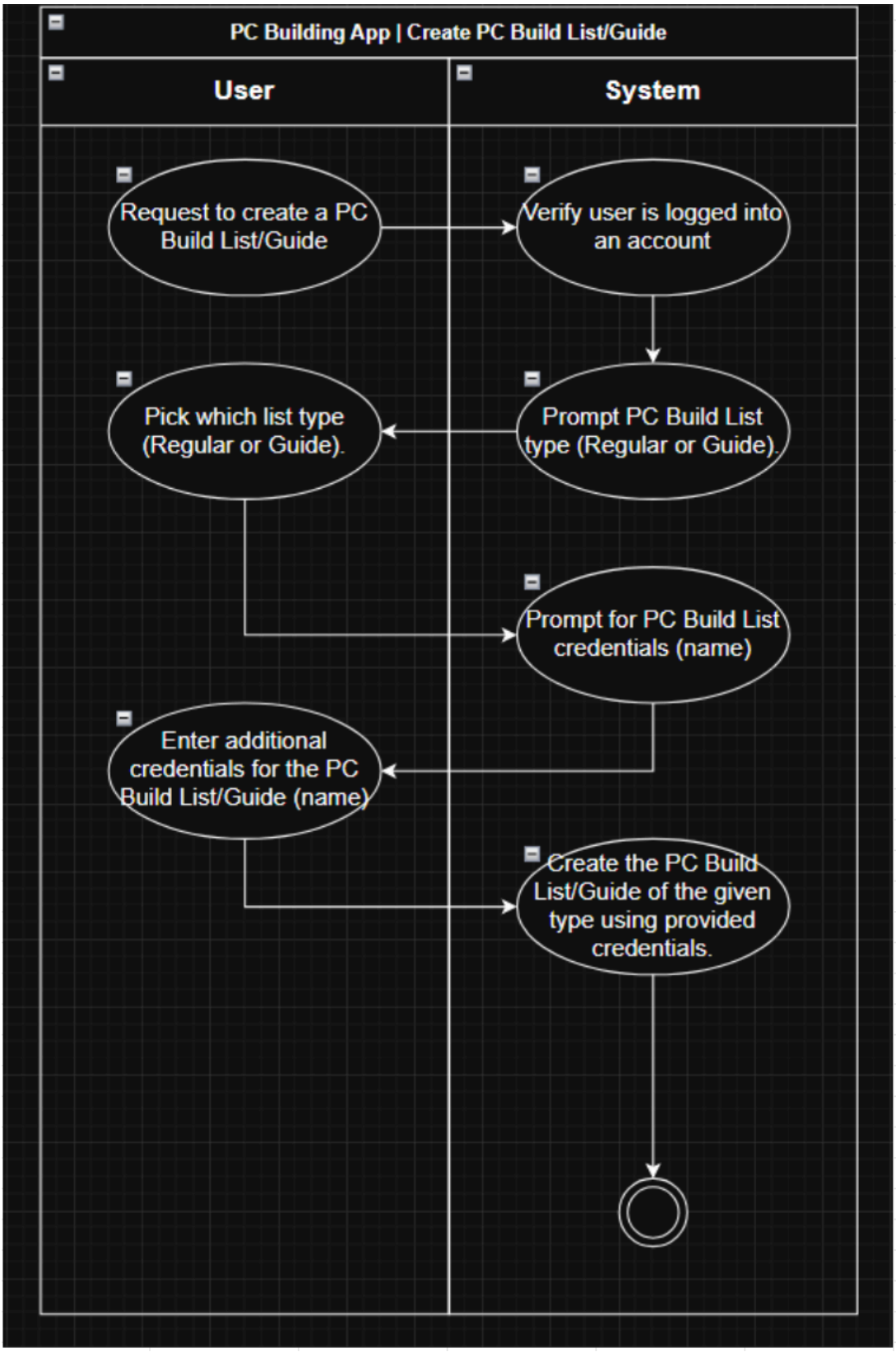
Create User Account Activity:



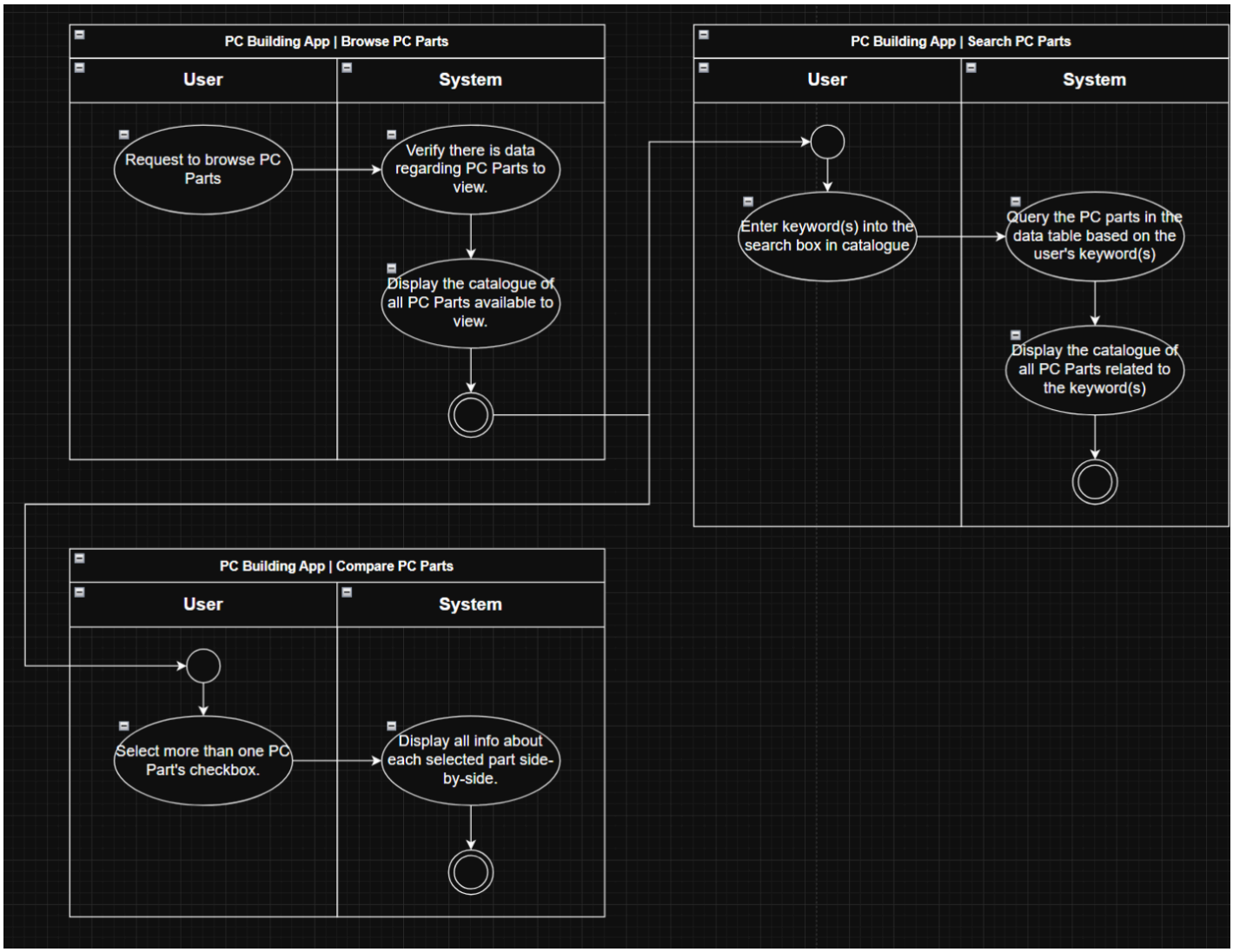
Login to User Account Activity:



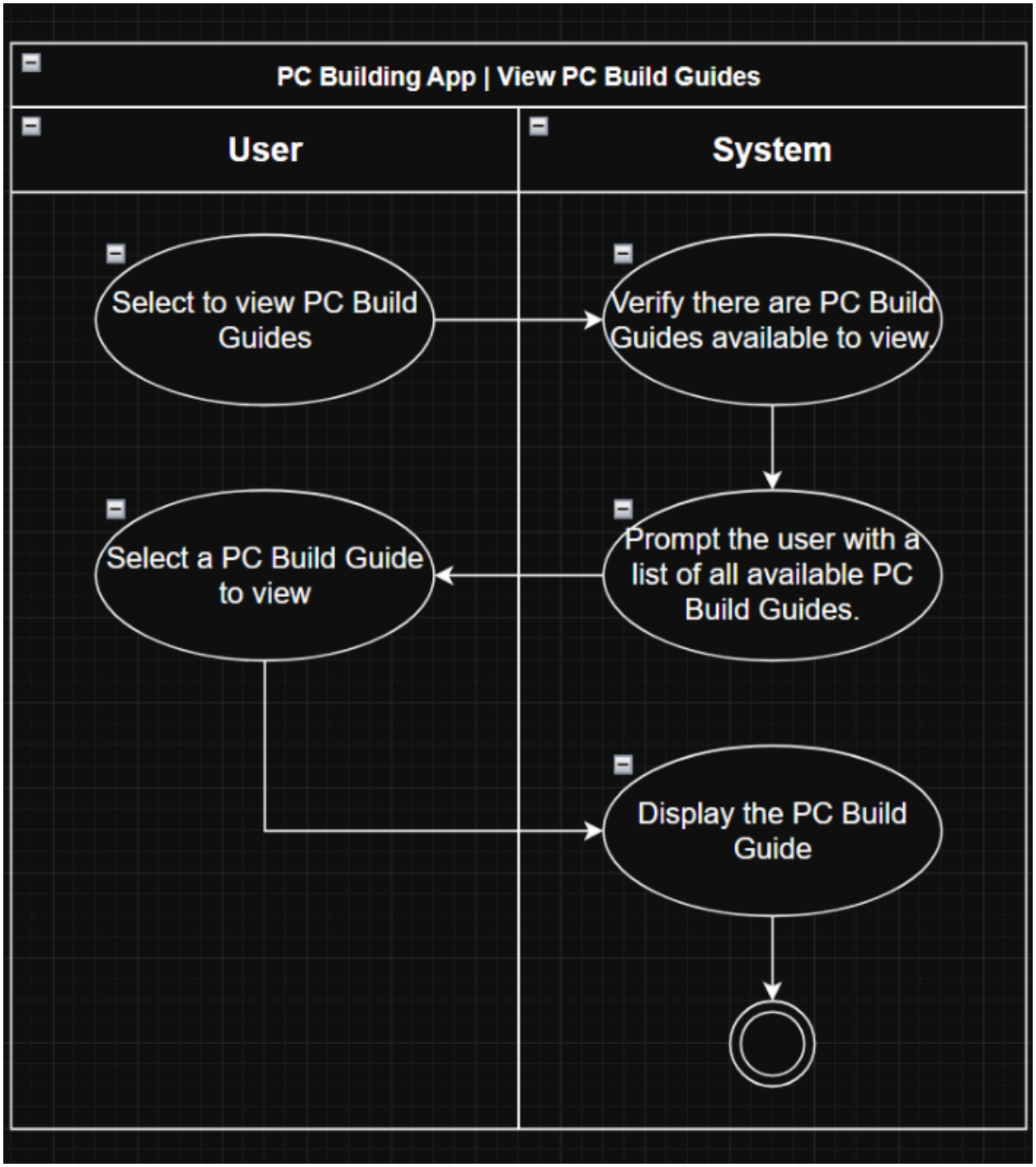
Create PC Build List Activity:



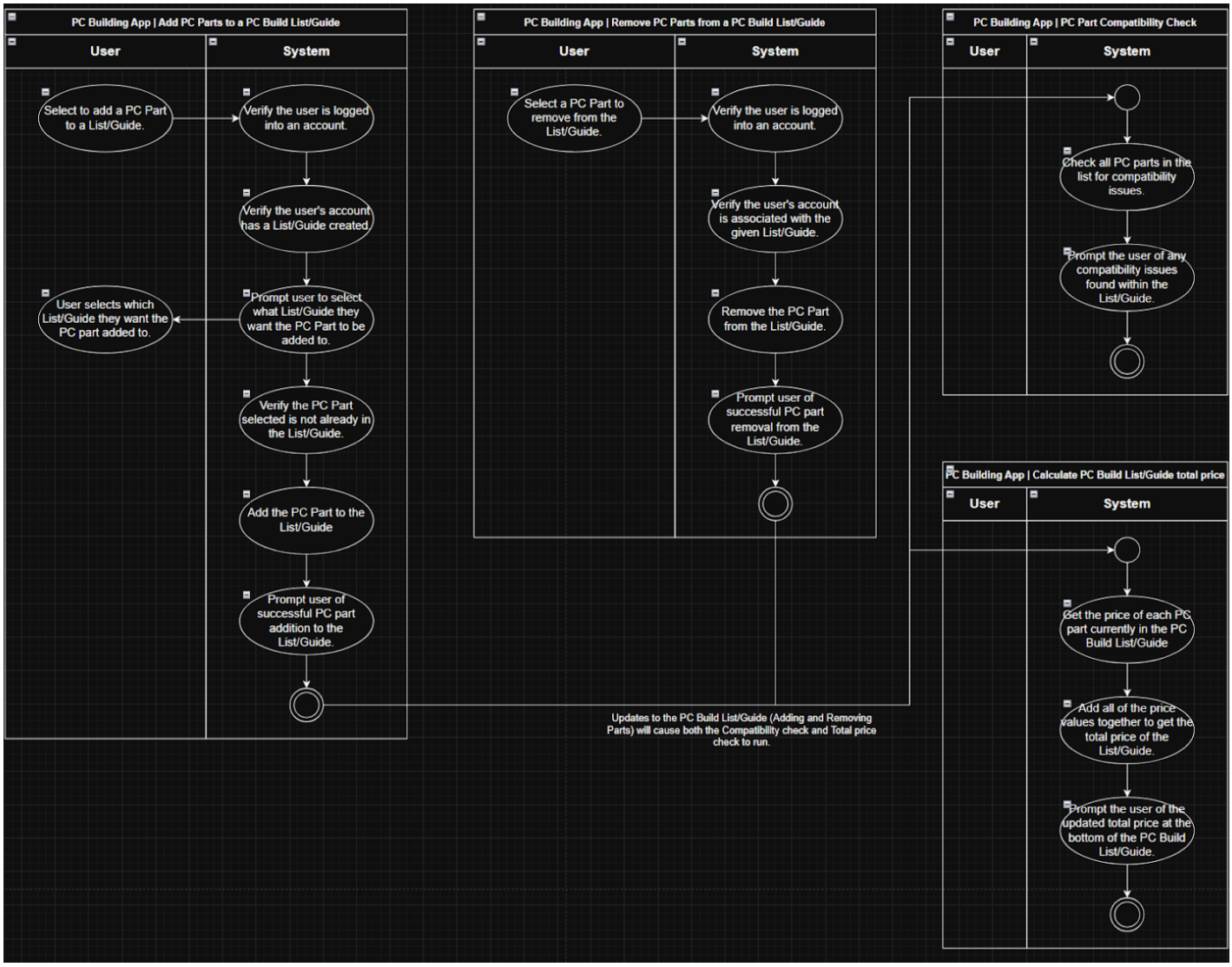
PC Part Catalogue Activities (Browse, Search, & Compare):



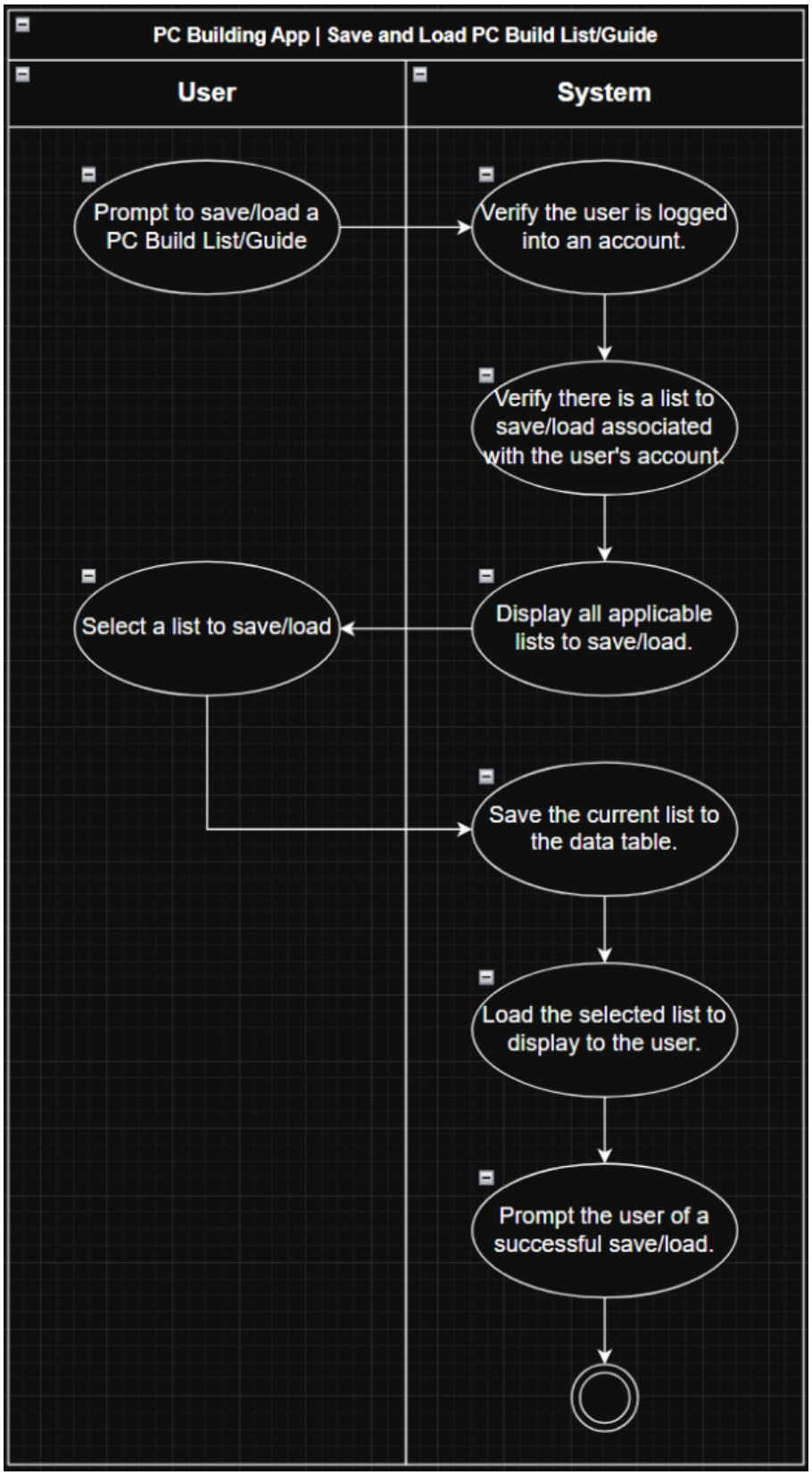
View PC Build Guides Activity:



PC Build List Activities (Add/Remove PC Parts, Total Price Updates, PC Part Compatibility Checks):

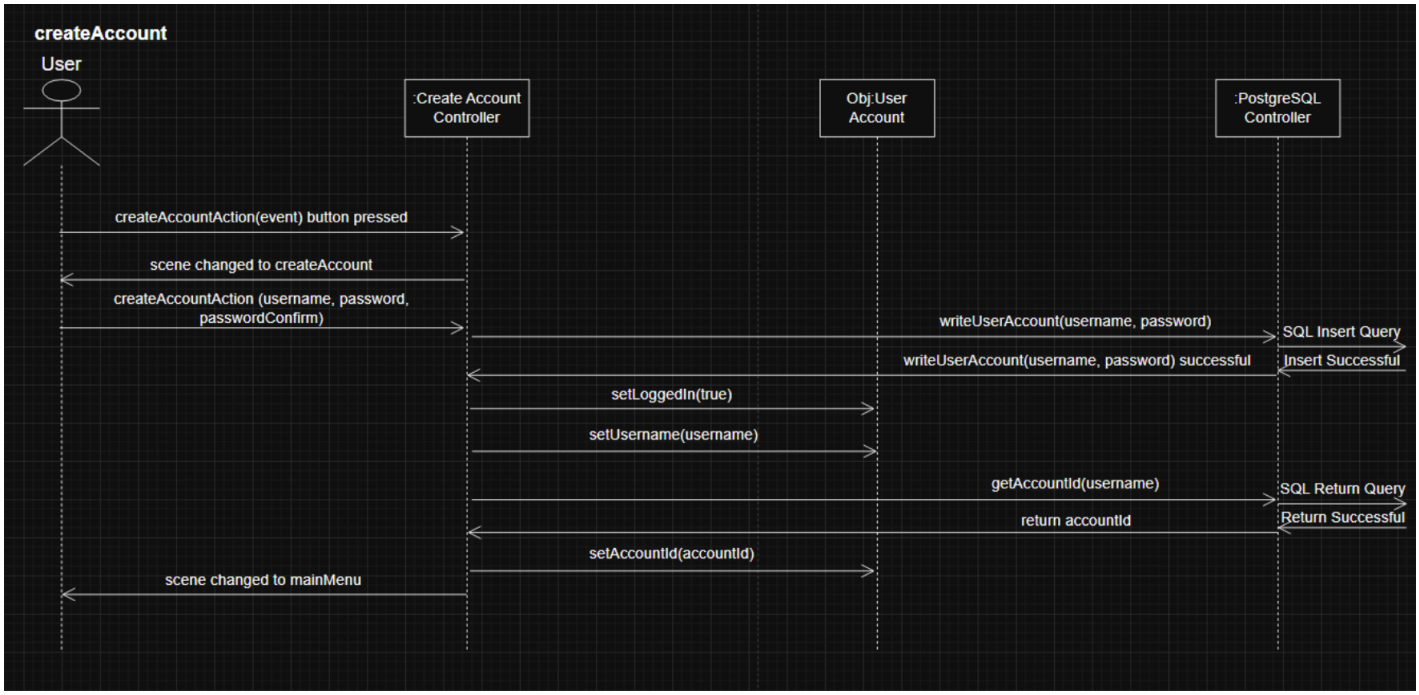


PC Build List Save/Load Activities:

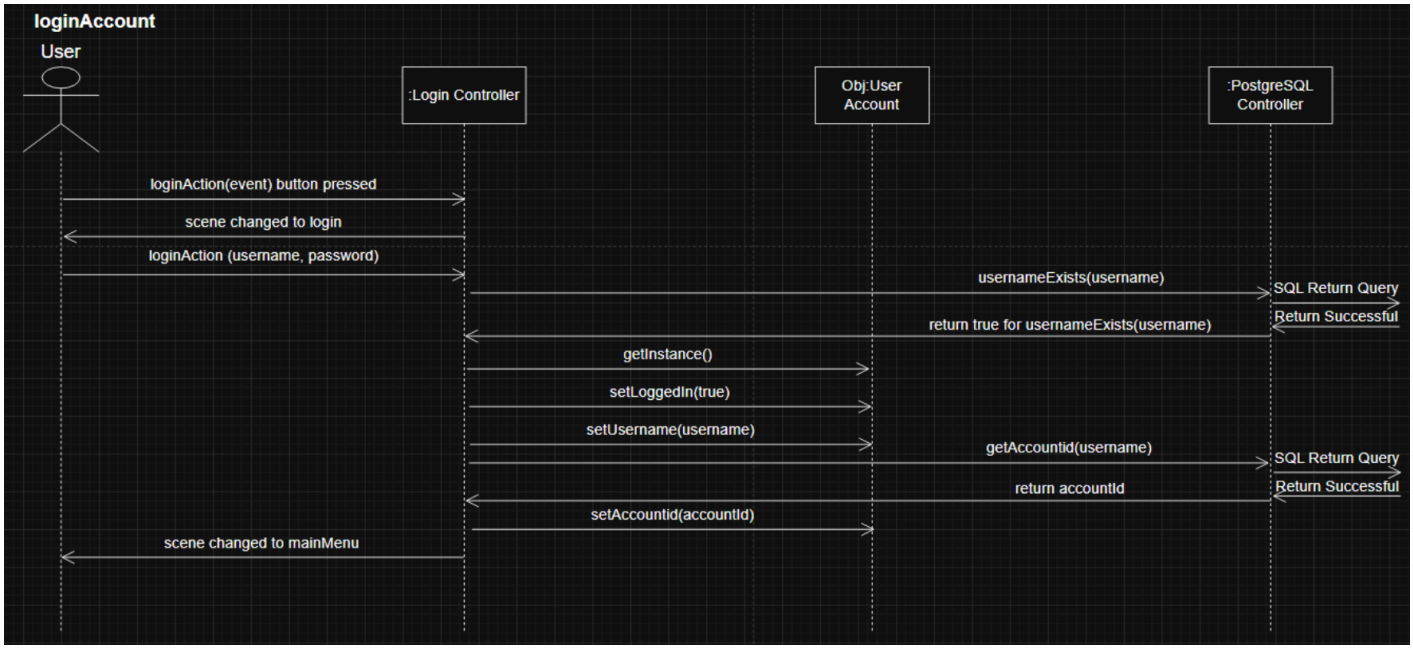


Appendix F: Sequence Diagrams with detailed design

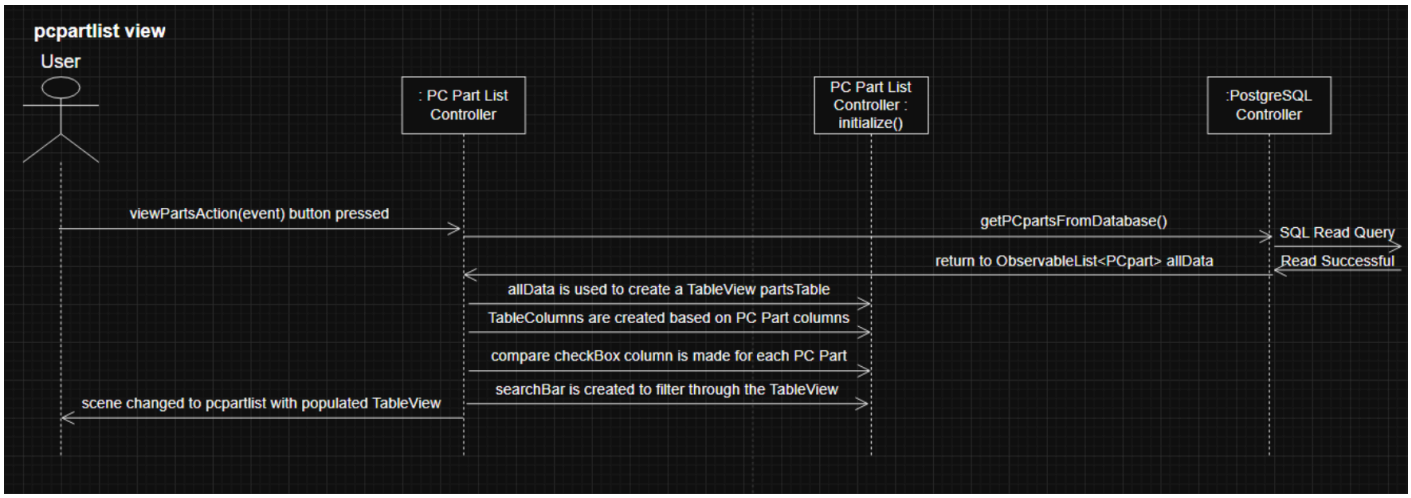
‘createAccount’ Sequence:



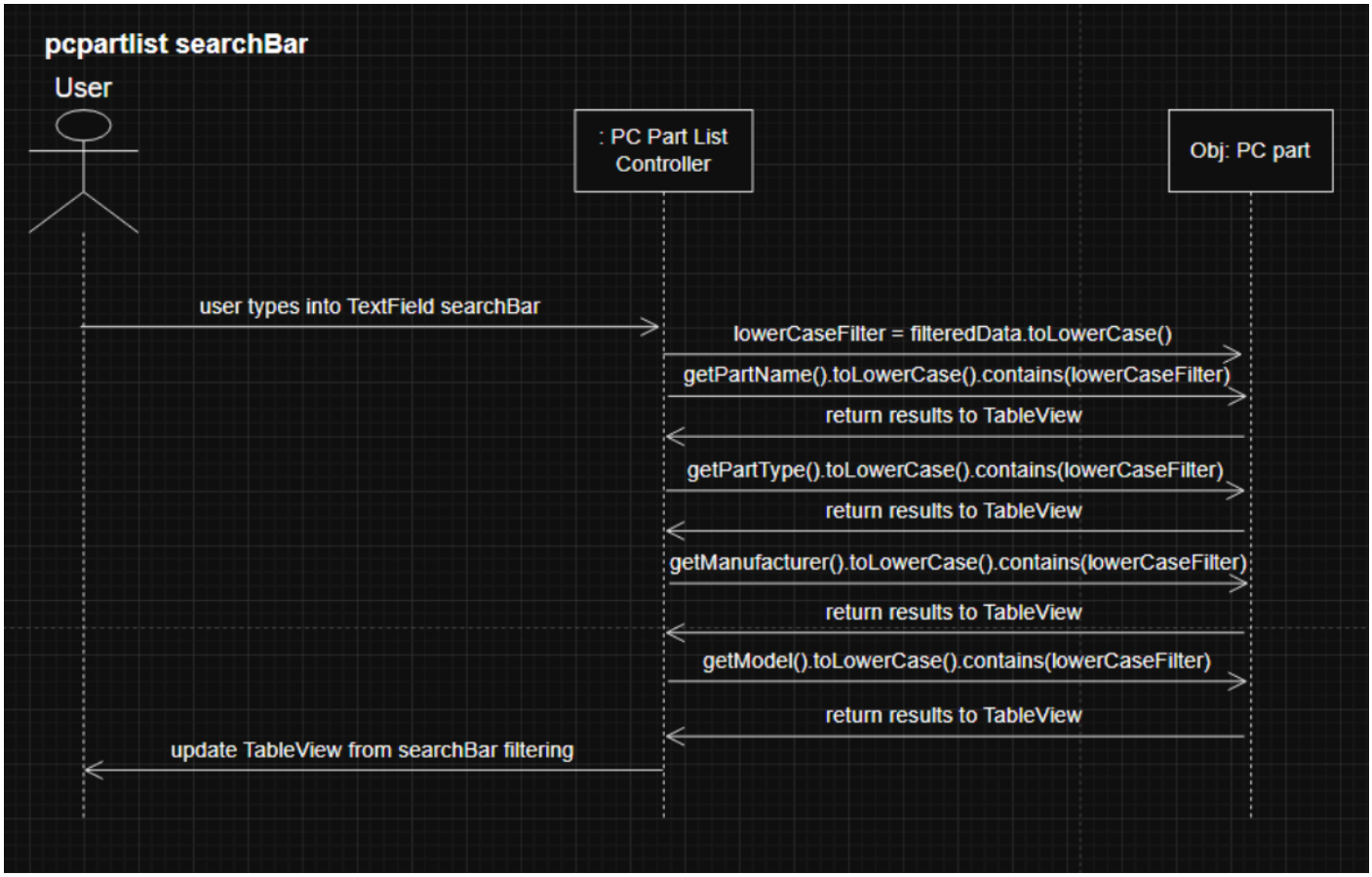
‘loginAccount’ Sequence:



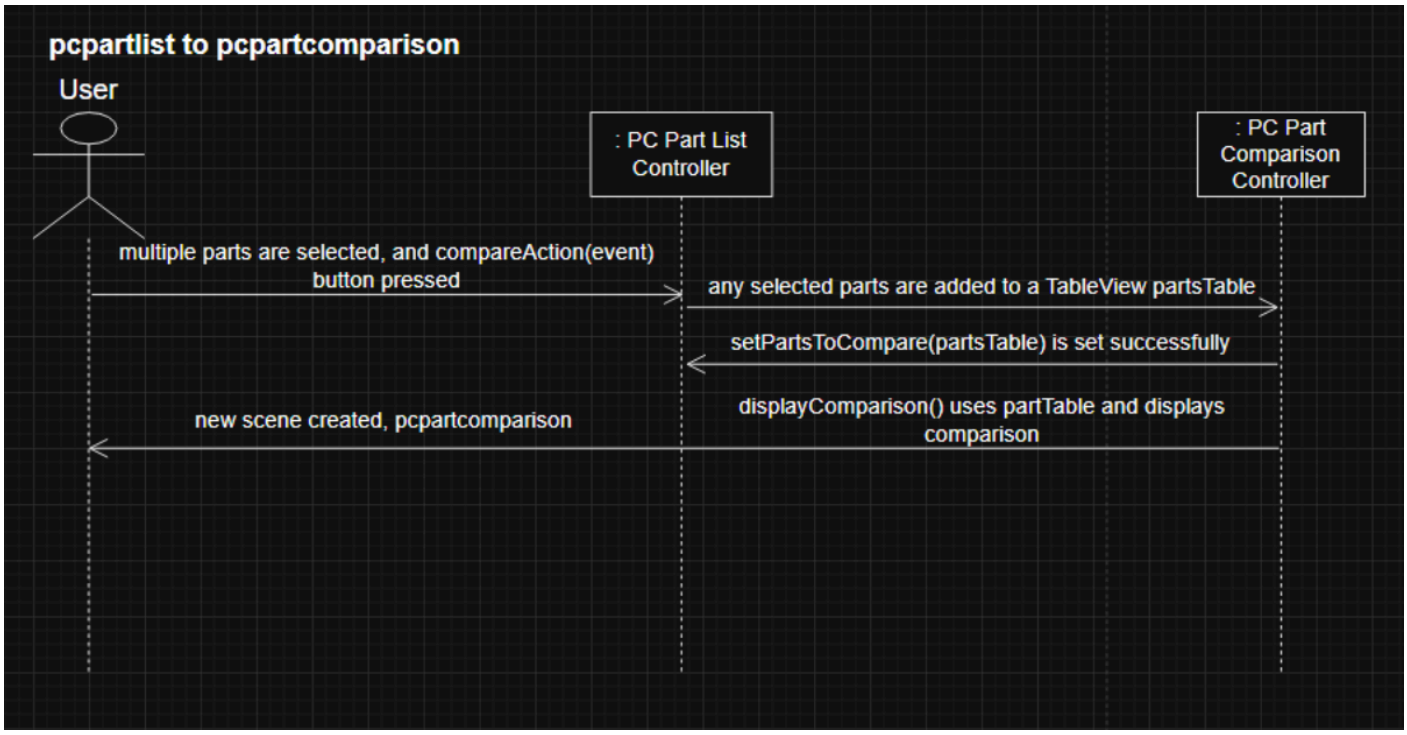
‘pcpartlist view’ Sequence:



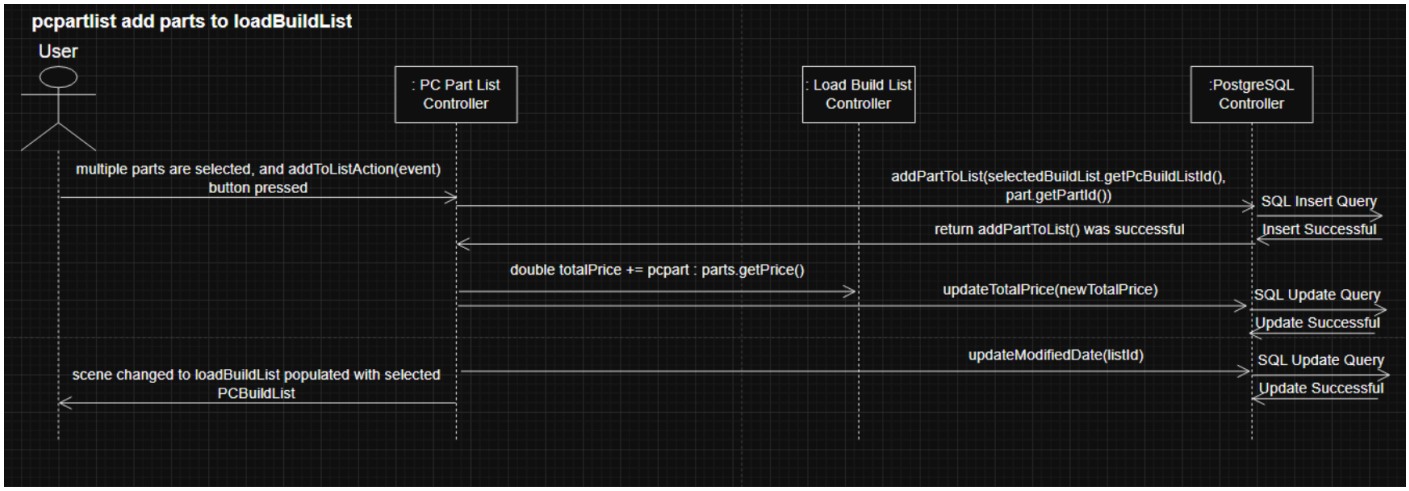
‘pcpartlist searchBar’ Sequence:



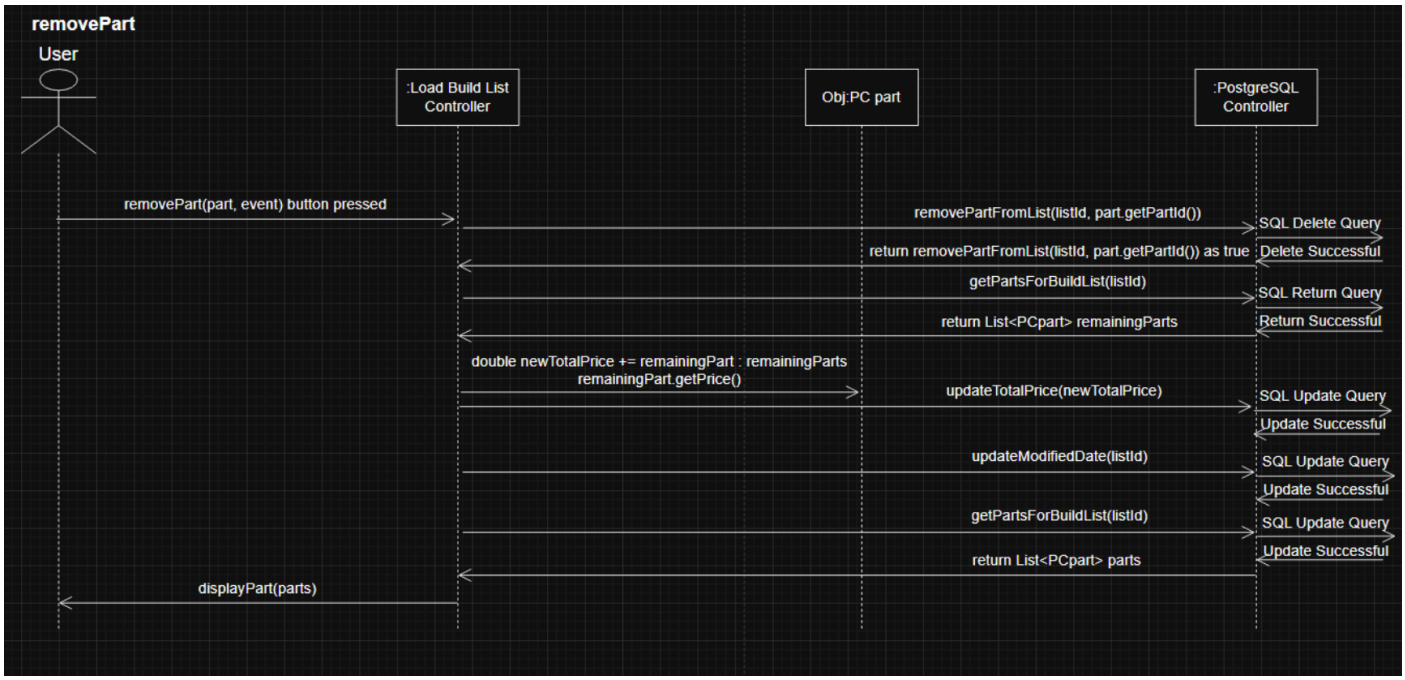
‘pcpartlist to pcpartcomparison’ Sequence:



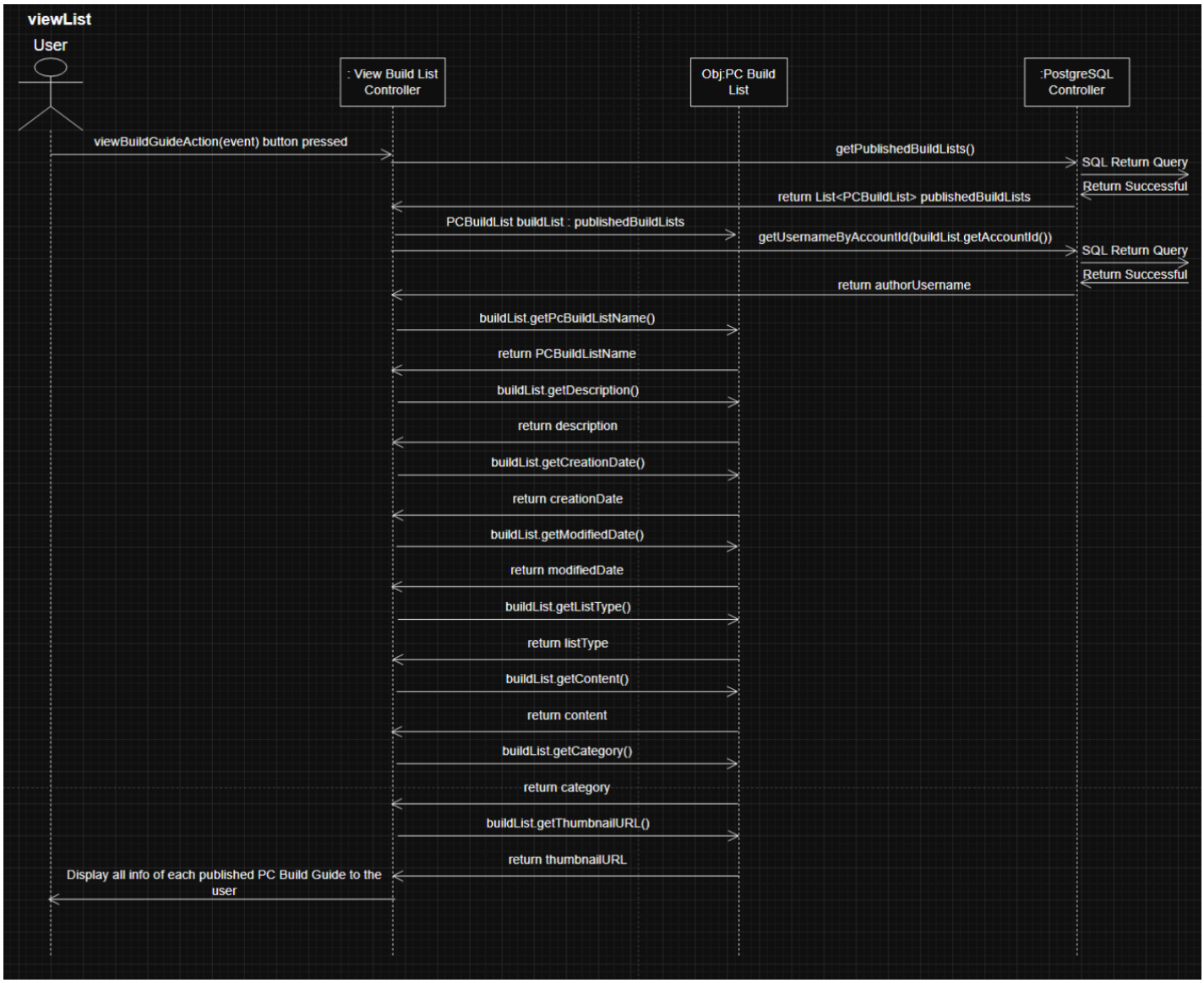
‘pcpartlist add parts to loadBuildList’ Sequence:



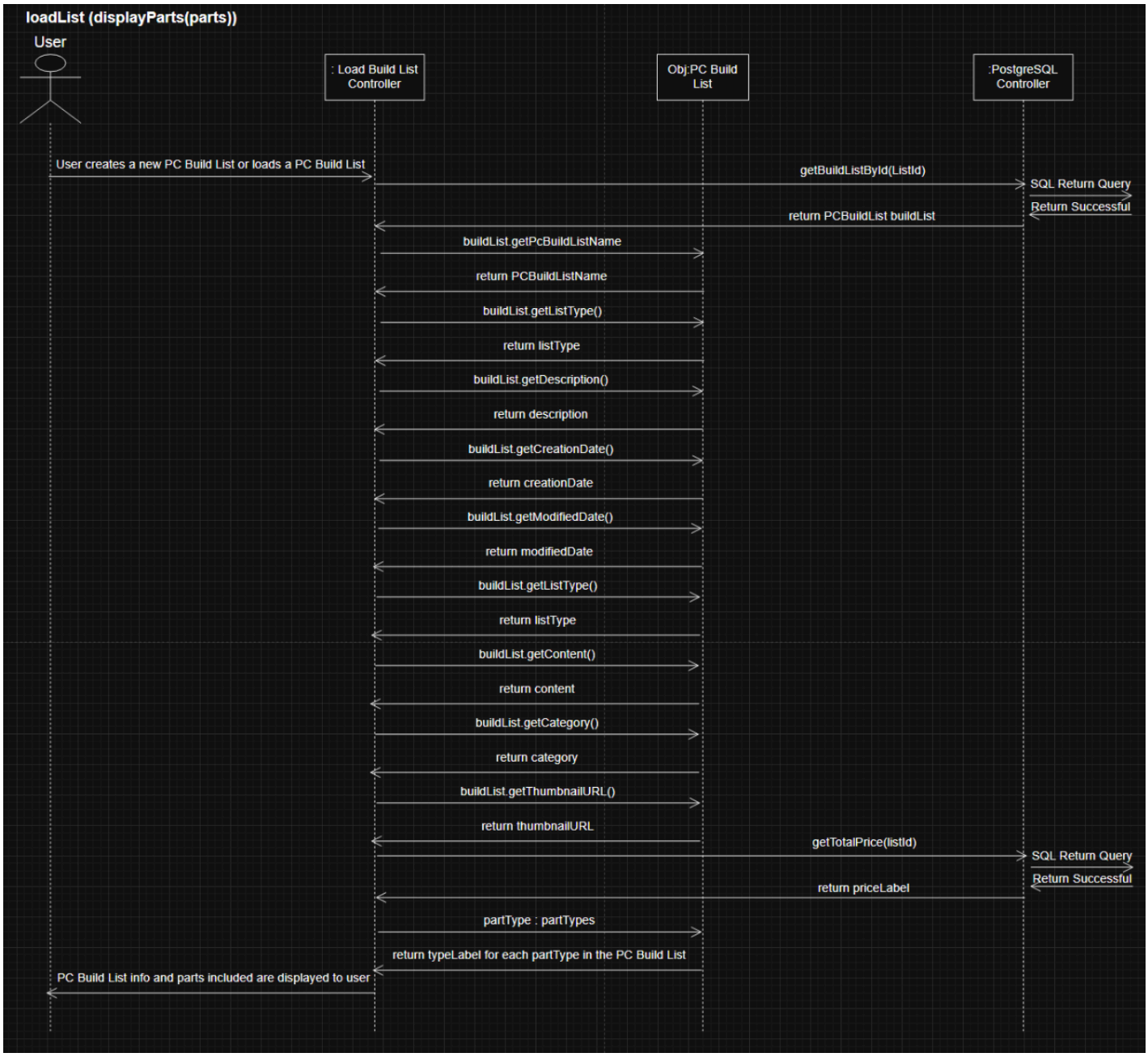
‘removePart’ Sequence:



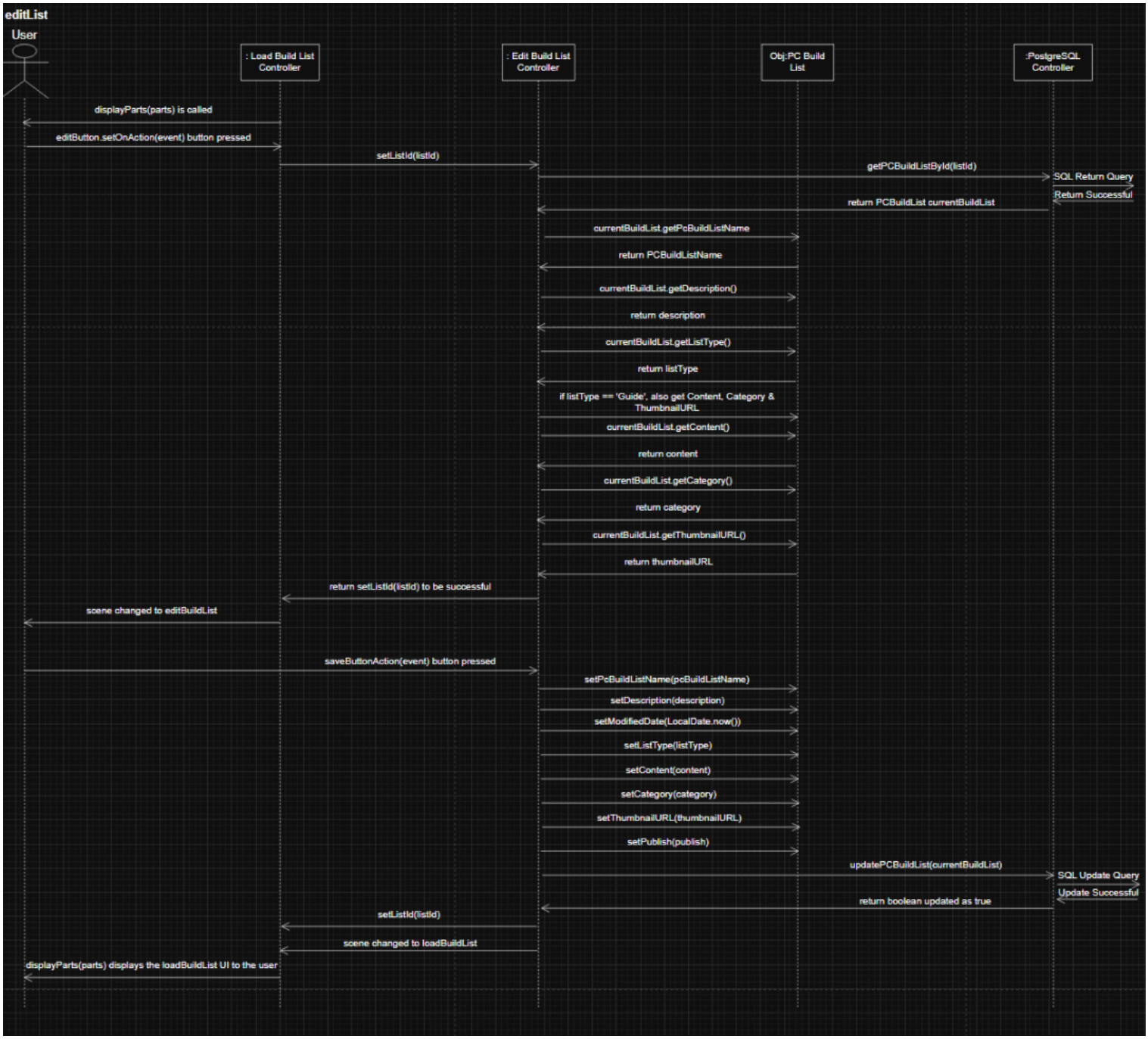
‘viewList’ Sequence:



‘loadBuildList’ Sequence:



‘editList’ Sequence:

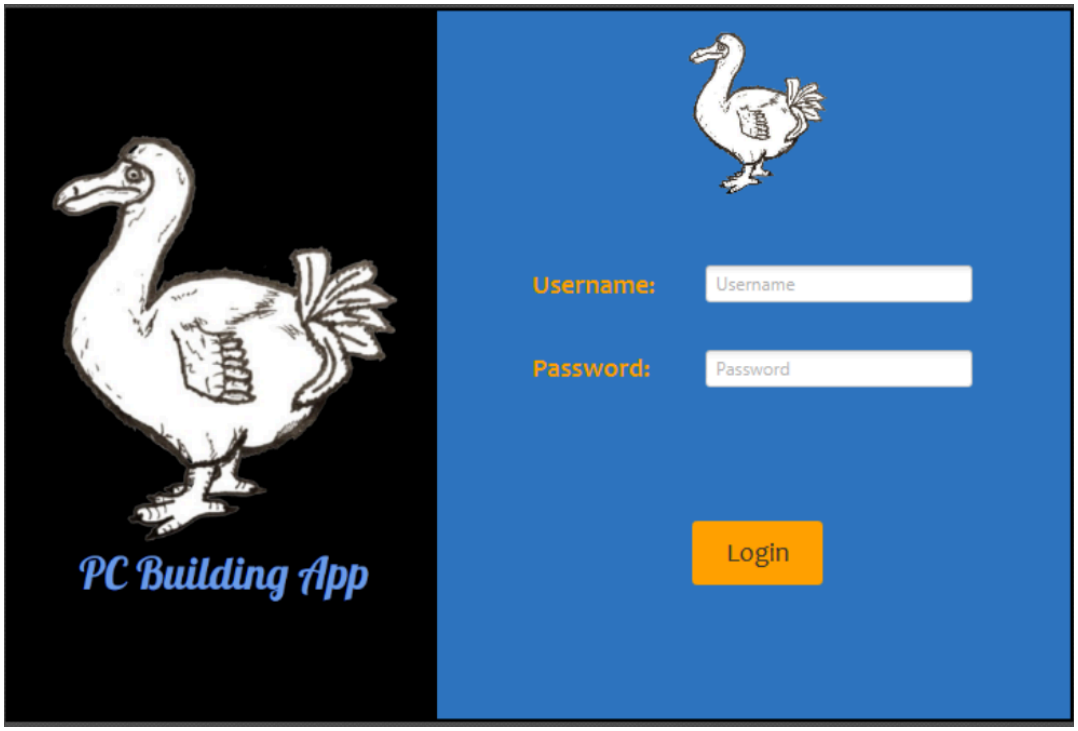


Appendix G: User Interfaces

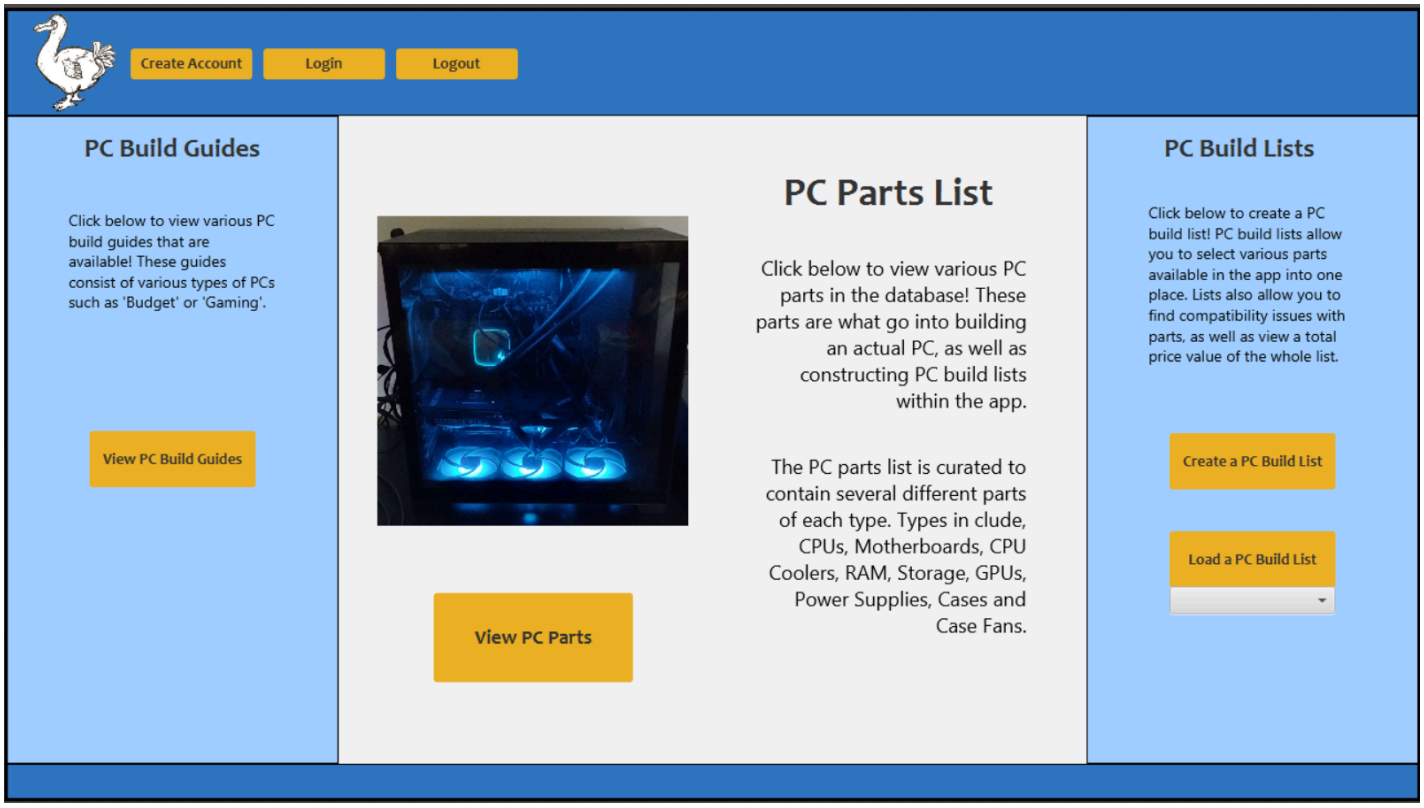
Start-up menu (start.fxml):

Create User Account menu (createAccount.fxml):

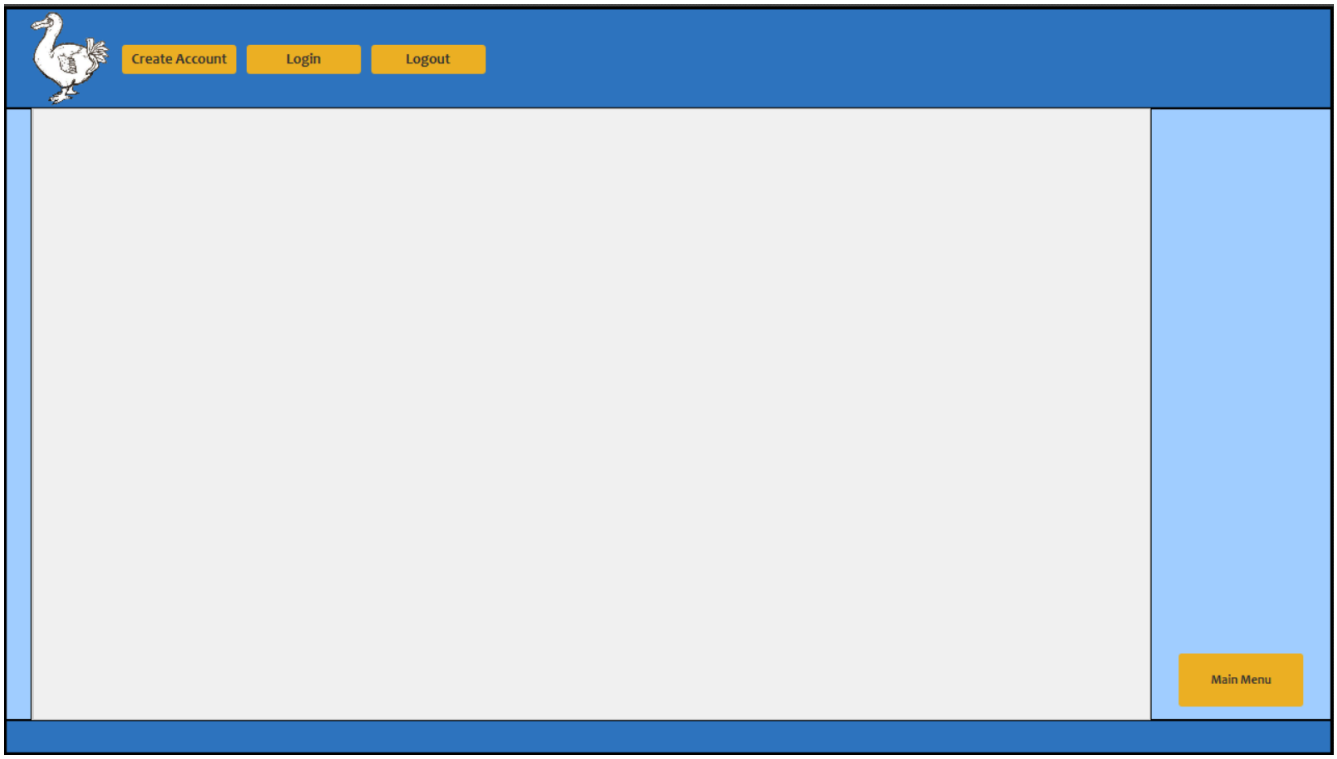
Login User Account menu (loginAccount.fxml):



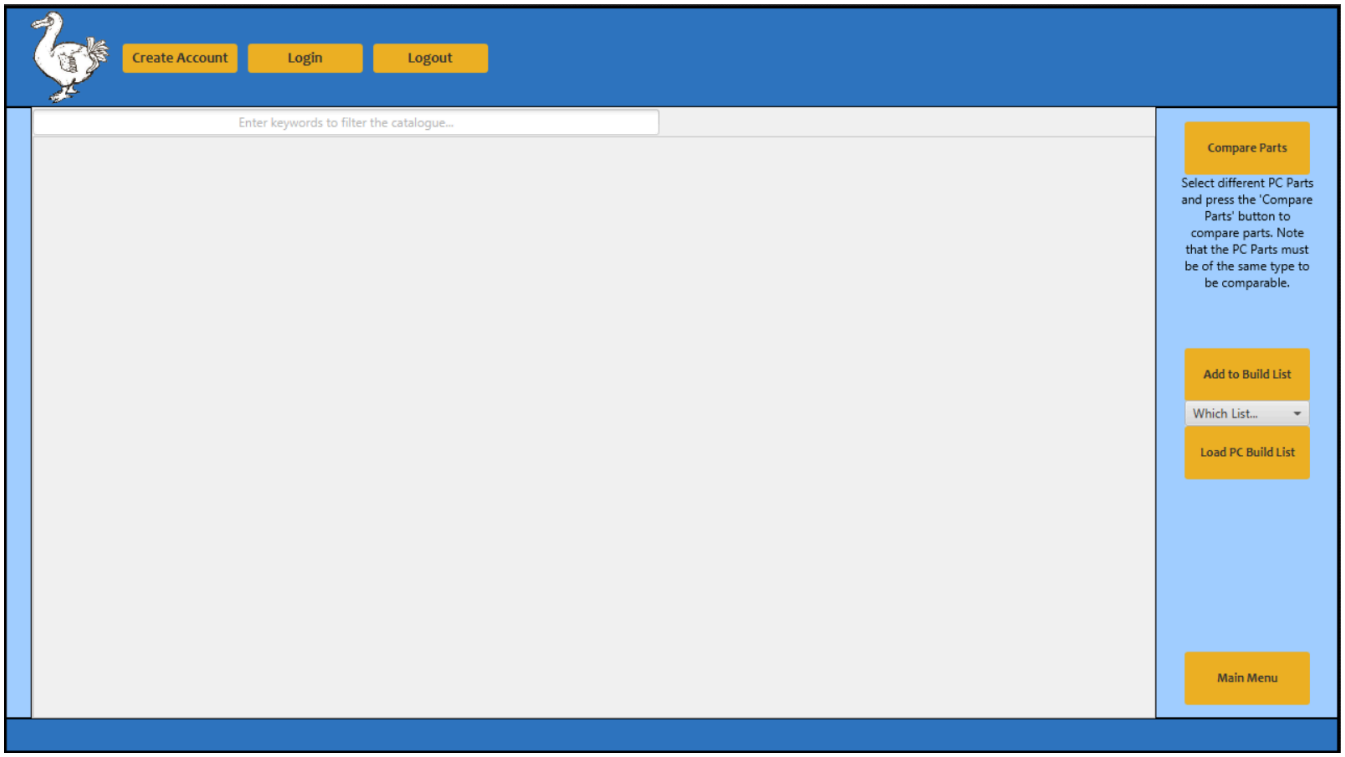
Main menu (mainMenu.fxml):



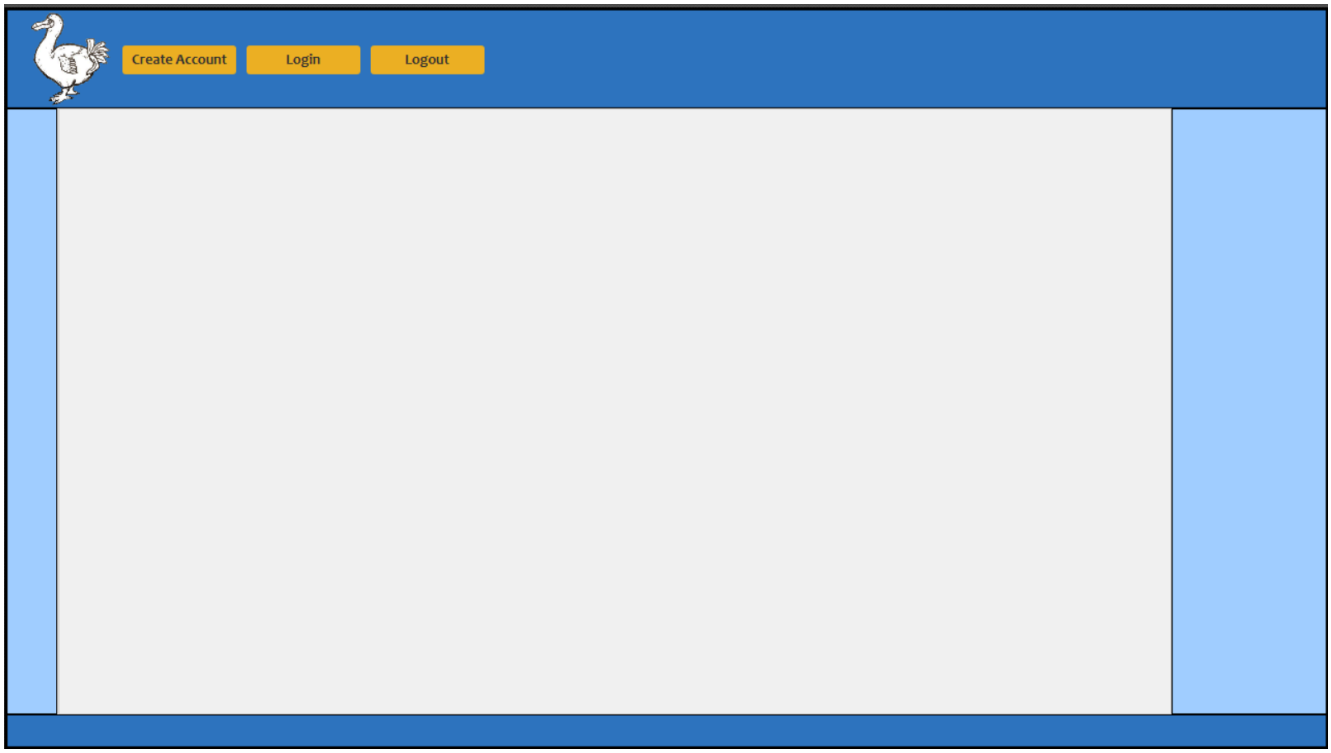
View PC Build List menu (viewBuildList.fxml):



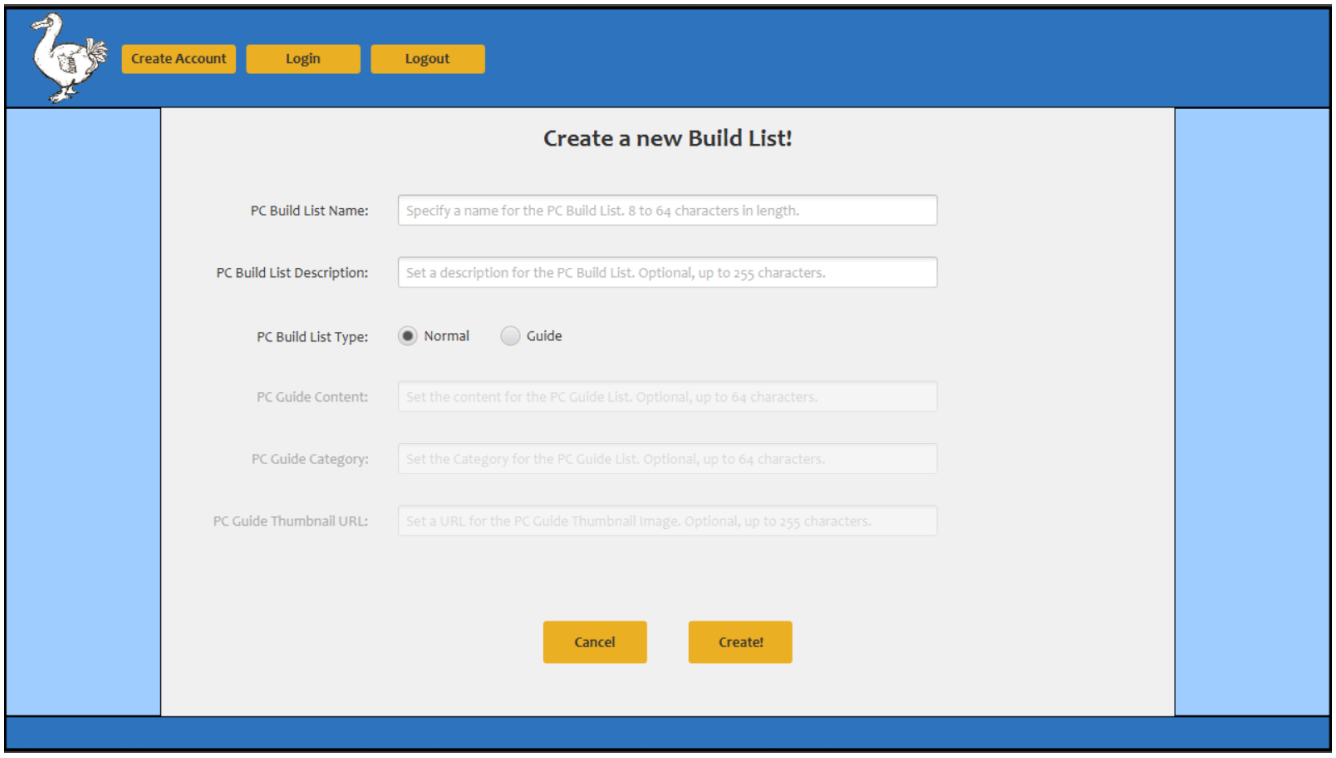
PC Part Catalogue Menu (pcpartlist.fxml):



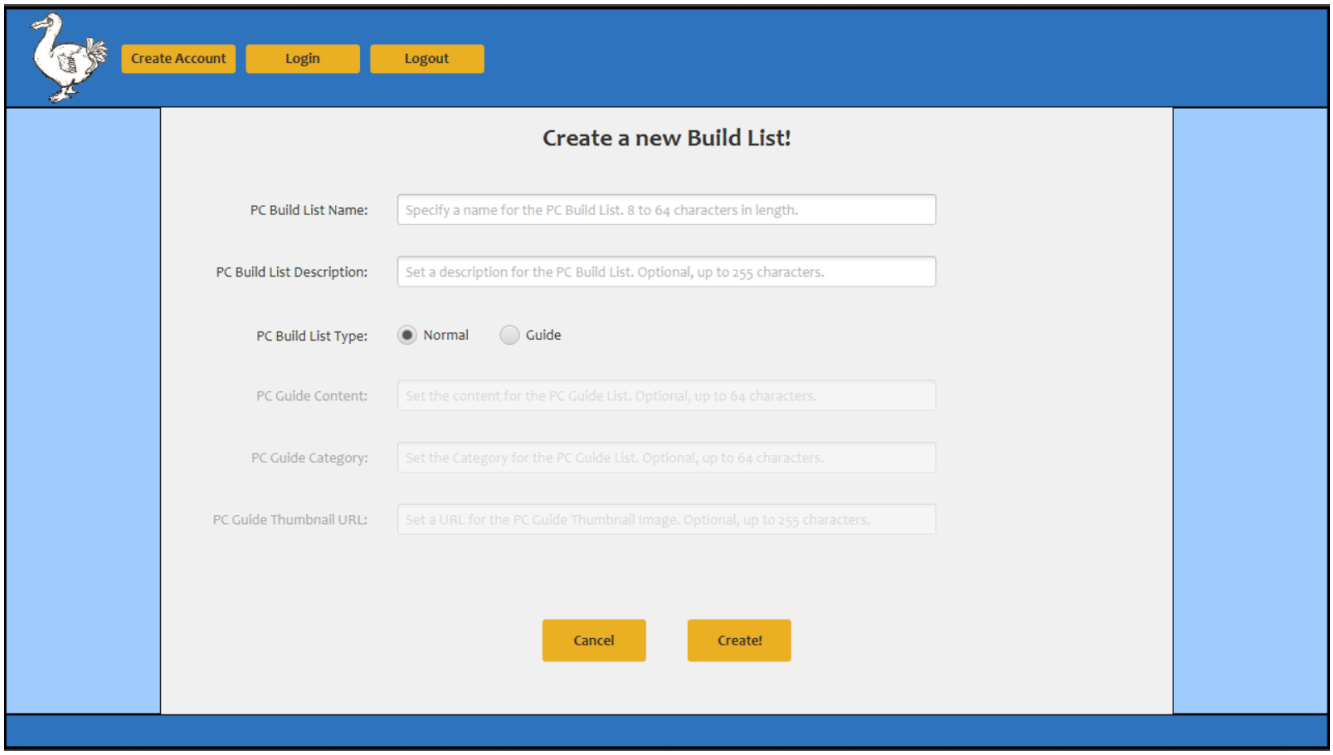
PC Part Comparison Window (pcpartcomparison.fxml):



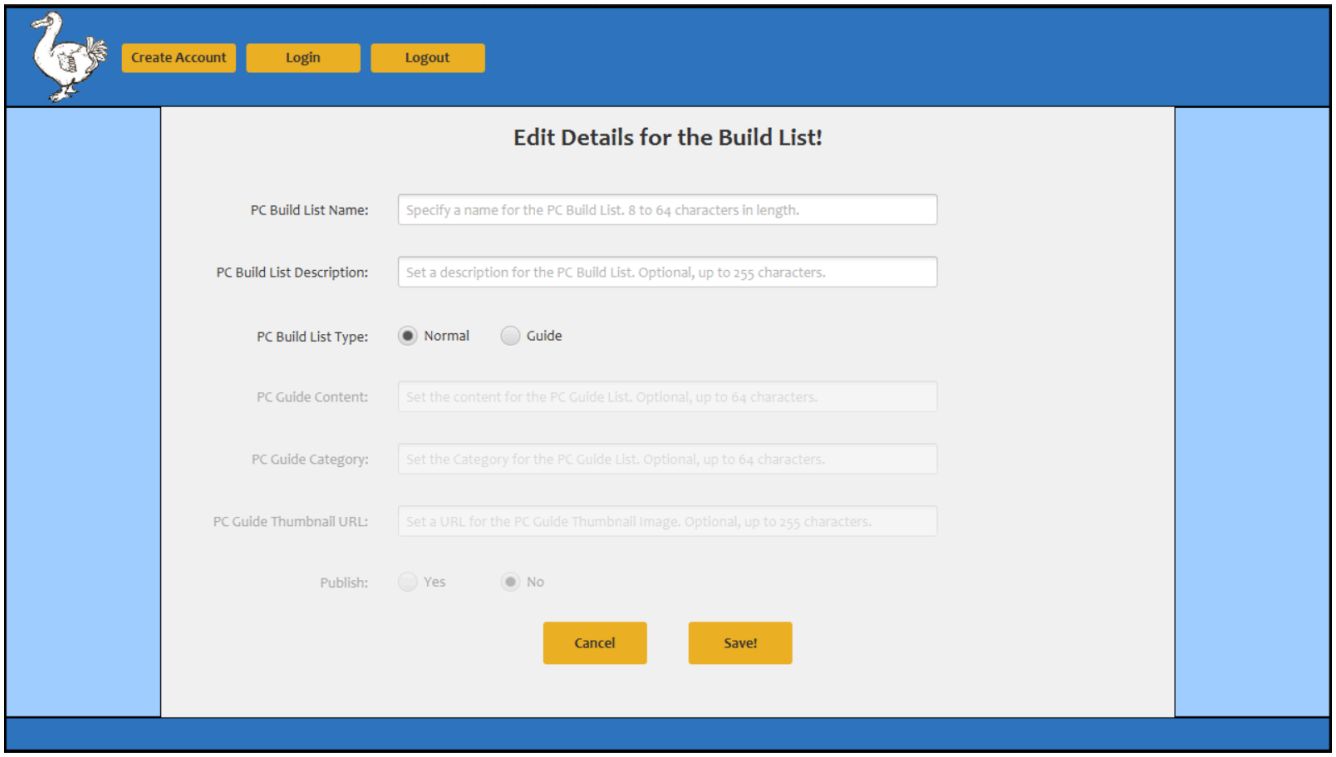
Create PC Build List menu (createBuildList.fxml):



Load PC Build List menu (loadBuildList.fxml):



Edit PC Build List menu (editBuildList.fxml):



Appendix H: Test Data

Test data for the PC Building App mainly includes all the database table entries. Some of these are difficult to display. Note that for the ‘useraccount’ database table, the ‘password’ column is an encrypted string. For the ‘pcpart’ database table, the specifications are stored as JSON files, and can be very large, so these will be displayed differently.

‘useraccount’ Database table entries:

	accountid [PK] integer	username character varying (32)	password character varying (128)
1	8	example1	6ccc5a4d4f3b83499f6ededfa95fa6369e6e05ee520380d983b0a94cb51b45ab
2	9	Keith123	3e4345b8f5d598e7c36039aeb68b410febe782e52f3192d2c517114300fdbcce
3	48	KeithH123	6c34c89b2a59b69c15cba7c9da9be77257bd6f78e32981e855380fbcf07e5974

For the ‘useraccount’ database table ‘password’ columns, the test data used within the PC Building App matches the username of the ‘useraccount’ database table entry. This means the ‘password’ to login to the account with the username ‘Keith123’ should simply be ‘Keith123’. This should hold true for all three of the test entries in the ‘useraccount’ database table.

‘pcpart’ Database table entries without specifications (1 to 25, then 26 to 36):

	partid [PK] integer	partname character varying (255)	parttype character varying (64)	manufacturer character varying (64)	model character varying (255)	price numeric (7,2)
1	1	AMD Ryzen 7 9800X3D	CPU	AMD	9800X3D	479.99
2	2	Intel Core i9-14900K	CPU	Intel	14900K	445.00
3	3	AMD Ryzen 7 7800X3D	CPU	AMD	7800X3D	449.99
4	4	Intel Core i7-13700K	CPU	Intel	13700K	378.00
5	5	Gigabyte Z790 AORUS PRO X	Motherboard	Gigabyte	Z790 AORUS PRO X	249.99
6	6	MSI MAG X870 TOMAHAWK	Motherboard	MSI	MAG X870 TOMAHAWK	279.99
7	7	Asus ROG STRIX B760-A GAMING	Motherboard	Asus	ROG STRIX B760-A GAMING	179.99
8	8	Asus TUF GAMING B650-PLUS	Motherboard	Asus	TUF GAMING B650-PLUS	169.99
9	9	Noctua NH-D15	CPU Cooler	Noctua	NH-D15	109.99
10	10	Corsair iCUE H150i ELITE CAPELLIX	CPU Cooler	Corsair	iCUE H150i ELITE CAPELLIX	398.00
11	11	Thermalright Peerless Assassin	CPU Cooler	Thermalright	Peerless Assassin	42.99
12	12	ARTIC Liquid Freezer III	CPU Cooler	ARCTIC	Liquid Freezer III	139.99
13	13	Corsair Vengeance RGB 32GB	RAM	Corsair	Vengeance RGB 32GB (2 x 16 GB)	89.99
14	14	Corsair Vengeance LPX 16GB (2 x 8 GB)	RAM	Corsair	Vengeance LPX 16GB (2 x 8 GB)	39.99
15	15	G.Skill Trident Z5 RGB 64GB (2 x 32 GB)	RAM	G.Skill	Trident Z5 RGB 64GB (2 x 32 GB)	209.99
16	16	G.Skill Ripjaws V 32GB (2 x 16 GB)	RAM	G.Skill	Ripjaws V 32GB (2 x 16 GB)	47.99
17	17	Samsung 990 Pro 2TB	Storage	Samsung	990 Pro 2TB	169.99
18	18	Crucial P3 Plus 1TB	Storage	Curcial	P3 Plus 1TB	59.99
19	19	Seagate Barracuda Compute 2TB	Storage	Seagate	Barracuda Compute 2TB	64.99
20	20	Western Digital WD_Black SN850X 2TB	Storage	Western Digital	WD_Black SN850X 2TB	147.00
21	21	MSI Ventus 2X BLACK OC GeForce RTX 4060	GPU	MSI	Ventus 2X BLACK OC	329.99
22	22	Asus ROG STRIX GAMING OC GeForce RTX 4090	GPU	Asus	ROG STRIX GAMING OC	3799.99
23	23	Gigabyte WINDFORCE V2 GeForce RTX 4080 SUPER	GPU	Gigabyte	WINDFORCE V2	1799.99
24	24	Asus TUF GAMING OC Radeon RX 7900 XT	GPU	Asus	TUF GAMING OC	799.99
25	25	Corsair RM850e (2023) 850W	Power Supply	Corsair	RM850e (2023)	129.99
26	26	Corsair RM1000e (2023) 1000W	Power Supply	Corsair	RM1000e (2023)	179.99
27	27	Cooler Master V Platinum V2 1600W	Power Supply	Cooler Master	V Platinum V2 1600W	309.99
28	28	MSI MAG A650GL 650W	Power Supply	MSI	MAG A650GL	86.99
29	29	Corsair 4000D Airflow	Case	Corsair	4000D Airflow	84.99
30	30	NZXT H9 Flow	Case	NZXT	H9 Flow	154.99
31	31	Fractal Design North XL	Case	Fractal Design	North XL	179.99
32	32	Azza Sanctum	Case	Azza	Sanctum	349.99
33	33	Thermalright TL-C12C-S X3	Case Fan	Thermalright	TL-C12C-S X3	13.59
34	34	Lian Li Uni Fan SL-Infinity	Case Fan	Lian Li	Uni Fan SL-Infinity	89.99
35	35	Corsair iCUE LINK QX120 RGB Starter Kit	Case Fan	Corsair	iCUE LINK QX120 RGB Starter Kit	129.00
36	36	Noctua NF-A12x25 PWM chromax.black.swap	Case Fan	Noctua	NF-A12x25 PWM chromax.black.swap	34.00

‘pcpart’ Database table entries ‘specifications’ columns will be displayed using the query statements used to create them:

```
-- CPUs (partID 1, 2, 3, 4) - All specifications
UPDATE pcbuildingappschema.pcpart
SET specifications = CASE
WHEN partID = 1 THEN '{
       "cores": 8,
       "threads": 16,
       "base_clock": "4.7 GHz",
       "boost_clock": "5.8 GHz",
       "cache": "128MB",
       "socket": "AM5",
       "tdp": "120W",
       "integrated_graphics": false,
       "instruction_set": "x86-64",
       "max_memory_speed": "5200"
       }'::JSONB
WHEN partID = 2 THEN '{
       "cores": 24,
       "threads": 32,
       "base_clock": "3.2 GHz",
       "boost_clock": "6.0 GHz",
       "cache": "36MB",
       "socket": "LGA 1700",
       "tdp": "125W",
       "integrated_graphics": true,
       "instruction_set": "x86-64",
       "max_memory_speed": "5600"
```

```
        }'::JSONB
WHEN partID = 3 THEN '{
        "cores": 8,
        "threads": 16,
        "base_clock": "4.2 GHz",
        "boost_clock": "5.0 GHz",
        "cache": "96MB",
        "socket": "AM5",
        "tdp": "120W",
        "integrated_graphics": false,
        "instruction_set": "x86-64",
        "max_memory_speed": "5200"
        }'::JSONB
WHEN partID = 4 THEN '{
        "cores": 16,
        "threads": 24,
        "base_clock": "3.4 GHz",
        "boost_clock": "5.4 GHz",
        "cache": "30MB",
        "socket": "LGA 1700",
        "tdp": "125W",
        "integrated_graphics": true,
        "instruction_set": "x86-64",
        "max_memory_speed": "5600"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (1, 2, 3, 4);
```

```
-- Motherboards (partID 5, 6, 7, 8) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 5 THEN '{

       "socket": "LGA 1700",

       "chipset": "Z790",

       "form_factor": "ATX",

       "memory_slots": 4,

       "max_memory": "192GB",

       "pcie_slots": "1 x PCIe 5.0 x16, 4 x PCIe 4.0 x16 (x4 mode)",

       "m2_slots": 5,

       "wifi": "Wi-Fi 7",

       "ethernet": "2.5GbE",

       "memory_type": "DDR5",

       "memory_speed_support": "8000",

       "sata_port_amount": 6,

       "usb_port_amount": "1 x USB 3.2 Gen 2x2 Type-C, 10 x USB 3.2 Gen 2

Type-A, 4 x USB 2.0",

       "audio_codec": "Realtek ALC4080",

       "pcie_version": "PCIe 5.0",

       "motherboard_dimensions": "30.5cm x 24.4cm",

       "vrm_quality": "20+1+2 Phases"

       }'::JSONB

WHEN partID = 6 THEN '{

       "socket": "AM5",

       "chipset": "X670E",

       "form_factor": "ATX",

       "memory_slots": 4,

       "max_memory": "192GB",
```

```
        "pcie_slots": "1 x PCIe 5.0 x16, 2 x PCIe 4.0 x16 (x4 mode)",
        "m2_slots": 4,
        "wifi": "Wi-Fi 7",
        "ethernet": "2.5GbE",
        "memory_type": "DDR5",
        "memory_speed_support": "8000",
        "sata_port_amount": 6,
        "usb_port_amount": "1 x USB 3.2 Gen 2x2 Type-C, 8 x USB 3.2 Gen 2
Type-A, 4 x USB 2.0",
        "audio_codec": "Realtek ALC4080",
        "pcie_version": "PCIe 5.0",
        "motherboard_dimensions": "30.5cm x 24.4cm",
        "vrm_quality": "18+2+2 Phases"
        }'::JSONB
WHEN partID = 7 THEN '{
        "socket": "LGA 1700",
        "chipset": "B760",
        "form_factor": "ATX",
        "memory_slots": 4,
        "max_memory": "128GB",
        "pcie_slots": "1 x PCIe 5.0 x16, 3 x PCIe 4.0 x16 (x4 mode)",
        "m2_slots": 3,
        "wifi": "Wi-Fi 6E",
        "ethernet": "2.5GbE",
        "memory_type": "DDR5",
        "memory_speed_support": "7200",
        "sata_port_amount": 4,
        "usb_port_amount": "1 x USB 3.2 Gen 2x2 Type-C, 6 x USB 3.2 Gen 2
Type-A, 4 x USB 2.0",
```

```
        "audio_codec": "Realtek ALC897",
        "pcie_version": "PCIe 5.0",
        "motherboard_dimensions": "30.5cm x 24.4cm",
        "vrm_quality": "12+1+1 Phases"
        }'::JSONB
WHEN partID = 8 THEN '{
        "socket": "AM5",
        "chipset": "B650",
        "form_factor": "ATX",
        "memory_slots": 4,
        "max_memory": "128GB",
        "pcie_slots": "1 x PCIe 4.0 x16, 3 x PCIe 4.0 x16 (x4 mode)",
        "m2_slots": 2,
        "wifi": "Wi-Fi 6E",
        "ethernet": "1GbE",
        "memory_type": "DDR5",
        "memory_speed_support": "6400",
        "sata_port_amount": 4,
        "usb_port_amount": "1 x USB 3.2 Gen 2x2 Type-C, 6 x USB 3.2 Gen 2
Type-A, 4 x USB 2.0",
        "audio_codec": "Realtek ALC897",
        "pcie_version": "PCIe 4.0",
        "motherboard_dimensions": "30.5cm x 24.4cm",
        "vrm_quality": "14+2+1 Phases"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (5, 6, 7, 8);
```

```
-- CPU Coolers (partID 9, 10, 11, 12) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 9 THEN '{

       "type": "Air Cooler",

       "socket_compatibility": "LGA 1700, AM5, LGA1200, LGA115x, AM4",

       "fan_size": "2 x 140mm",

       "tdp_rating": "220W+",

       "noise_level": "22.8 dBA (Max)",

       "cooler_height": "165mm",

       "fan_speed_range": "300-1500 RPM",

       "cooler_dimensions": "150 x 161 x 165 mm"

       }'::JSONB

WHEN partID = 10 THEN '{

       "type": "Liquid Cooler (AIO)",

       "radiator_size": "360mm",

       "socket_compatibility": "LGA 1700, AM5, LGA1200, LGA115x, AM4, sTRX4,

sTR4",

       "fan_size": "3 x 120mm",

       "tdp_rating": "300W+",

       "noise_level": "36 dBA (Max)",

       "rgb": true,

       "pump_speed": "2800 RPM",

       "fan_speed_range": "400-2400 RPM",

       "cooler_dimensions": "397 x 120 x 27 mm (Radiator)"

       }'::JSONB

WHEN partID = 11 THEN '{

       "type": "Air Cooler",

       "socket_compatibility": "LGA 1700, AM5, LGA1200, LGA115x, AM4",
```

```
        "fan_size": "2 x 120mm",
        "tdp_rating": "265W",
        "noise_level": "25.6 dBA (Max)",
        "fan_speed_range": "600-1550 RPM",
        "cooler_height": "155mm",
        "cooler_dimensions": "125 x 135 x 157 mm"
        }'::JSONB
WHEN partID = 12 THEN '{
        "type": "Liquid Cooler (AIO)",
        "radiator_size": "360mm",
        "socket_compatibility": "LGA 1700, AM5, LGA1200, LGA115x, AM4",
        "fan_size": "3 x 120mm",
        "tdp_rating": "300W+",
        "pump_speed": "2800 RPM",
        "fan_speed_range": "200-1700 RPM",
        "cooler_dimensions": "398 x 120 x 38 mm (Radiator)"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (9, 10, 11, 12);
```

```
-- RAMs (partID 13, 14, 15, 16) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 13 THEN '{

       "type": "DDR5",

       "capacity": "32GB",

       "speed": "6000",

       "cas_latency": "CL36",

       "sticks": 2,

       "stick_capacity": "16GB",

       "rgb": true,

       "voltage": "1.35V",

       "ram_dimensions": "135 x 7 x 35 mm"

       }'::JSONB

WHEN partID = 14 THEN '{

       "type": "DDR4",

       "capacity": "16GB",

       "speed": "3200",

       "cas_latency": "CL16",

       "sticks": 2,

       "stick_capacity": "8GB",

       "rgb": false,

       "voltage": "1.35V",

       "ram_dimensions": "135 x 7 x 31.25 mm"

       }'::JSONB

WHEN partID = 15 THEN '{

       "type": "DDR5",

       "capacity": "64GB",

       "speed": "6400",
```

```
       "cas_latency": "CL32",
       "sticks": 2,
       "stick_capacity": "32GB",
       "rgb": true,
       "voltage": "1.40V",
       "ram_dimensions": "133.35 x 8 x 43.5 mm"
       }'::JSONB
WHEN partID = 16 THEN '{
       "type": "DDR4",
       "capacity": "32GB",
       "speed": "3200",
       "cas_latency": "CL16",
       "sticks": 2,
       "stick_capacity": "16GB",
       "rgb": false,
       "voltage": "1.35V",
       "ram_dimensions": "135 x 7 x 33 mm"
       }'::JSONB
ELSE specifications
END
WHERE partID IN (13, 14, 15, 16);
```

```
-- Storages (partID 17, 18, 19, 20) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 17 THEN '{

       "type": "SSD (NVMe)",

       "capacity": "2TB",

       "form_factor": "M.2-2280",

       "interface": "PCIe 4.0 x4",

       "technology": "NVMe",

       "read_speed": "7450 MB/s",

       "write_speed": "6900 MB/s",

       "nvme_version": "NVMe 2.0"

       }'::JSONB

WHEN partID = 18 THEN '{

       "type": "SSD (NVMe)",

       "capacity": "1TB",

       "form_factor": "M.2-2280",

       "interface": "PCIe 4.0 x4",

       "technology": "NVMe",

       "read_speed": "5000 MB/s",

       "write_speed": "3600 MB/s",

       "nvme_version": "NVMe 1.4"

       }'::JSONB

WHEN partID = 19 THEN '{

       "type": "HDD",

       "capacity": "2TB",

       "form_factor": "3.5\"",

       "rpm": 7200,

       "interface": "SATA III",
```

```
        "read_speed": "210 MB/s",
        "write_speed": "210 MB/s"
        }'::JSONB
WHEN partID = 20 THEN '{
        "type": "SSD (NVMe)",
        "capacity": "2TB",
        "form_factor": "M.2-2280",
        "interface": "PCIe 4.0 x4",
        "technology": "NVMe",
        "read_speed": "7300 MB/s",
        "write_speed": "6600 MB/s",
        "nvme_version": "NVMe 2.0"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (17, 18, 19, 20);
```

```
-- GPUs (partID 21, 22, 23, 24) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 21 THEN '{

       "chipset": "GeForce RTX 4060",

       "vram": "8GB",

       "memory_type": "GDDR6",

       "interface": "PCIe 4.0 x8",

       "clock_speed": "OC Edition",

       "cooling": "Dual Fan",

       "gpu_length": "199mm",

       "gpu_width": "2 Slots",

       "gpu_height": "120mm",

       "power_connectors": "1 x 8-pin",

       "gpu_dimensions": "199 x 120 x 41 mm"

       }'::JSONB

WHEN partID = 22 THEN '{

       "chipset": "GeForce RTX 4090",

       "vram": "24GB",

       "memory_type": "GDDR6X",

       "interface": "PCIe 4.0 x16",

       "clock_speed": "OC Edition",

       "cooling": "Triple Fan",

       "gpu_length": "357.6mm",

       "gpu_width": "3.5 Slots",

       "gpu_height": "149.3mm",

       "power_connectors": "1 x 16-pin (12VHPWR)",

       "gpu_dimensions": "357.6 x 149.3 x 70.1 mm"

       }'::JSONB
```

```
WHEN partID = 23 THEN '{
       "chipset": "GeForce RTX 4080 SUPER",
       "vram": "16GB",
       "memory_type": "GDDR6X",
       "interface": "PCIe 4.0 x16",
       "clock_speed": "Standard",
       "cooling": "Triple Fan",
       "gpu_length": "330mm",
       "gpu_width": "3 Slots",
       "gpu_height": "136mm",
       "power_connectors": "1 x 16-pin (12VHPWR)",
       "gpu_dimensions": "330 x 136 x 57 mm"
       }'::JSONB
WHEN partID = 24 THEN '{
       "chipset": "Radeon RX 7900 XT",
       "vram": "20GB",
       "memory_type": "GDDR6",
       "interface": "PCIe 4.0 x16",
       "clock_speed": "OC Edition",
       "cooling": "Triple Fan",
       "gpu_length": "320mm",
       "gpu_width": "2.96 Slots",
       "gpu_height": "135mm",
       "power_connectors": "2 x 8-pin",
       "gpu_dimensions": "320 x 135 x 59.5 mm"
       }'::JSONB
ELSE specifications
END
WHERE partID IN (21, 22, 23, 24);
```

```
-- Power Supplies (partID 25, 26, 27, 28) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 25 THEN '{

       "wattage": "850W",

       "efficiency": "80+ Gold",

       "modularity": "Fully Modular",

       "form_factor": "ATX",

       "pcie_connectors": "3 x 6+2-pin",

       "sata_connectors": "8",

       "atx_version": "ATX 3.0",

       "psu_dimensions": "150 x 86 x 140 mm"

       }'::JSONB

WHEN partID = 26 THEN '{

       "wattage": "1000W",

       "efficiency": "80+ Gold",

       "modularity": "Fully Modular",

       "form_factor": "ATX",

       "pcie_connectors": "3 x 6+2-pin",

       "sata_connectors": "8",

       "atx_version": "ATX 3.0",

       "psu_dimensions": "150 x 86 x 140 mm"

       }'::JSONB

WHEN partID = 27 THEN '{

       "wattage": "1600W",

       "efficiency": "80+ Platinum",

       "modularity": "Fully Modular",

       "form_factor": "ATX",

       "pcie_connectors": "6 x 6+2-pin",
```

```
        "sata_connectors": "12",
        "atx_version": "ATX 3.0",
        "psu_dimensions": "170 x 86 x 150 mm"
        }'::JSONB
WHEN partID = 28 THEN '{
        "wattage": "650W",
        "efficiency": "80+ Gold",
        "modularity": "Fully Modular",
        "form_factor": "ATX",
        "pcie_connectors": "2 x 6+2-pin",
        "sata_connectors": "8",
        "atx_version": "ATX 3.0",
        "psu_dimensions": "140 x 86 x 150 mm"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (25, 26, 27, 28);
```

```
-- Cases (partID 29, 30, 31, 32) - All specifications
UPDATE pcbuildingappschema.pcpart
SET specifications = CASE
WHEN partID = 29 THEN '{
       "type": "Mid Tower",
       "form_factor": "ATX",
       "airflow": "High",
       "material": "Steel, Tempered Glass",
       "max_gpu_length": "360mm",
       "max_cpu_cooler_height": "170mm",
       "max_psu_length": "220mm",
       "radiator_support": "Front: 360mm, Top: 280mm, Rear: 120mm",
       "drive_bays": "2 x 3.5\", 2 x 2.5\"",
       "case_dimensions": "453 x 230 x 466 mm",
       "motherboard_form_factor": "ATX, Micro-ATX, Mini-ITX",
       "front_fan_support": "3x 120mm / 2x 140mm",
       "top_fan_support": "2x 120mm / 2x 140mm",
       "rear_fan_support": "1x 120mm",
       "included_fans": "2x 120mm",
       "dust_filters": "Top, Front, Bottom",
       "fan_hub_included": false
       }'::JSONB
WHEN partID = 30 THEN '{
       "type": "Mid Tower",
       "form_factor": "ATX",
       "airflow": "High",
       "material": "Steel, Tempered Glass",
       "max_gpu_length": "435mm",
       "max_cpu_cooler_height": "165mm",
```

```
        "max_psu_length": "210mm",
        "radiator_support": "Top: 360mm, Side: 360mm, Bottom: 360mm, Rear:
120mm",
        "drive_bays": "4 x 2.5\"",
        "case_dimensions": "495 x 290 x 495 mm",
        "motherboard_form_factor": "ATX, Micro-ATX, Mini-ITX",
        "front_fan_support": "3x 120mm / 3x 140mm",
        "top_fan_support": "3x 120mm / 3x 140mm",
        "rear_fan_support": "1x 120mm / 1x 140mm",
        "side_fan_support": "3x 120mm / 3x 140mm",
        "bottom_fan_support": "3x 120mm",
        "included_fans": "0x 120/140mm",
        "dust_filters": "Top, Bottom, Side",
        "fan_hub_included": false
        }'::JSONB
WHEN partID = 31 THEN '{
        "type": "Full Tower",
        "form_factor": "ATX",
        "airflow": "High",
        "material": "Steel, Wood, Tempered Glass",
        "max_gpu_length": "413mm",
        "max_cpu_cooler_height": "185mm",
        "max_psu_length": "270mm",
        "radiator_support": "Front: 420mm, Top: 360mm, Rear: 120mm",
        "drive_bays": "2 x 3.5\", 2 x 2.5\"",
        "case_dimensions": "525 x 240 x 553 mm",
        "motherboard_form_factor": "E-ATX, ATX, Micro-ATX, Mini-ITX",
        "front_fan_support": "3x 120mm / 3x 140mm",
        "top_fan_support": "3x 120mm / 2x 140mm",
```

```
        "rear_fan_support": "1x 120mm",
        "included_fans": "2x 140mm",
        "dust_filters": "Front, Top, Bottom",
        "fan_hub_included": true,
        "fan_hub_type": "PWM"
        }'::JSONB
WHEN partID = 32 THEN '{
        "type": "Desktop",
        "form_factor": "ATX",
        "airflow": "Medium",
        "material": "Steel, Tempered Glass",
        "max_gpu_length": "360mm",
        "max_cpu_cooler_height": "160mm",
        "max_psu_length": "ATX",
        "radiator_support": "Top: 280mm",
        "drive_bays": "2 x 3.5\", 2 x 2.5\"",
        "case_dimensions": "480 x 220 x 480 mm",
        "motherboard_form_factor": "ATX, Micro-ATX, Mini-ITX",
        "front_fan_support": "2x 120mm / 2x 140mm",
        "top_fan_support": "2x 120mm / 2x 140mm",
        "rear_fan_support": "1x 120mm",
        "included_fans": "1x 120mm",
        "dust_filters": "Top, Front",
        "fan_hub_included": false
        }'::JSONB
ELSE specifications
END
WHERE partID IN (29, 30, 31, 32);
```

```
-- Case Fans (partID 33, 34, 35, 36) - All specifications

UPDATE pcbuildingappschema.pcpart

SET specifications = CASE

WHEN partID = 33 THEN '{

       "size": "120mm",

       "cfm": "66.17 CFM",

       "rgb": false,

       "quantity": 3,

       "fan_thickness": "25mm",

       "fan_connector": "4-pin PWM",

       "fan_dimensions": "120 x 120 x 25 mm",

       "fan_speed_range": "1550 RPM (Max)"

       }'::JSONB

WHEN partID = 34 THEN '{

       "size": "120mm",

       "cfm": "61.3 CFM",

       "rgb": true,

       "quantity": 3,

       "fan_thickness": "25mm",

       "fan_connector": "4-pin PWM",

       "fan_dimensions": "120 x 120 x 25 mm",

       "fan_speed_range": "2100 RPM (Max)"

       }'::JSONB

WHEN partID = 35 THEN '{

       "size": "120mm",

       "cfm": "63.1 CFM",

       "rgb": true,

       "quantity": 3,

       "fan_thickness": "25mm",
```

```
        "fan_connector": "iCUE LINK",
        "fan_dimensions": "120 x 120 x 25 mm",
        "fan_speed_range": "2400 RPM (Max)"
        }'::JSONB
WHEN partID = 36 THEN '{
        "size": "120mm",
        "cfm": "60.09 CFM",
        "rgb": false,
        "quantity": 1,
        "fan_thickness": "25mm",
        "fan_connector": "4-pin PWM",
        "fan_dimensions": "120 x 120 x 25 mm",
        "fan_speed_range": "450-2000 RPM"
        }'::JSONB
ELSE specifications
END
WHERE partID IN (33, 34, 35, 36);
```

‘pcbuildlist’ Database table entries (columns 1 to 6):

	pcbuildlistid [PK] integer	accountid integer	pcbuildlistname character varying (128)	description character varying (255)	creationdate date	modifieddate date
1	46	9	ExampleGuide	This is a very updated example.	2025-04-01	2025-04-15
2	47	9	ExampleList	This is an example	2025-04-01	2025-04-01
3	49	48	Keith's Cool List	This is a cool list!	2025-04-22	2025-04-22

‘pcbuildlist’ Database table entries (columns 7 to 12):

totalprice numeric (8,2)	listtype character varying (64)	content character varying (64)	category character varying (64)	thumbnailurl character varying (255)	publish boolean
3216.52	Guide	Guide Stuff	Cool		true
1498.51	Normal				false
1673.52	Guide				true

‘pcbuildlist_pcpart’ Database table entries:

	listpartid [PK] integer	listid integer	partid integer	quantity integer
1	128	46	5	1
2	129	46	11	1
3	130	46	15	1
4	131	46	17	1
5	132	46	23	1
6	134	46	30	1
7	137	46	25	1
8	138	47	1	1
9	139	47	5	1
10	140	47	11	1
11	141	47	14	1
12	142	47	17	1
13	148	47	28	1
14	150	47	21	1
15	151	47	29	1
16	152	47	33	1
17	156	46	33	1
18	157	46	2	1
19	160	49	9	1
20	161	49	13	1
21	162	49	17	1
22	163	49	21	1
23	164	49	26	1
24	165	49	29	1
25	166	49	33	1
26	171	49	5	1
27	173	49	2	1