

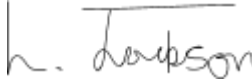


Testing Documentation

Team 7A.1

ComPChecker

<u>Name</u>	<u>Contributions</u>	<u>Signature</u>
Thomas Chate	Test Cases Stress Testing	
Lillie Hogg	Test Cases Tools used Usability Evaluation	
Luke Jackson	Test Cases Component Testing	

Test Cases

Login Test Case

This test case is testing if the user can successfully login to the system, and if upon login the user is directed to the correct menu.

Valid Inputs:	Invalid Inputs:
Username = {username stored in database} Password = {entered password matches the corresponding username in the database}	Username = {Any username not stored in the database} Password = {A password that doesn't match a stored username} Empty values

Pre-conditions:	Post-conditions:
<ul style="list-style-type: none">User has application installedUser has already created account for successful login	<ul style="list-style-type: none">On successful login, user's details are stored in a variable passed between forms to ensure correct permissions based on user type.

<u>Test Case ID</u>	<u>Test Scenario</u>	<u>Test Steps</u>	<u>Test Data</u>	<u>Expected Results</u>	<u>Actual Results</u>	<u>Pass/Fail</u>
TC01	Admin user logs in with correct details	1. Open application 2. Enter username 3. Enter password 4. Click login button/press enter key	username = "TomChate" password = "password"	Login form closed, Admin Menu opened	As expected	Pass
TC02	General user logs in with correct details		username = "JohnSmith" password = "testing"	Login form closed, General user Menu opened	As expected	Pass
TC03	User attempts to logs in with password field left blank		username = "TomChate" password = ""	An error message is displayed and the user stays on the login form, a password must be entered.	As expected	Pass
TC04	User attempts to login with username field left blank		username = "" password = "password"	An error message is displayed and the user stays on the login form, a username must be entered.	As expected	Pass

TC05	Unknown (not stored in DB) user attempts to log in		username = "LillieHogg" password = "computer"	An error message is displayed and the user stays on the login form, the user cannot be found.	As expected	Pass
TC06	User tries to login without filling in fields		username = "" password = ""	An error message is displayed and the user stays on the login form, both fields need to be filled in.	As expected	Pass

Adding a Motherboard

This test case is testing the adding of a motherboard component to the system. Here all fields are required to be complete before adding a new motherboard. This function is similar across the other components, such as adding a GPU.

Valid Inputs:	Invalid Inputs:
Model = {string} Make = {make selected from combo box} Socket = {socket selected from combo box} Form Factor = {form factor selected from combo box} RAM slots = {positive integer} Max RAM = {positive integer of appropriate RAM size}	Model = {string with no chars/integers} Make = {no selection made} Socket = {no selection made} Form Factor = {no selection made} RAM slots = {anything not an integer, inappropriately large integer} Max RAM = {anything not an integer, an integer not of RAM size} (i.e. 8Gb is valid, 7.435Gb is not) Empty values

Pre-conditions:	Post-conditions:
<ul style="list-style-type: none"> User has application installed User has successfully logged in with a valid account Component does not already exist 	<ul style="list-style-type: none"> On successful creation of a motherboard, it is stored in the Motherboard Table in the database where it can be used to add a motherboard to a build.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TC07	The user adds a motherboard to the database to be ready for selection in a make	1. Enter part details in text box / select from combo box 2. Repeat for all part variables 3. Click save when complete	Model = "AS-12x" Make = "Acer" Socket = "LGA1151" Form Factor = "ATX" RAM slots = 2 Max RAM = 32	The data is accepted, and added to the Motherboard Table in the database	As expected	Pass
TC08	Attempting to add a motherboard		Model = "AS-12x"	Invalid Data is not accepted if one	Empty String error in	Pass*

	with one input being incorrect		Make = "Acer" Socket = "LGA1151" Form Factor = "ATX" RAM slots = -2 Max RAM = 32	component is wrong, error message is provided to user when submitted	compiler	
TC09	Trying to add a motherboard with no inputs selected		Model = "" Make = "" Socket = "" Form Factor = "" RAM slots = "" Max RAM = ""	Empty inputs are invalid and a motherboard cannot be accepted. Error message is produced.	Empty string error in compiler	Pass*
TC10	Adding a motherboard which is already in the database.		Model = "AS-12X" Make = "Acer" Socket = "LGA1151" Form Factor = "ATX" RAM slots = 2 Max RAM = 32	Data is not accepted, due it it already being stored. An error message is provided.	'Connected database successfully'	Fail

Saving A Build

This test case is testing the addition of a build, created by a user, to the database. This is one of the core functions of the system. All fields must be completed for the build to be saved to the database. Please note that the username input is automatically retrieved from the system and is not manually entered by the user. Also, that data for components is not typed in, it is selected from a list.

Valid Inputs:	Invalid Inputs:
Part values must be valid selection from table name = {string of appropriate length} username = {existing username passed from menu}	No part selected name = {duplicate name} username = {unknown/incorrect user} Empty values

Pre-conditions:	Post-conditions:
<ul style="list-style-type: none"> User has application installed User has successfully logged in with a valid account Build with same name does not already exist 	<ul style="list-style-type: none"> On successful creation of a build, it is stored in the Build Table in the database where it can be later accessed.

<u>Test Case ID</u>	<u>Test Scenario</u>	<u>Test Steps</u>	<u>Test Data</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Pass/Fail</u>
TC11	Selecting build components and successfully saving the build	1. User selects a part component 2. In the pop-up window, the user selects their part choice 3. Click save when complete 4. Repeat for all parts 5. Enter a build name 6. Click save when complete	motherboard = "MSI H110M Gaming" CPU = "Intel Core i5-7600K" RAM = "MSI 100x3" storage = "Intel SSDPEKKW128" GPU = "MSI GTX 970 GAMING" PSU = "Gigabyte GE-P450P-C2" PCCase= "Asus Vento" cooler = "Intel Cooler V8" accessory = "AMD TORX FAN 12CM" name = "Gaming Build" username = "TomChate"	The build is saved successfully into the Build Table in the database	As Expected	Pass
TC12	Selecting build components and attempting to save the build with one part missing		motherboard = "" CPU = "Intel Core i5-7600K" RAM = "MSI 100x3" storage = "Intel SSDPEKKW128" GPU = "MSI GTX 970 GAMING" PSU = "Gigabyte GE-P450P-C2" PCCase = "Asus Vento" cooler = "Intel Cooler V8" accessory = "AMD TORX FAN 12CM" name = "Gaming Build" username = "TomChate"	Build can not be saved with parts missing. Error message is provided to the user	'Connected database successfully'	Fail
TC13	Selecting build components and attempting to save the build with no name		motherboard = "MSI H110M Gaming" CPU = "Intel Core i5-7600K" RAM = "MSI 100x3" storage = "Intel SSDPEKKW128" GPU = "MSI GTX 970 GAMING" PSU = "Gigabyte GE-P450P-C2" PCCase = "Asus Vento" cooler = "Intel Cooler V8" accessory = "AMD TORX FAN 12CM" name = "" username = "TomChate"	Build can not be saved with a name missing. Error message is provided to the user	As expected	Pass
TC14	Selecting build components and attempting to save the build with all inputs blank		motherboard = "" CPU = "" RAM = "" storage = "" GPU = "" PSU = ""	Build cannot be saved if the selections are blank. Error message is provided to user.	As expected	Pass

			PCCase = "" Cooler = "" accessory = "" name = "" username = "TomChate"			
--	--	--	--	--	--	--

Adding a Make

This test case is testing the adding of a new make of a component to the system by the admin user. Here all fields are required to be complete before adding a new make.

Valid Inputs:	Invalid Inputs:
name= {string of appropriate length} website= {website URL including 'www.' and '.com' or other similar domain}	name = {duplicate name} website = {invalid URL, plain string, numbers on their own} Empty values

Pre-conditions:	Post-conditions:
<ul style="list-style-type: none"> User has application installed User has successfully logged in with a valid account Make with same name does not already exist 	<ul style="list-style-type: none"> On successful creation of a make, it is stored in the Make Table in the database where it can be later selected when adding a part.

<u>Test Case ID</u>	<u>Test Scenario</u>	<u>Test Steps</u>	<u>Test Data</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Pass/Fail</u>
TC15	User enters valid make details	1. User enters make name 2. User enters website 3. User clicks save button	name = "AMD" website = "www.amd.com"	Valid input, data is accepted and stored in the database in 'Make table'.	As expected	Pass
TC16	User enters valid website but no name		name = "" website = "www.amd.com"	Invalid input, no data is stored in database and error message is provided to the user	As expected	Pass
TC17	User enters valid name but no website		name = "AMD" website = ""	Invalid input, no data is stored in database and error message is provided to the user	As expected	Pass
TC18	User attempts to save empty details		name = "" website = ""	Invalid input, no data is stored in database and error message is provided to the user	As expected	Pass

TC19	User enters a duplicate make		name = "AMD" website = "www.amd.com"	Not saved, due to the make already being saved in the database. User is provided with an error message.	As expected	Pass
------	------------------------------	--	---	---	-------------	------

Create User Test Case

This test case is testing the creation of a new user account.

Valid Inputs:	Invalid Inputs:
username = {appropriately lengthed string} password = {password with sufficient numbers/uppercase chars} passwordConfirm = {string matching initial password} forename = {appropriately lengthed string} surname = {appropriately lengthed string} email = {email address containing @, etc.}	username = {too long string} password = {too long string, no numbers/uppercase} passwordConfirm = {string not matching initial password} forename = {too long string, just numbers/special chars} surname = {too long string, just numbers/special chars} email = {string with no @, just numbers/special chars}
Pre-conditions:	Post-conditions:
<ul style="list-style-type: none"> User has application installed 	<ul style="list-style-type: none"> On successful creation of a user account, it is stored in the Account Table in the database where the type of account is used to ensure the user has correct permissions throughout use of the application.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
TC20	Creating a user account with valid details	1. Enter user details in text box 2. Repeat for all part details 3. Click create account button when complete	username = "Claudialacob" password = "inse" passwordConfirm = "inse" forename = "Claudia" surname = "Iacob" email = "claudial@test.com"	Valid inputs entered, data saved to 'Account Table' in database as a general user.	'Account created you will be returned to the login page'	pass
TC21	Creating a user account with one entry missing		username = "" password = "inse" passwordConfirm = "inse" forename = "Claudia"	Invalid data will not be accepted. error message is prompted	'Please complete all fields!'	Pass

			surname = "Iacob" email = "claudial@test.com"			
TC22	Creating a user account with mismatched passwords		username = "Claudialacob" password = "inse" passwordConfirm = "wrong" forename = "Claudia" surname = "Iacob" email = "claudial@test.com"	User will not be saved as passwords do not match	'Account created you will be returned to the login page'	Fail
TC23	Creating a user account with an invalid email		username = "Claudialacob" password = "inse" passwordConfirm = "inse" forename = "Claudia" surname = "Iacob" email = "claudia.com"	User will not be saved An error message providing details of email structure will be provided.	"Email issue, please ensure the email field contains an '@'"	Pass
TC24	Creating a user account with empty entry fields		username = password passwordConfirm = forename = surname = email =	No data entered. Not valid entries therefore data will not be saved	'Please complete all fields'	Pass
TC25	Attempting to create a duplicate		username = password passwordConfirm = forename = surname = email =	User is not saved as a user already has this username. An error message is provided.	'Account in use'	Pass

Component Testing

Parameter interfaces

Data passed from one component to another; methods.

See appendix 3.1

In the 'AddMotherboard' form, when the form is initialised via the constructor it calls the 'populateComboBoxes' from within the form (AddMotherboard.java) where it fills up the combo boxes.

Shared-memory interfaces

A block of memory is shared between components; embedded systems.

See appendix 3.2

Additionally we have examples of our 'user' parameter from the 'Login.java' form being shared across forms. When the user logs in, an instance of UserAccount is created which is passed between forms to ensure users have the appropriate permissions and access to the correct menus.

See appendix 3.3

Another example of this is where we have our 'myPart' variable which is passed to the 'EditComponent.java' form where it instantiates the correct part based on the parameter passed.

Procedural interfaces

One component encapsulates a set of procedures that can be called by other components; objects

See appendix 3.4

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.¹ In 'Make.java', the name and website variables of the Make class are declared as private, and there are public setter methods to modify the variable values.

Message passing interfaces

One component requests a service from another component via a message; client-server.

See appendix 3.5

Our database is running on a Raspberry Pi so to connect to it, the code in 'DatabaseConnection.java' uses the host IP and uName/uPass to access the database remotely.

¹ https://www.tutorialspoint.com/java/java_encapsulation.htm

System Testing

The goal of system testing is to test the system from start to finish to ensure it meets all of the requirements stated previously in our SRS. In this stage we are using black-box testing, by using this type we can see what real interactions are with the system (as users will have no knowledge the internal structure of the system), using this type of testing would allow us to find inconsistencies in the system such as incorrect functions; interface errors; and performance errors. This involved us preparing a test plan (i.e. the goals of the tests), instantiating the test cases of the system, and performing the system test, this was carried out by participants for usability of the system. The participants did not know the programming language used or how the software had been implemented and are specifically tested through the users point of view.

Stress Testing

To simulate the software being run by multiple users JMeter was used, a plugin for the Netbeans development environment. A script was created to work out the highest possible amount of users the system could handle. Testing started at 1 user and then was multiplied by 10 until the test failed. After it failed a manual change of the number of users being simulated was required. After testing a maximum of 50,000 users is predicted to be the upper boundary of maximum amount of users. **Appendix 2** illustrates the results. The database connection was also tested. This was only able to deal with 15 requests per second, this may be due to the hardware that the database is running on (Raspberry Pi). Overall, the the stress test shows that the developed system will be able to deal with a small user base. And with possible hardware upgrades for the database server more requests may be able to be dealt with.

Tools Used

There have been a number of tools used in the process of testing the ComPChecker system, these processes have been used in the testing of functionality, performance and sustainability of the system. One of the tools we used while testing our ComPChecker application was JMeter, used for performance testing of Java applications such as loading data and functional testing, and is an additional plugin for Java Netbeans². This tool was used to stress test our application and results of the stress test are displayed in Appendix 2. Results of each of the tests have been listed in the appendices at the end of this document. The findings of the stress test show that the application itself could handle a substantial amount of users for a small company, however the performance of the system can be improved by enabling more than 15 users to request data from the database at one time. The findings from the usability testing found that overall the application is fairly usable for a number of different kinds of users, there were however a few issues linked with the functionality of some of the application and directing of pages, which was particularly present with the 'create build' menu.

JUnit was also used to test the functionality of the application to ensure that the ComPChecker is running as it should be. The results of the JUnit tests have been demonstrated using automated testing in the test cases above with the test inputs, its expected outputs and its actual outputs of running the tests.

² <http://jmeter.apache.org/>

Usability Evaluation and Process

To evaluate the usability of the system, we had a number of participants carry out a set of tasks we created while we observed the user interacting with the application system. Users knowledge of computer parts and PC builds ranged from a substantial amount of knowledge to no knowledge at all. This is to prevent any inconsistency in the final results of the usability test carried out, as those with more knowledge may find it easier to use an application based on computer and technologies. The tasks listed below and are as follows, user notes taken by us are listed in **appendix 1**. By using 5 participants we are likely to highlight the majority of usability issues (Nielsen, 2000, para. 9)³

Tasks which will be carried out by each user:

Task one: Create Account for yourself

Task Two: Log into your account

Task Three: Find Create Build

Task Four: Create a build and Select your required components

- The User participants must consider selecting components for CPU, Motherboard, RAM, Graphics card, Storage, Case, Power Supply, Cooling, Accessories and name their build accordingly.

Task Five: Save your build

Task Six: View the build you have just created

As well as identifying the main usability issues of the ComPChecker application, some logical and system issues were highlighted by the users when using the system, and these are listed in the notes below for each of the participants we tested. To gain additional user feedback on their thoughts on the usability of the application ComPChecker, we also had each of the participants fill out a System Usability Scale form (An SUS), to determine what their initial opinions of the system. These are shown in appendix 1 also. The benefits of using sus forms is to be able to keep a record of users main issues with the system and then compare users first thoughts of the application with a final product, to keep records of usability improvements.

One key issue raised by the majority of tested users was that when they registered and created an account they were not taken to the home page. Instead they had to re-log in. This issue is a valid point, however, from a security perspective getting users to login promotes them remembering their passwords. Additionally, it decreases temporary accounts being created.

³ Nielsen, J. (2000). *Why You Only Need to Test with 5 Users*. Nngroup.com. Retrieved 20 March 2017, from <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>

Appendix 1

Usability Testing Participant notes

Participant User 1 -

- User navigated to create account easily
- Created User account with ease
- Did not particularly like that the user had to log in again - wanted to be taken straight to home screen
- Was slightly confused with the amount of options (took a while to look through page)
- Thought some pages took too long to load
- Slight confusion with selecting components due to no knowledge with computer parts
- Was confused when they saved a build and there was no message to say it had been saved
- Located view build with ease

Participant User 2 -

- User at first created an account by typing a username and password into the log in input box, instead of clicking create account
- Entered data into create account with ease
- Thought the system loaded pages quickly
- Logged into the system without any problems
- Took some time to find 'create build'
- Did not understand task 4 at first 'select components in create build', needed some explanation
- Selected components easily
- Was confused when the user was not re directed to the menu page or message wasn't prompted to inform the build had been saved
- Located view build with ease
-

Participant User 3 -

- User located and created an account with ease
- Did go back and change passwords a couple of times due to incorrect passwords match and validation message appearing
- Logged into the system easily
- Found create build with no problems
- Had some trouble understanding how selecting components worked due to no knowledge on computing
- Saved build easily

- Like other participants, was confused when he was not prompted with save message or re directed to main menu
- Located view build with ease.

Participant User 4-

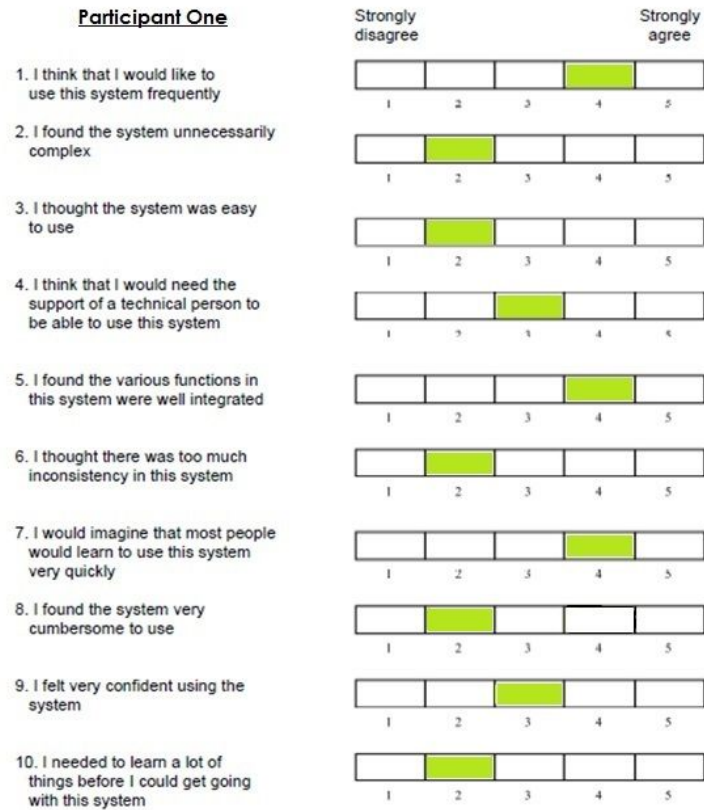
- Found create user with ease
 - Was confused about which fields were required inputs (no asterisks)
 - No confirmation that passwords match when a account is created
 - Did not like that the system didn't log the user in straight away
- Found 'Create Build' with ease
- No problems with selecting components
 - Thought that price should be displayed at the end of of display for component
 - Did not like that there was no currency symbol for price (confusion on what type of currency is being displayed)
 - Type of RAM is not displayed
 - Core-clock looks wrong
 - Some components were displayed twice
- Did not like that no prompt message to say that a build had been saved
- Doesn't like that it opened a new build menu straight away
- Opens another user menu when you return from a build - so two menus are now displayed

Participant User 5-

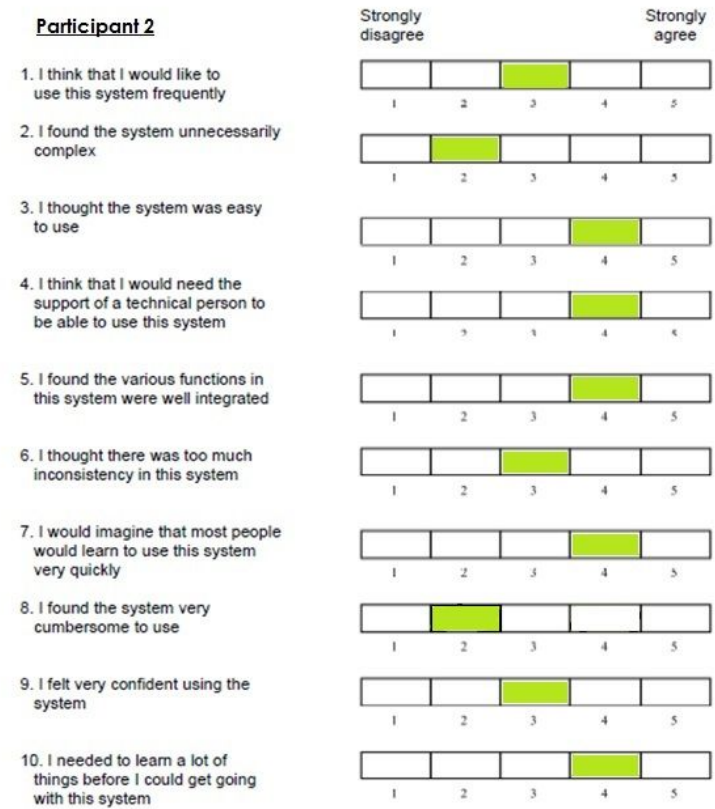
- Found create user with ease
- Forgot to enter '@' in email, and was happy when notified about errors
- When the 'account created' message appeared, the user clicked the ok button twice, which made the system bring up log in menu twice
- Logged in with ease
- Found create build button with ease
 - Did not like that under title 'graphics' it was a boolean - should include more information
 - Would prefer size to have its units e.g. 16GB
 - Did not like that speed also had no units
 - User thought that there should be a double click feature on choosing components. They did not particularly like that they had to keep scrolling to the bottom of the page to select components
 - The user selected 'Storage on the create build menu, turned their head for a second, and when they looked back did not know what the components they were looking at were. They pointed out that each component selector should have a proper name such as 'Select Storage' rather than 'Select Component' as they had to return to the build screen to see what they had been on.
- User thought save should return to main menu rather than opening up a new build
- User found view build with ease

User SUS

Participant One

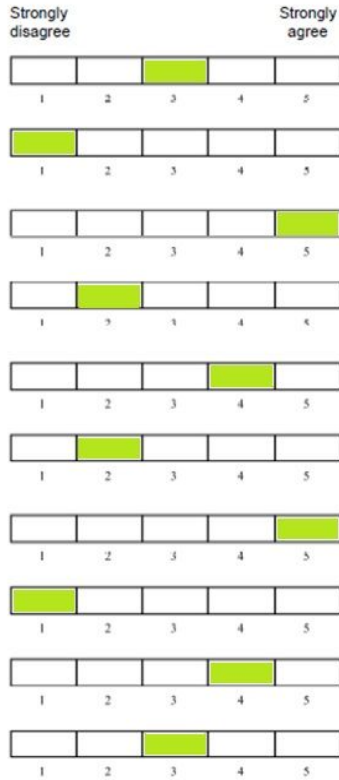


Participant 2



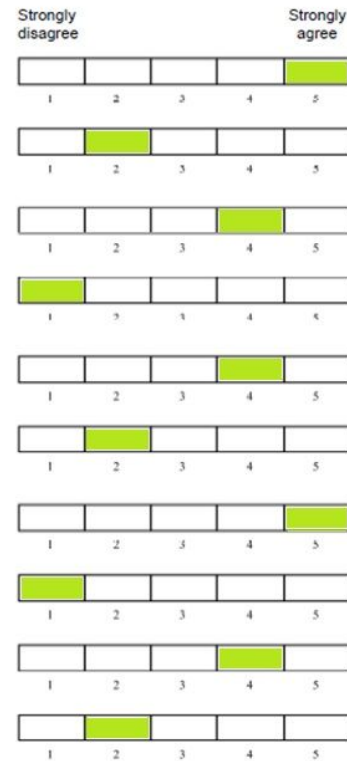
Participant 3

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system



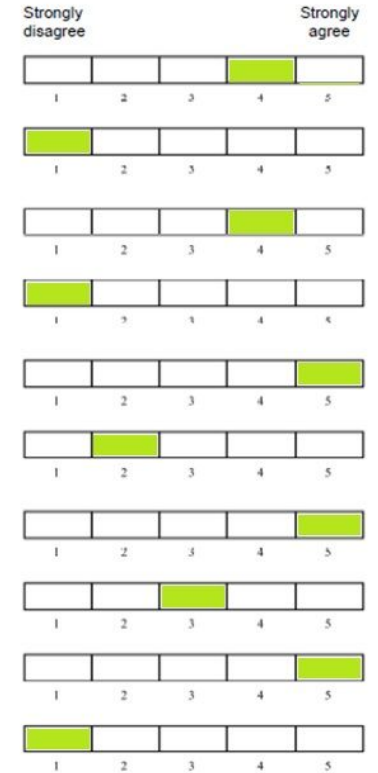
Participant 4

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

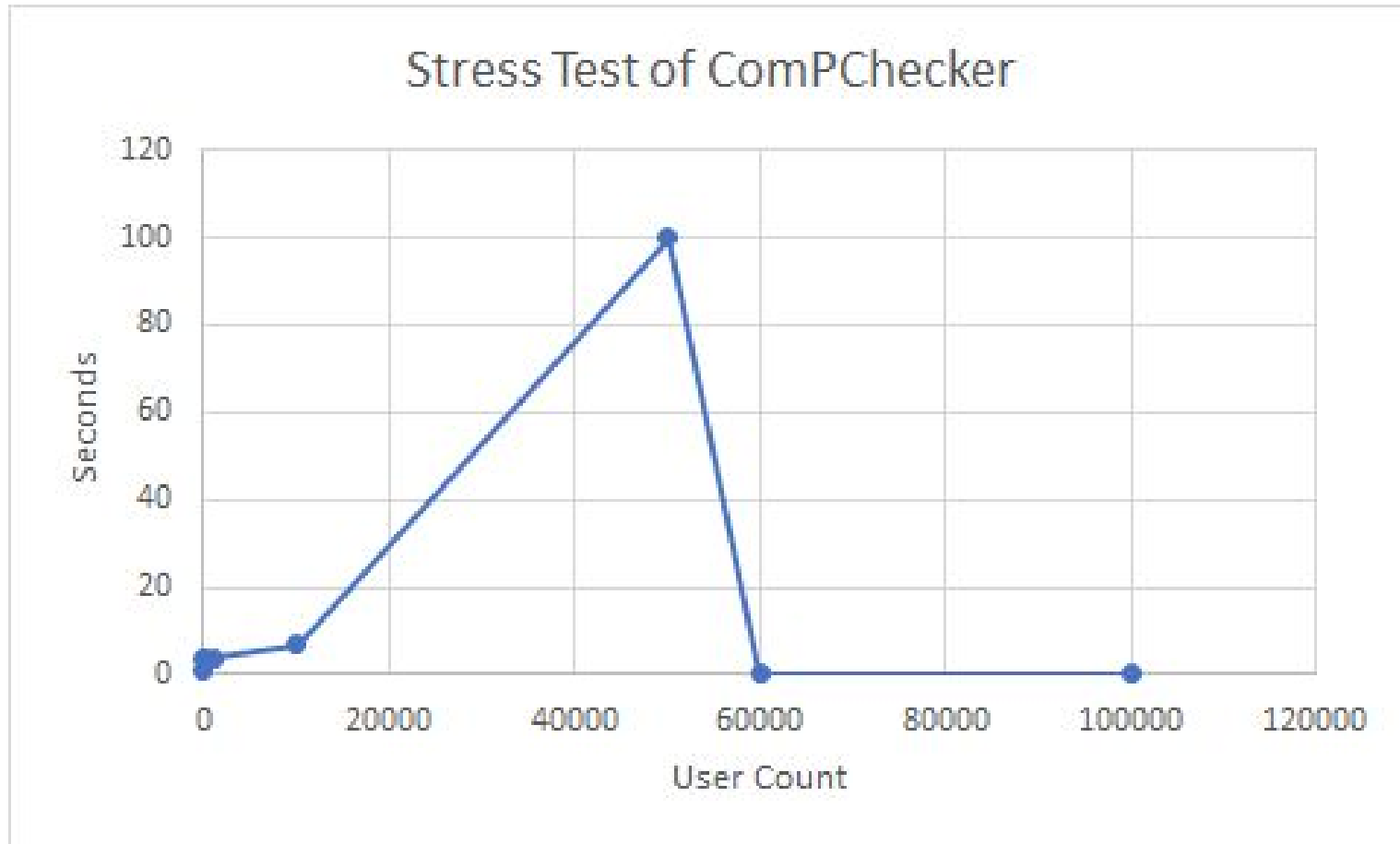


Participant 5

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system



Appendix 2



Appendix 3

3.1

```
AddMotherboard(UserAccount currentUser) {  
    initComponents();  
    populateComboBoxes();  
    this.setTitle("Add CPU"); //Adds a t  
    setLocationRelativeTo(null); //Center  
}  
  
private void populateComboBoxes() {  
    cmbboxMake.removeAllItems();  
    ResultSet rs;  
    Make make = new Make();  
    rs = make.getMakes();
```

3.2

```
UserAccount user = new UserAccount();  
boolean successful = user.LogInService(username)  
if (successful) {  
    boolean type = user.getType(); // if detai  
    if (type) {  
        AdminMenu frm = new AdminMenu(user); //  
        this.setVisible(false);  
    }  
}  
  
public AdminMenu(UserAccount user) {  
    initComponents();  
    this.setTitle("Admin Menu"); //  
    setLocationRelativeTo(null); //  
    currentUser = user; //Assigns
```

3.3

```
String myPart;  
EditComponent frm; //  
switch (input) {  
  
    case "Accessory":  
        myPart = "Accessory";  
        frm = new EditComponent(myPart); //  
        frm.setVisible(true);  
        break;  
}  
  
public EditComponent(String type) {  
    partType = type;  
    initComponents();  
    this.setTitle("Select Component");  
    setLocationRelativeTo(null); //Ce
```

3.4

```
public class Make {  
    private String name;  
    private String website;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
public void setWebsite(String website) {  
    this.website = website;  
}
```

3.5

```
public static Connection establishConnection() {    //This method connects  
  
    try {  
  
        String host = "jdbc:mysql://213.104.129.95:3306/INSE";    //Location  
        String uName = "root";    //account details for accessing database  
        String uPass = "root";  
        Connection con = DriverManager.getConnection(host, uName, uPass);  
        System.out.println("Connected database successfully...");  
  
        return con;  
    } catch (SQLException err) {  
        System.out.println(err.getMessage());    //Prints out SQL error if c  
        return null;  
    }  
}
```