****

**SSE3306**

**HUMAN COMPUTER INTERFACE**

**SEMESTER 4 2018/2019**

**INDIVIDUAL LAB REPORT**

**My Invitation**

NAME: MUHAMMAD HARITH BIN ZAINUDIN

MATRIC NO: 192171

LECTURER:  DR. AZRINA BINTI KAMARUDDIN

Table of Contents

[High Fidelity Prototype 3](#_Toc8733215)

[Describing the design with Nielsen Heuristics for UI 4](#_Toc8733216)

[The rationale of the widgets or UI components selection 5](#_Toc8733217)

[Use Case Diagram 5](#_Toc8733218)

[Java Source Code 6](#_Toc8733219)

[Main.java 6](#_Toc8733220)

[Main.fxml 6](#_Toc8733221)

[MainController.java 9](#_Toc8733222)

[Attendance.fxml 12](#_Toc8733223)

[AttendanceController.java 13](#_Toc8733224)

[Person.java 15](#_Toc8733225)

[Help Description 18](#_Toc8733226)

[Executive Summary 21](#_Toc8733227)

[Purpose of study 21](#_Toc8733228)

[Methodology 21](#_Toc8733229)

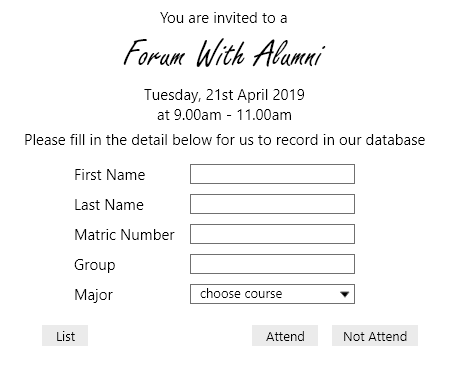
[Method 21](#_Toc8733230)

[Findings and Recommendations 22](#_Toc8733231)

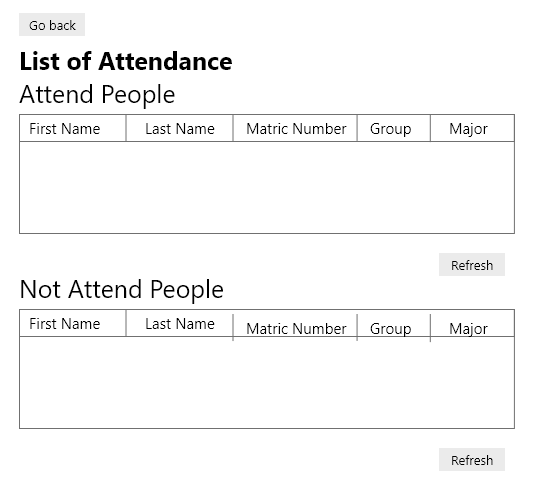
[Discussion 23](#_Toc8733232)

[Appendix 23](#_Toc8733233)

# High Fidelity Prototype



Prototype of the Main Page



Prototype of the Attendance List

# Describing the design with Nielsen Heuristics for UI

1. Match between system and the real world

The system uses the language that most of the people in the world use that is English. It is easy to understand as the English that is use is simple. For example, if the user wants to go back, they just need to click the “go back” button, and the system will go back to the main page.

1. Consistency and standards

The users will not wonder whether the words, situations or action that is use will mean the same thing. This system is direct. If the user wants to attend the program, then, they just need to click attend button. If not, just click not attend button. All the button in this system is direct and will not make the user think twice.

1. Error prevention

There is error prevention in this system where, if the user not fill in all the information, then, they will be an alert box pop out saying that “Please fill in all the information”.

1. Recognition rather than recall

This system minimizes the user’s memory by making all the buttons are visible. They should not have to remember information from one part to another. The instruction is visible. For example, at the main page, saying that, “Please fill in the information below for us to store in our database”. The instruction is direct and does not need the user to remember.

1. Flexibility and efficiency of use

This system is flexible and efficient as if the user wanted to go back to the main page, then they just need to click the “go back” button and they will be redirected back to the main page. If the user wanted to see the list of the attendance from the main page, then they just need to click the “List” button and the system will show the attendance list.

1. Aesthetic and minimalist design

There is no irrelevant information or dialogue that will show throughout the system. It only show what important to the user. It is minimalist and does not have much button or info on a screen.

1. Help users recognize, diagnose and recover from errors

If the user does not fill in all the information, and click attend or not attend, then there will be an alert dialog saying that the user needs to fill in all the information. If not, they cannot proceed. In the alert box, there is no codes, only plain language and instructions that the user need to do to attend or not attend the program.

1. Help and documentation

Although this system is simple and easy to understand, there is help and documentation provided for this system for the user to use the system. If the user needs any help regarding the system, then they can read the documentation.

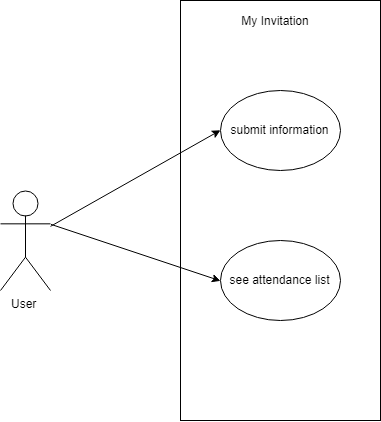
# The rationale of the widgets or UI components selection

The widgets and UI that is use in this program is using the JAVAFX widgets. So, all the interface will follow JAVAFX style of programming. The button is just a rectangle that have greyish colour so that user can differentiate the button with the textfield.

For the user to choose the major, they will be a dropdown. This is to prevent the user from typing many types of major, as there are only 4 majors in CSIT faculty. By doing this, the user only needs to choose the major that they are now. This will make the user easy as they do not need to type their major in the textfield.

The title is differentiated from the content by having a different font and stylize so that for the user focus on what to focus. What is the event about and what is the table about.

# Use Case Diagram



# Java Source Code

## Main.java

// Author: Muhammad Harith Bin Zainudin

// Matric Number: 192171

// Course code and name: SSE3306 Human Computer Interaction

// Lab Question: 1

// Lecturer: Dr. Azrina Binti Kamaruddin

// Description: This java class is for the main stage that will be load and launch calling the scene

// Objective: Analyze Problem and Construct Program Using GUI Components and applying design principles.

// Last Update: 12 May 2019

package application;

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.fxml.FXMLLoader;

public class Main extends Application {

private static Object primaryStage;

@Override

public void start(Stage primaryStage) {

try {

primaryStage.setTitle("My Invitation");

VBox root = (VBox)FXMLLoader.load(getClass().getResource("Main.fxml"));

Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());

primaryStage.setScene(scene);

primaryStage.show();

} catch(Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

launch(args);

}

}

## Main.fxml

// this is fxml file for the main page interface

<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.\*?>

<?import javafx.collections.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.text.\*?>

<VBox alignment="TOP\_CENTER" disable="false" fillWidth="true" focusTraversable="false" minHeight="-1.0" minWidth="-1.0" mouseTransparent="false" pickOnBounds="false" prefHeight="543.0000999999975" prefWidth="632.0" snapToPixel="true" visible="true" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="application.MainController">

<!-- TODO Add Nodes -->

<children>

<Label alignment="CENTER" contentDisplay="CENTER" text="You are invited to a">

<font>

<Font name="Arial" size="20.0" fx:id="x1" />

</font>

</Label>

<Label alignment="CENTER" contentDisplay="CENTER" text="Forum with Alumni" textAlignment="CENTER">

<font>

<Font name="Freestyle Script" size="70.0" />

</font>

</Label>

<Label alignment="CENTER" contentDisplay="CENTER" font="$x1" text="Tuesday, 21st April 2019" />

<Label alignment="CENTER" contentDisplay="CENTER" font="$x1" text="at 9.00am - 11.00am">

<VBox.margin>

<Insets top="5.0" />

</VBox.margin>

</Label>

<Label alignment="CENTER" contentDisplay="CENTER" font="$x1" rotate="0.0" text="Please fill in the detail below for us to record in our database">

<VBox.margin>

<Insets bottom="15.0" top="5.0" />

</VBox.margin>

</Label>

<BorderPane maxHeight="-1.0" minHeight="-1.0" prefHeight="200.0" prefWidth="200.0" VBox.margin="$x2" VBox.vgrow="ALWAYS">

<top>

<GridPane alignment="CENTER" prefHeight="267.0" prefWidth="531.0" BorderPane.alignment="CENTER" BorderPane.margin="$x2">

<children>

<Label alignment="CENTER" contentDisplay="CENTER" text="First Name" GridPane.columnIndex="0" GridPane.halignment="LEFT" GridPane.rowIndex="0">

<font>

<Font size="20.0" fx:id="x4" />

</font>

<GridPane.margin>

<Insets fx:id="x2" />

</GridPane.margin>

</Label>

<Label alignment="CENTER" contentDisplay="CENTER" font="$x4" text="Last Name" GridPane.columnIndex="0" GridPane.halignment="LEFT" GridPane.rowIndex="1" />

<Label alignment="CENTER" contentDisplay="CENTER" font="$x4" text="Matric Number" GridPane.columnIndex="0" GridPane.halignment="LEFT" GridPane.rowIndex="2" />

<Label alignment="CENTER" contentDisplay="CENTER" font="$x4" text="Group" GridPane.columnIndex="0" GridPane.halignment="LEFT" GridPane.rowIndex="3" />

<Label alignment="CENTER" contentDisplay="CENTER" font="$x4" text="Major" GridPane.columnIndex="0" GridPane.halignment="LEFT" GridPane.rowIndex="4" />

<TextField fx:id="fname" maxWidth="-Infinity" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="0" />

<TextField fx:id="lname" maxWidth="-Infinity" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />

<TextField fx:id="matricNum" maxWidth="-Infinity" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />

<TextField fx:id="groupNum" maxHeight="-1.0" maxWidth="-Infinity" prefWidth="261.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />

<AnchorPane prefHeight="200.0" prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="5">

<children>

<Button id="clear" fx:id="attend" font="$x4" layoutX="57.0" layoutY="9.0" maxWidth="-Infinity" mnemonicParsing="false" onAction="#attendButton" text="Attend" />

<Button fx:id="notAttend" font="$x4" layoutX="176.0" layoutY="9.0" mnemonicParsing="false" onAction="#notAttendButton" text="Not Attend" />

</children>

<GridPane.margin>

<Insets top="40.0" />

</GridPane.margin>

</AnchorPane>

<ComboBox fx:id="course" disable="false" editable="false" focusTraversable="true" prefWidth="261.0" promptText="choose course" GridPane.columnIndex="1" GridPane.rowIndex="4">

<items>

<FXCollections fx:factory="observableArrayList" />

</items>

</ComboBox>

<Button fx:id="list" font="$x4" mnemonicParsing="false" onAction="#goToList" text="List" GridPane.columnIndex="0" GridPane.rowIndex="5">

<GridPane.margin>

<Insets top="50.0" />

</GridPane.margin>

</Button>

</children>

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="455.0" minWidth="10.0" prefWidth="129.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="528.0" minWidth="10.0" prefWidth="346.0" />

</columnConstraints>

<padding>

<Insets left="110.0" />

</padding>

<rowConstraints>

<RowConstraints maxHeight="45.0" minHeight="10.0" prefHeight="45.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="45.0" minHeight="10.0" prefHeight="45.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="45.0" minHeight="10.0" prefHeight="45.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="44.0" minHeight="10.0" prefHeight="44.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="44.0" minHeight="10.0" prefHeight="44.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="44.0" minHeight="10.0" prefHeight="44.0" vgrow="SOMETIMES" />

</rowConstraints>

</GridPane>

</top>

</BorderPane>

</children>

<padding>

<Insets top="20.0" fx:id="x3" />

</padding>

</VBox>

## MainController.java

// Author: Muhammad Harith Bin Zainudin

// Matric Number: 192171

// Course code and name: SSE3306 Human Computer Interaction

// Lab Question: 1

// Lecturer: Dr. Azrina Binti Kamaruddin

// Description: This is the main controller class that will control the main.fxml

// Objective: Analyze Problem And Construct Program Using GUI Components and applying design principles.

// Last Update: 12 May 2019

package application;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.net.URL;

import java.util.Optional;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ComboBox;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;

import javafx.scene.control.ToggleGroup;

import javafx.stage.Stage;

import javafx.scene.Node;

public class MainController {

@FXML private ResourceBundle resources;

@FXML private URL location;

@FXML private TextField fname;

@FXML private TextField groupNum;

@FXML private TextField lname;

@FXML private TextField matricNum;

@FXML private ComboBox course;

/\*

\* change scene from main page scene to attendance scene and give alert if the textfield is empty.

\* If the textfield is fully fill in, then it will be write to filewriter.

\* This method will be call if the attend button is click.

\*/

public void attendButton(ActionEvent event) throws IOException {

StringBuilder sb = new StringBuilder();

if ((fname.getText().trim().isEmpty()) || (lname.getText().trim().isEmpty()) || (matricNum.getText().trim().isEmpty()) || (groupNum.getText().trim().isEmpty())) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Empty Information");

alert.setHeaderText("No Information inserted");

alert.setContentText("Please fill in the information to proceed!");

alert.showAndWait();

}

else {

sb.append(fname.getText().toString()+ " , ");

sb.append(lname.getText().toString()+ " , ");

sb.append(matricNum.getText().toString()+ " , ");

sb.append(groupNum.getText().toString()+ " , ");

sb.append(course.getValue().toString());

File file = new File("C:\\Users\\Admin\\Desktop\\attend.txt");

FileWriter w;

try {

String newLine = System.getProperty("line.separator");

w = new FileWriter(file,true);

w.write(sb.toString() + newLine);

w.close();

Parent mainParent = FXMLLoader.load(getClass().getResource("Attendance.fxml"));

Scene mainScene = new Scene(mainParent);

Stage window = (Stage)((Node)event.getSource()).getScene().getWindow();

window.setScene(mainScene);

window.show();

} catch (IOException e) {

e.printStackTrace();

}

} // end of else statement

} // end of attendButton method

// change scene from main page to attendance scene if the not attend button is clicked.

public void notAttendButton(ActionEvent event) throws IOException {

StringBuilder sb = new StringBuilder();

if ((fname.getText().trim().isEmpty()) || (lname.getText().trim().isEmpty()) || (matricNum.getText().trim().isEmpty()) || (groupNum.getText().trim().isEmpty())) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Empty Information");

alert.setHeaderText("No Information inserted");

alert.setContentText("Please fill in the information to proceed!");

alert.showAndWait();

} else {

sb.append(fname.getText().toString()+ " , ");

sb.append(lname.getText().toString()+ " , ");

sb.append(matricNum.getText().toString()+ " , ");

sb.append(groupNum.getText().toString()+ " , ");

sb.append(course.getValue().toString());

File file = new File("C:\\Users\\Admin\\Desktop\\not attend.txt");

FileWriter w;

try {

String newLine = System.getProperty("line.separator");

w = new FileWriter(file,true);

w.write(sb.toString() + newLine);

w.close();

Parent mainParent = FXMLLoader.load(getClass().getResource("Attendance.fxml"));

Scene mainScene = new Scene(mainParent);

Stage window = (Stage)((Node)event.getSource()).getScene().getWindow();

window.setScene(mainScene);

window.show();

} catch (IOException e) {

e.printStackTrace();

}

} // end of else statement

} // end of method notAttendButton

// this method for list button that when click will go to attendance scene

@FXML

public void goToList(ActionEvent event) throws IOException {

Parent mainParent = FXMLLoader.load(getClass().getResource("Attendance.fxml"));

Scene mainScene = new Scene(mainParent);

Stage window = (Stage)((Node)event.getSource()).getScene().getWindow();

window.setScene(mainScene);

window.show();

}

@FXML

void initialize() {

// initialize for course combo box

course.getItems().addAll("Software Engineering", "Network", "Multimedia", "Computer System");

}

} // end of class

## Attendance.fxml

// this is the fxml file that is the interface of the attendance scene. It will show the

// tableview of attend and not attend person

<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.\*?>

<AnchorPane prefHeight="767.6015625" prefWidth="853.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="application.AttendanceController">

<!-- TODO Add Nodes -->

<children>

<Label layoutX="41.0" layoutY="82.0" text="List of Attendance">

<font>

<Font name="Arial Bold" size="30.0" />

</font>

</Label>

<TableView fx:id="attend" layoutX="41.0" layoutY="162.0" prefHeight="225.0" prefWidth="745.0">

<columns>

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="156.0" text="First Name" fx:id="fnameColumn" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="175.0" text="Last Name" fx:id="lnameColumn" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="151.0" text="Matric Number" fx:id="matricNumColumn" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="86.0" text="Group" fx:id="groupNumColumn" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="180.0" text="Major" fx:id="courseColumn" />

</columns>

</TableView>

<Label layoutX="41.0" layoutY="127.0" text="Attend People">

<font>

<Font name="Arial" size="25.0" fx:id="x1" />

</font>

</Label>

<Label font="$x1" layoutX="41.0" layoutY="442.0" text="Not Attend People" />

<Button layoutX="41.0" layoutY="27.0" mnemonicParsing="false" onAction="#goToMain" text="&lt;- Go back">

<font>

<Font size="20.0" fx:id="x2" />

</font>

</Button>

<Button fx:id="refresh" font="$x2" layoutX="686.0" layoutY="400.0" mnemonicParsing="false" onAction="#refresh" text="Refresh" />

<TableView id="attend" fx:id="notAttend" layoutX="41.0" layoutY="479.0" prefHeight="225.0" prefWidth="745.0">

<columns>

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="156.0" text="First Name" fx:id="fnameColumnNA" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="175.0" text="Last Name" fx:id="lnameColumnNA" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="151.0" text="Matric Number" fx:id="matricNumColumnNA" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="86.0" text="Group" fx:id="groupNumColumnNA" />

<TableColumn maxWidth="5000.0" minWidth="10.0" prefWidth="180.0" text="Major" fx:id="courseColumnNA" />

</columns>

</TableView>

<Button id="refresh" font="$x2" layoutX="686.0" layoutY="713.0" mnemonicParsing="false" onAction="#refreshNotAttend" text="Refresh" />

</children>

</AnchorPane>

## AttendanceController.java

// Author: Muhammad Harith Bin Zainudin

// Matric Number: 192171

// Course code and name: SSE3306 Human Computer Interaction

// Lab Question: 1

// Lecturer: Dr. Azrina Binti Kamaruddin

// Description: This is the attendance controller class that will control the attendance.fxml

// Objective: Analyze Problem And Construct Program Using GUI Components and applying design principles.

// Last Update: 12 May 2019

package application;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.nio.file.Files;

import java.util.Collection;

import java.util.Scanner;

import java.util.stream.Collectors;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.stage.Stage;

public class AttendanceController {

// column for attend

@FXML private TableView<Person> attend;

@FXML private TableColumn<Person, String> fnameColumn;

@FXML private TableColumn<Person, String> lnameColumn;

@FXML private TableColumn<Person, String> matricNumColumn;

@FXML private TableColumn<Person, String> groupNumColumn;

@FXML private TableColumn<Person, String> courseColumn;

// column for not attend

@FXML private TableView<Person> notAttend;

@FXML private TableColumn<Person, String> fnameColumnNA;

@FXML private TableColumn<Person, String> lnameColumnNA;

@FXML private TableColumn<Person, String> matricNumColumnNA;

@FXML private TableColumn<Person, String> groupNumColumnNA;

@FXML private TableColumn<Person, String> courseColumnNA;

// method for closing the stage when close button is click

public void exitStage(ActionEvent event) throws IOException {

System.exit(0);

}

// method for go back button. when this button is click, it will go back to main page

@FXML

public void goToMain(ActionEvent event) throws IOException {

Parent mainParent = FXMLLoader.load(getClass().getResource("Main.fxml"));

Scene mainScene = new Scene(mainParent);

Stage window = (Stage)((Node)event.getSource()).getScene().getWindow();

window.setScene(mainScene);

window.show();

}

// method for refresh attend button. if this button is clicked , the table for attend will be refresh

public void refresh() throws Exception {

// read for attend

Collection<Person> read = Files.readAllLines(new File("C:\\Users\\Admin\\Desktop\\attend.txt").toPath()).stream().map(line -> {

String[] details = line.split (" , ");

Person p = new Person();

p.setFName(details[0]);

p.setLName(details[1]);

p.setMatricNum(details[2]);

p.setGroupNum(details[3]);

p.setCourse(details[4]);

return p;

}).collect(Collectors.toList());

ObservableList<Person> details = FXCollections.observableArrayList(read);

fnameColumn.setCellValueFactory(data -> data.getValue().fnameProperty());

lnameColumn.setCellValueFactory(data -> data.getValue().lnameProperty());

matricNumColumn.setCellValueFactory(data -> data.getValue().matricNumProperty());

groupNumColumn.setCellValueFactory(data -> data.getValue().groupNumProperty());

courseColumn.setCellValueFactory(data -> data.getValue().courseProperty());

attend.setItems(details);

}

// method for refresh not attend button. if this button is clicked , the table for not attend will be refresh

public void refreshNotAttend() throws Exception {

// read for not attend

Collection<Person> readNot = Files.readAllLines(new File("C:\\Users\\Admin\\Desktop\\not attend.txt").toPath()).stream().map(line -> {

String[] detailsNot = line.split (" , ");

Person p = new Person();

p.setFName(detailsNot[0]);

p.setLName(detailsNot[1]);

p.setMatricNum(detailsNot[2]);

p.setGroupNum(detailsNot[3]);

p.setCourse(detailsNot[4]);

return p;

}).collect(Collectors.toList());

ObservableList<Person> detailsNot = FXCollections.observableArrayList(readNot);

fnameColumnNA.setCellValueFactory(data -> data.getValue().fnameProperty());

lnameColumnNA.setCellValueFactory(data -> data.getValue().lnameProperty());

matricNumColumnNA.setCellValueFactory(data -> data.getValue().matricNumProperty());

groupNumColumnNA.setCellValueFactory(data -> data.getValue().groupNumProperty());

courseColumnNA.setCellValueFactory(data -> data.getValue().courseProperty());

notAttend.setItems(detailsNot);

}

} // end of class

## Person.java

// Author: Muhammad Harith Bin Zainudin

// Matric Number: 192171

// Course code and name: SSE3306 Human Computer Interaction

// Lab Question: 1

// Lecturer: Dr. Azrina Binti Kamaruddin

// Description: This is the person class that hold the getter and setter for all the variable

// Objective: Analyze Problem and Construct Program Using GUI Components and applying design principles.

// Last Update: 12 May 2019

package application;

import javafx.beans.property.IntegerPropertyBase;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

public class Person {

StringProperty fname = new SimpleStringProperty();

StringProperty lname = new SimpleStringProperty();

StringProperty matricNum = new SimpleStringProperty();

StringProperty groupNum = new SimpleStringProperty();

StringProperty course = new SimpleStringProperty();

// getter and setter for fname

public final StringProperty getFName() {

return this.fname;

}

public final void setFName(final java.lang.String fname) {

this.fnameProperty().set(fname);

}

public final StringProperty fnameProperty() {

return this.fname;

}

// getter and setter for lname

public final StringProperty getLName() {

return this.lname;

}

public final void setLName(final java.lang.String lname) {

this.fnameProperty().set(lname);

}

public final StringProperty lnameProperty() {

return this.lname;

}

// getter and setter for matricNum

public final StringProperty getmatricNum() {

return this.matricNum;

}

public final void setMatricNum(final java.lang.String matricNum) {

this.matricNumProperty().set(matricNum);

}

public final StringProperty matricNumProperty() {

return this.matricNum;

}

// getter and setter for groupNum

public final StringProperty getGroupNum() {

return this.groupNum;

}

public final void setGroupNum(final java.lang.String groupNum) {

this.groupNumProperty().set(groupNum);

}

public final StringProperty groupNumProperty() {

return this.groupNum;

}

// getter and setter for course

public final StringProperty getCourse() {

return this.course;

}

public final void setCourse(final java.lang.String course) {

this.courseProperty().set(course);

}

public final StringProperty courseProperty() {

return this.course;

}

}

# Help Description

This is the documentation and guide to use the system.

The system is easy to use and direct forward. When the user launches the system, figure below will be shown.

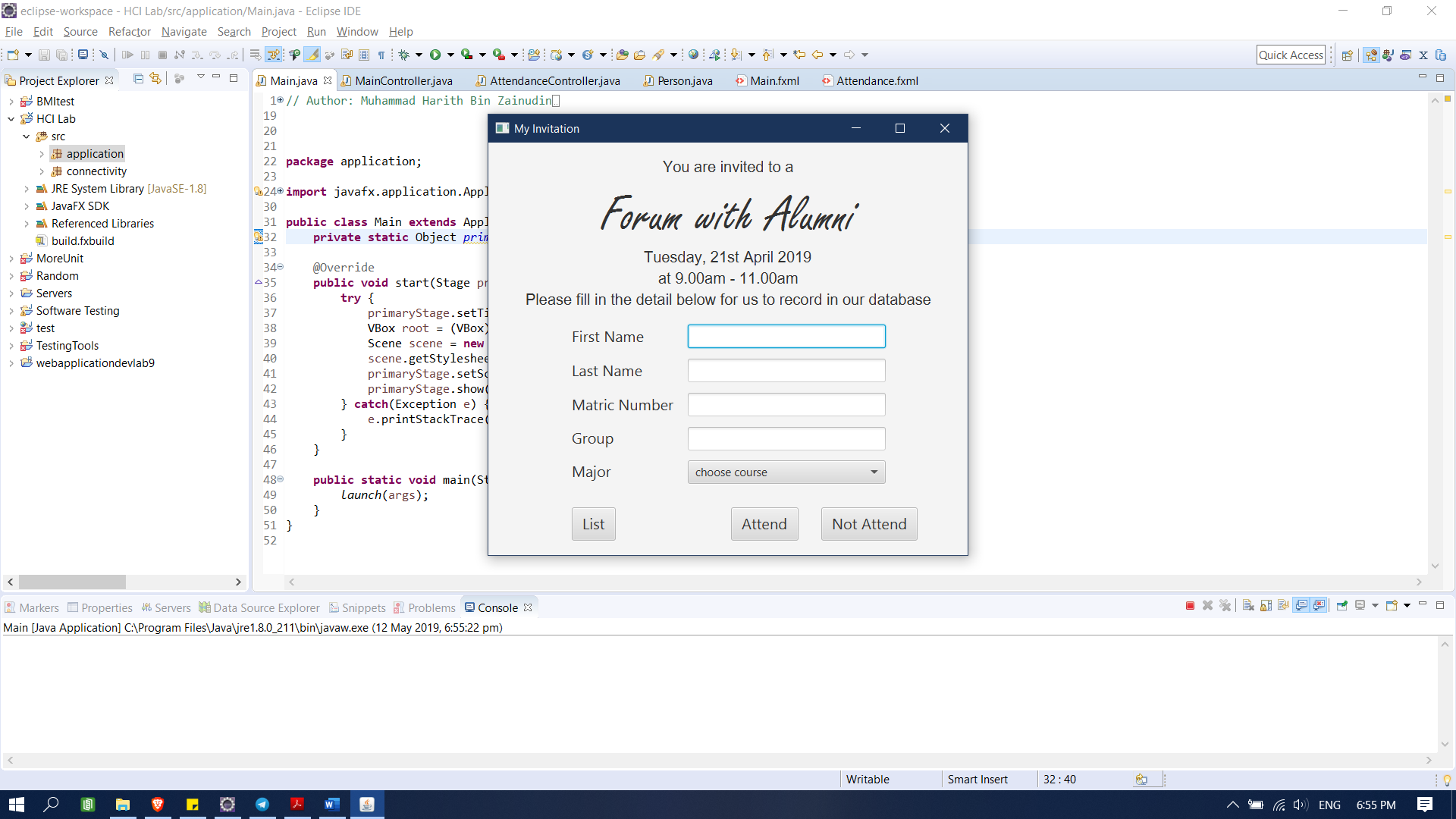


Figure 1: Main page

This is the page where the user will be informed about the program and all the information regarding the program. User need to fill in all the information in order to store their information in the database for references.

There are 3 buttons in the main page. That is:

1. *List button* – This button is to redirect to attendance list and show all the people that attend or not attend the program.
2. *Attend button* – The user should click this button in order to store their information in attend list.
3. *Not attend button* – the user needs to click this button if they not come to the event. Their info will be store in the not attend list people.

If the user click the *List button*, then figure below will be shown.

When the user have fll in all the information and click attend/not attend button, figure 3 will be shown. After that, they need to click refresh to see the latest list.

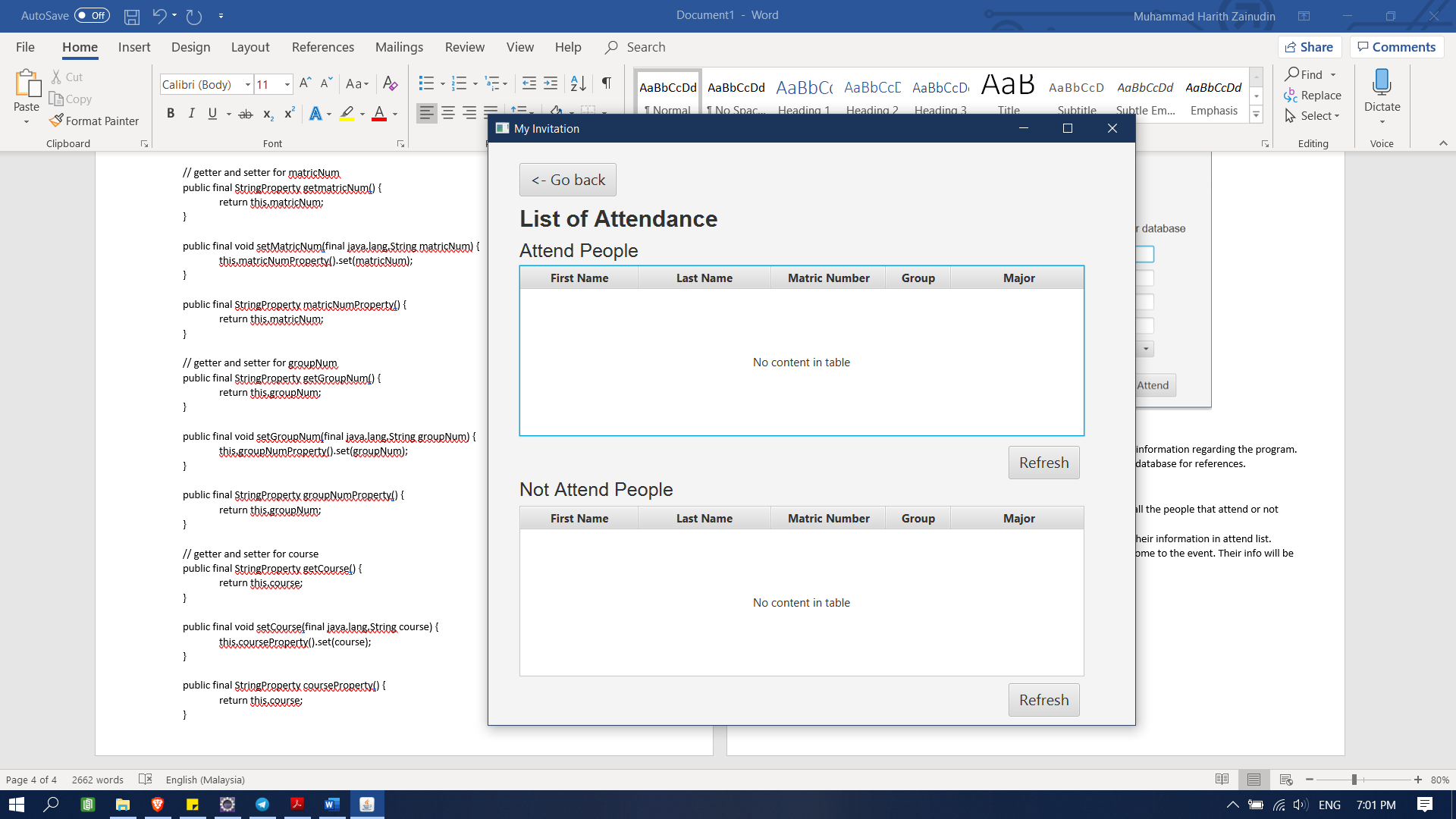


Figure 2: Attendance list page

When this page is being shown, there is 2 table. One is the attend list, and the other one is not attend list. The user need to click the refresh button in order to see the latest list in the table.

If the user click the refresh button, then the list will be shown in the table (figure 3)

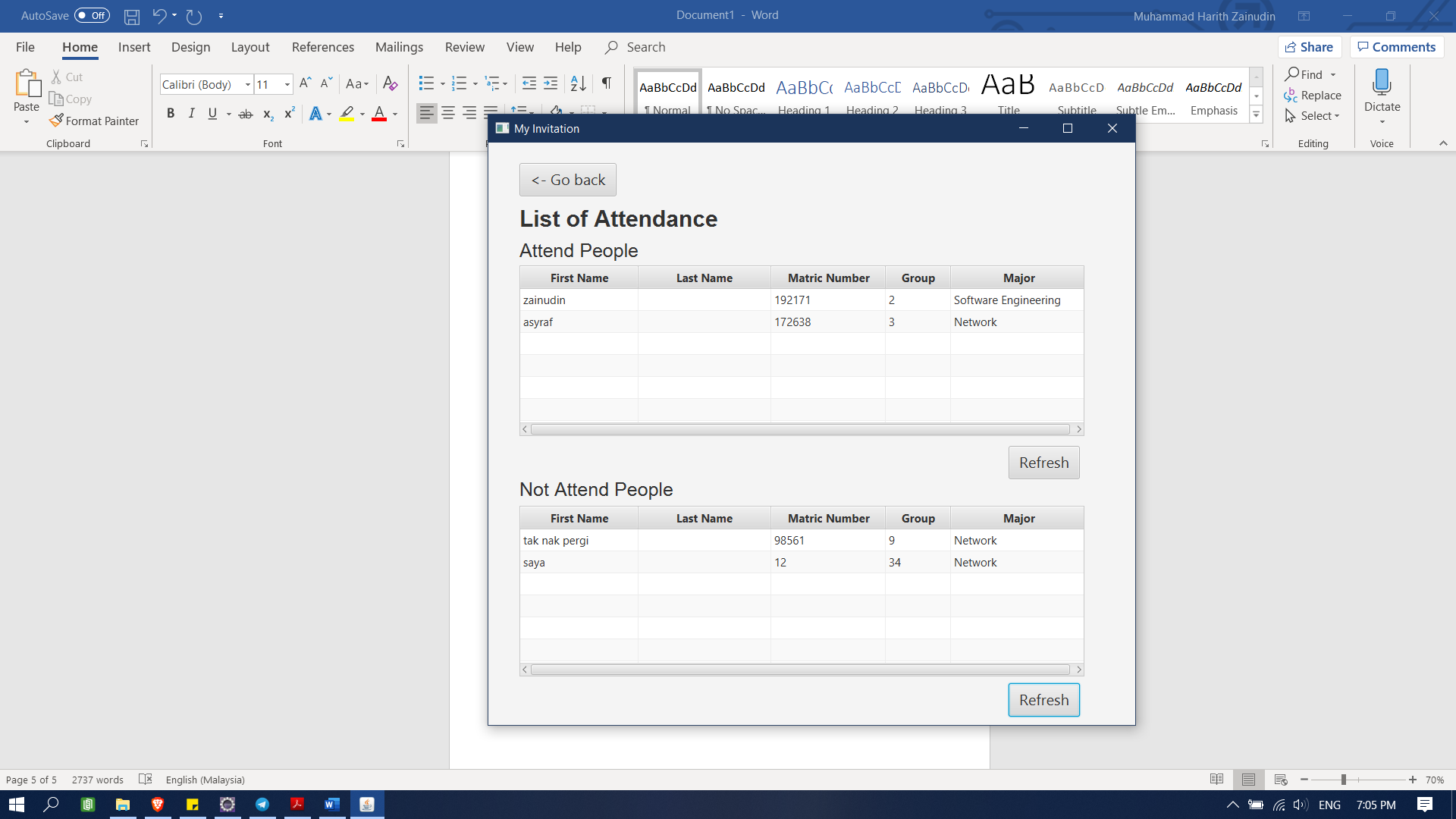


Figure 3: Attendance list page when refresh button is clicked

At this page, the user can go back to the main page by clicking the go back button that is at the upper left of the system. Then, the user will be redirected back to the main page.

USABILITY REPORT FOR MY INVITATION

# Executive Summary

The usability testing of this My invitation system is been done by using System Usability Scale Questionnaire. The purpose for conducting this test is to evaluate the system and the usability. We want to know what the user opinion about the system. The total number of participants that involves in this usability is 5 peoples. The user needs to use the system first and after that, they need to answer the questionnaire. All in all, the user are satisfied with the system and the SUS score is above 68.

# Purpose of study

The purpose of this usability test is to better understand how real users interact with the system and to know the SUS score whether it is usable or not. The primary purpose of a usability test is to improve a design. In a typical usability test, real users try to accomplish typical goals, or tasks, with a product under controlled conditions.

# Methodology

The participants that were chosen for this test is being choose randomly around the Universiti Putra Malaysia. All of them are from Faculty of Science Computer and Information Technology. This is because, this system can be used by anyone and easy to understand. The total number of users that is being choose for this test is 5 user. The age of participants is not being asked in this test.

# Method

The questionnaire that is being used in this test is System Usability Scale (SUS). It consists of a 10-item questionnaire with five response options for respondents; from Strongly agree to Strongly disagree. Originally created by John Brooke in 1986, it allows you to evaluate a wide variety of products and services, including hardware, software, mobile devices, websites and applications. When a SUS is used, participants are asked to score the following 10 items with one of five responses that range from Strongly Agree to Strongly disagree:

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.

To interpret the questionnaire, the participant’s score for each question are converted to a new number, added together and then multiplied by 2.5 to convert the original scores of 0-40 to 0-100. Based on research, a SUS score above a 68 would be considered above average and anything below 68 is below average, however the best way to interpret your results involves “normalizing” the scores to produce a percentile ranking. Here’s an overview of how your scores should measure:

* 80.3 or higher is an A. People love your site and will recommend it to their friends
* 68 or thereabouts gets you a C. You are doing OK but could improve
* 51 or under gets you a big fat F. Make usability your priority now and fix this fast.

# Findings and Recommendations

**Post-Task Overall Questionnaire (System Usability Scale)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Strongly Disagree** | **Disagree** | **Neutral** | **Agree** | **Strongly Agree** | **SUS Score** |
| I think that I would like to use this website frequently |  |  | 1 | 4 |  | 2+12=14  14/5=2.8 |
| I found this website unnecessarily complex | 4 | 1 |  |  |  | 16+3=19  19/5=3.8 |
| I thought this website was easy to use |  |  |  | 2 | 3 | 6+12=18  18/5=3.6 |
| I think that I would need assistance to be able to use this website | 5 |  |  |  |  | 20/5=4 |
| I found the various functions in this website were well integrated |  |  |  | 2 | 3 | 6+12=18  18/5=3.6 |
| I thought there was too much inconsistency in this website | 4 | 1 |  |  |  | 16+3=19  19/5=3.8 |
| I would imagine that most people would learn to use this website very quickly |  |  |  | 1 | 4 | 3+16=19  19/5=3.8 |
| I found this website cumberstone/awkward to use | 5 |  |  |  |  | 20/5=4 |
| I felt very confident using this website |  |  |  | 1 | 4 | 3+16=19  19/5=3.8 |
| I needed to learn a lot of things before I could get going with this website. | 5 |  |  |  |  | 20/5=4 |
| **TOTAL** |  |  |  |  |  | 2.8+3.8+3.6+4+3.6+3.8+3.8+4+3.8+4 = 37.2  37.2\*2.5=**93** |

Based on the table, the result of SUS score is 93.

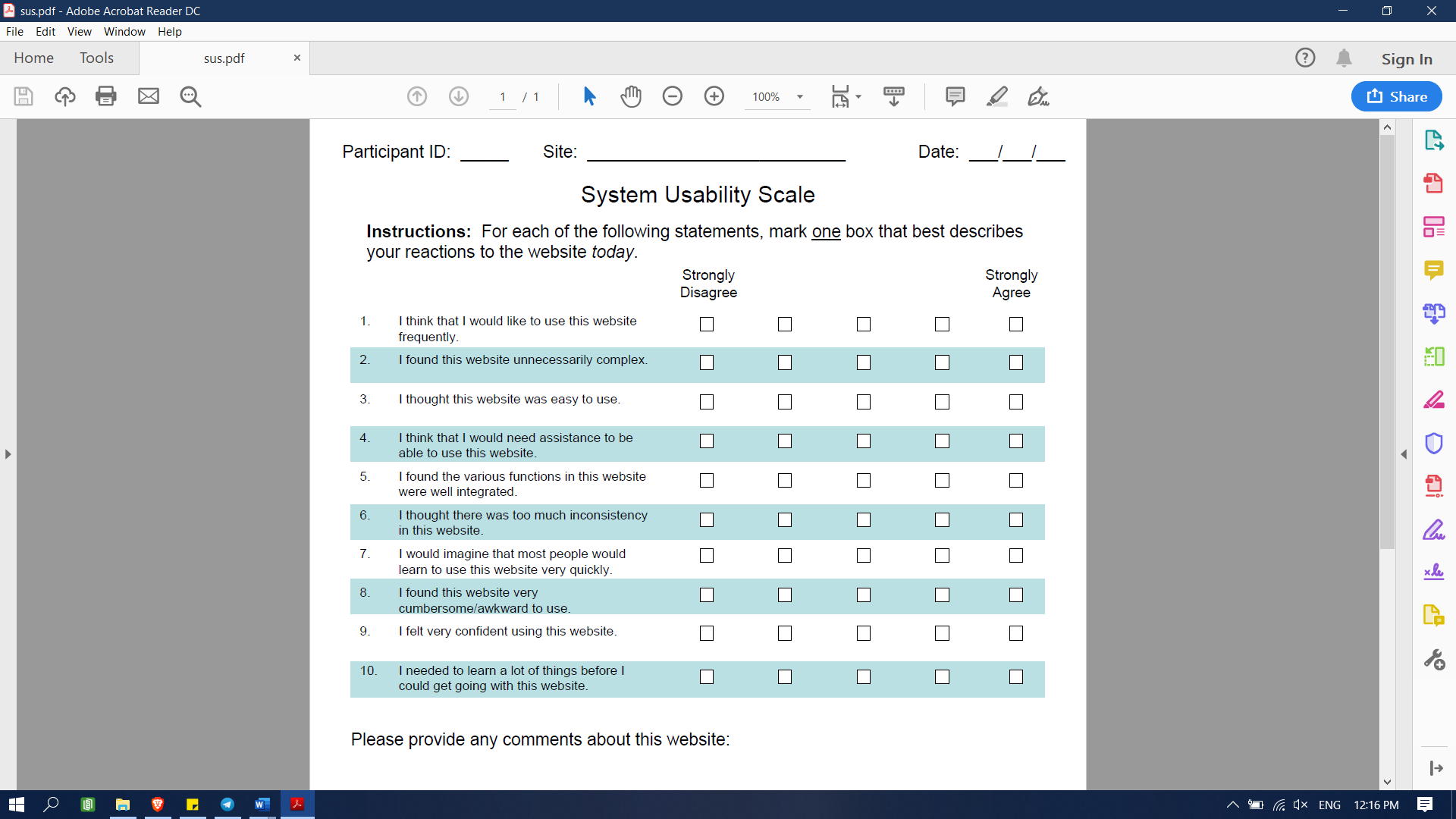
This is NOT a percentage, but it is a clear way of seeing your score. The System Usability Scale is not diagnostic and will not tell what specific problems we face, but it will give us red or green light to know how badly the usability needs work.

If we compare to the overview, the system gets A!

# Discussion

All in all, the system has been designed so that it caters all the need and it is user friendly for the user to use. Most of the participants finds that the system is neatly organized and easy to use. Although it is their first time to use this system, but they manage to fill in all the information and show the attendance list. The designed of the system is minimalist so that it will make the eyes of the user feel comfortable and easy to look.

# Appendix



*Figure 1: SUS that the user will answer after using the system*