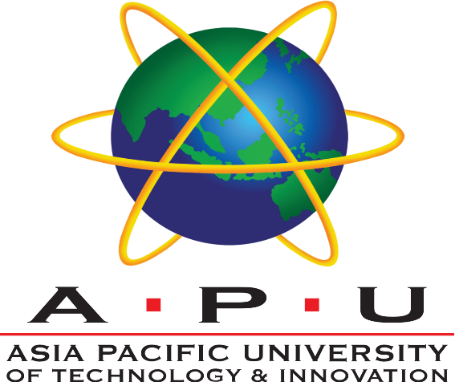
****

**Java Programming (JP)**

**Technology Park Malaysia**

**AAPP004-4-2-JP**

**Individual Assignment**

**Name: SIA CHEE ZHEN (TP056392)**

**Project Name: DOTA 2 MACHINE (D2M)**

**Lecturer: MR. USMAN HASHMI**

**Module Code: AAPP004-4-2-JP (JAVA PROGRAMMING)**

**Intake Code: UCDF1905ICT(SE)**

**Hand-out-Date: 22nd JANUARY 2021**

**Hand-in-Date: 16th APRIL 2021**

***Acknowledgement***

First and foremost, I am feeling grateful and thankful to **The** **Almighty** **God** for enabling us to be able to complete this assignment with no trouble and is a smooth sailing.

Besides that, I, as a team will wish to sincerely thank **Datuk Dr. Parmjit Singh**, ***Chairman of Asia Pacific University***, for providing us with all the necessary facilities and the suitable environment that contributed to the successful and smooth-sailing completion of my assignment.

Aside from that, I would also like to place on record, my sincere gratitude to **Prof. Ron Edwards**, ***Vice Chancellor of Asia Pacific University***, for the speech that he has given us during my first Orientation Day in Auditorium One that has continued to be an inspiration to achieve academic excellence despite all the hardships that I have faced during the journey.

I would also like to thank **Mr Usman Hashmi,** my **Java Programming (*JP*)**, ***Faculty of Computing, Engineering & Technology School of Computing Computer Science & Software Engineering***. I am highly grateful, thankful, and indebted to him for his expert, dedicated and valuable guidance and encouragement that was provided to us that helped us in trying to strive for the best result possible and in accomplishing this arduous but enjoyable assignment.

I would also like to take this opportunity to sincerely thank all my **friends** **and** **family** for their encouragement, support and believing in us that I will be able to accomplish this assignment to completion.

Last but not least, I would like to place my record, sense of gratitude to **everyone** and **all** who, directly or indirectly, have lent their helping hands in ensuring a smooth sailing for the completion of this assignment, thank you.

**Table of Contents**

[**1.0 Abstract** 4](#_Toc69163241)

[**2.0 Discussion on Dota 2 Machine (D2M)** 5](#_Toc69163242)

[**2.1 Introduction to Dota 2 Machine (D2M**) 5](#_Toc69163243)

[**2.2 D2M Sample Outputs** 6](#_Toc69163244)

[**2.2.1 Admin** 6](#_Toc69163245)

[**2.2.1.1 Admin Main Menu** 6](#_Toc69163246)

[**2.2.1.2 Admin Registration** 10](#_Toc69163247)

[**2.2.1.3 Add Item** 15](#_Toc69163248)

[**2.2.2 Customer** 20](#_Toc69163249)

[**2.2.2.1 User Main Menu** 20](#_Toc69163250)

[**2.2.2.2 Check Out** 25](#_Toc69163251)

[**2.3 Discussion of Object-Oriented (OO) Concepts & Java Features** 30](#_Toc69163252)

[**2.3.1 Class** 30](#_Toc69163253)

[**2.3.2 Creation of Objects** 31](#_Toc69163254)

[**2.3.3 Constructor** 33](#_Toc69163255)

[**2.3.4 Access Modifier** 34](#_Toc69163256)

[**2.3.5 Package** 35](#_Toc69163257)

[**2.3.6 Exception Handling** 36](#_Toc69163258)

[**2.3.7 Event Handling** 37](#_Toc69163259)

[**2.4 Additional Features Added** 39](#_Toc69163260)

[**2.4.1 Admin** 39](#_Toc69163261)

[**2.4.1.1 Login** 39](#_Toc69163262)

[**2.4.1.2 Logout** 42](#_Toc69163263)

[**2.4.1.3 Delete Account** 44](#_Toc69163264)

[**2.4.1.4 Edit Password** 48](#_Toc69163265)

[**2.4.1.5 Update Item** 52](#_Toc69163266)

[**2.4.1.6 Delete Item** 57](#_Toc69163267)

[**2.4.1.7 View Past Transaction** 60](#_Toc69163268)

[**2.4.2 Customer** 63](#_Toc69163269)

[**2.4.2.1 Get Item Suggestion** 63](#_Toc69163270)

[**2.5 Assumptions** 64](#_Toc69163271)

[**2.5.1 System-Oriented** 64](#_Toc69163272)

[**2.5.2 Business Model** 64](#_Toc69163273)

[**2.5.3 Miscellaneous** 64](#_Toc69163274)

[**3.0 Reference** 65](#_Toc69163275)

# 

# **1.0 Abstract**

This assignment **documentation** **contains** **detailed** **information** on the **Dota** **2** **Machine** (**D2M**) which is a **vending** **machine** that **sells** **Dota** **2** **Figurine**. The **documentation** **covers** the **Introduction** of the **machine**, **sample** **outputs** of the **machine** of both the **user** **roles** **available**, which are the **normal** **user** or **customer** and **admin** or **staff**, **covering** the **basic** and **important** **functions** **requested** in the **assignment** **questions**, **discussion** of the **Object**-**Oriented** (**OO**) **Concept** and **Java** **features** **implemented** in the **machine**, **additional** **features** **added** (such as **account** **management**, **item** **management** and **security** **feature** **implemented**) and last but not least, an **assumptions** which **covers** the **System**-**Oriented**, **Business** **Model** and **Miscellaneous** **aspects** of the **machine**.

# **2.0 Discussion on Dota 2 Machine (D2M)**

## **2.1 Introduction to Dota 2 Machine (D2M**)

The **Dota** **2** **Machine** (***D2M***) is a **vending** **machine** that **sells** **Dota** **2** **Figurine**, the **vending** **machine** for now can only **accommodate** **6** **items** in total and a **quantity** of **10** for **each** of the **items**. The **D2M** **system** consists of **two** **user** **roles** which are “**Admin** (*or* ***Staff***)” and **Customer** (*or* ***Normal******User***). The **Admin** **role** **possesses** more **functions** **compared** to the **Customer** as the **Admin** **role** is **focused** more on **items** and **admin’s** **accounts** **management** (*which* ***includes******functions*** *such as* ***delete****,* ***update****, and* ***add***). Aside from that, the **admin** can also **view** **past** **transactions** **made** by the **customer** which **includes** **information** such as the ***item******bought****, the* ***quantity******bought****,* ***amount******paid****,* ***total*** *of the* ***transaction****,* ***change******given******back****,* ***time*** *and* ***date*** *the* ***transaction*** was **processed** and many more. **Customer** **role** on the other hand only **possesses** a **handful** of **functions** which are **selecting** the **Dota** **2** **figurines** which they want to **purchase**, **making** **payment** for the **items** and **generating** a **random** **item** **ID**, **providing** them a **suggestion** since all the **figurines** **available** are all **desirable** which leads to the **difficulty** in **making** the **choice**.

A picture containing graphical user interface

Description automatically generated

***Figure******1*** *shows the* ***Dota******2******Machine*** *(****D2M****)* ***User******Main******Menu.***

## **2.2 D2M Sample Outputs**

### **2.2.1 Admin**

#### **2.2.1.1 Admin Main Menu**

Graphical user interface, application

Description automatically generated

***Figure******2*** *shows the* ***D2M’s******Admin******Main******Menu****.*

**Figure** **2** above shows the **Dota** **2** **Machine** (**D2M**)’s **Admin** **Main** **Menu** in which the **main** **menu** is **displaying** all the **items** that are **currently** **available** in the **machine**, followed by the **item’s** **ID**, **name**, **price** and **available** **quantity**. In addition to that, the **admin** **main** **menu** also **contains** many **different** **icons** which when **pressed** **on** can **lead** to **different** **panels** which **serves** many **different** **functions** such as **Register** **Admin**, **Add** **Item**, **Delete** **Item**, **Past** **Transaction**, **Delete** **Account** and **Logout** as shown in **figures** **3**, **4**, **5**, **6**, **7** and **8**. The **output** in this **particular** **page** is the **items’** **information** which are **displayed** after being **retrieved** from the **itemlist.txt file** and the **displaying** of **different** **panels** as **aforementioned**. In addition to that, the **D2M** **logo** at the **top** when **clicked** **on** will **lead** back to the **Main** **Menu** as **shown** in **Figure** **2.** Another output will be the **displaying** of the **welcome** **message** with the **Admin’s** **Username** and **Admin** **ID** **displayed**.

Graphical user interface, application, website

Description automatically generated

***Figure******3*** *shows the* ***Staff/******Admin******Registration******Panel****.*

*Graphical user interface, website

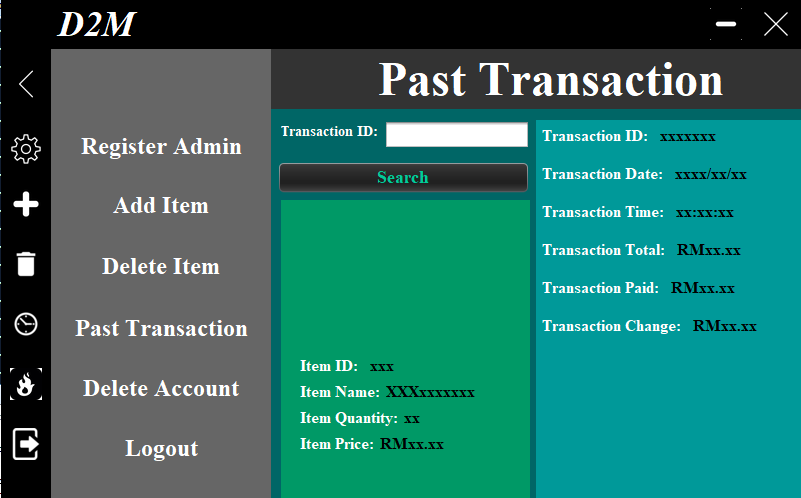
Description automatically generated*

***Figure******4*** *shows the* ***Add******Item******Panel****.*

*Graphical user interface, text

Description automatically generated*

***Figure******5*** *shows the* ***Delete******Item******Panel****.*

**

***Figure******6*** *shows the* ***Past******Transaction******Page****.*

*Graphical user interface, text, website

Description automatically generated*

***Figure******7*** *shows the* ***Delete******Account******Panel****.*

*Graphical user interface, website

Description automatically generated*

***Figure******8*** *shows the* ***Admin******Login******page****.*

#### **2.2.1.2 Admin Registration**

Graphical user interface, application, website

Description automatically generated

***Figure******9*** *shows the* ***Admin/******Staff******Registration******Panel****.*

**Figure** **9** shows the **Admin** (or **Staff**) **Registration** **panel** in which only **existing** **Admins** or **Staffs** with a **valid** **account** could use to **register** their other **fellow** **new** **staff** **members**. The **Registration** **function** is only **accessible** with a **valid** **account** rather than **displaying** it at the **login** **page** is due to **security** **purposes**. Besides that, it is also due to the fact that, the **Admin** **Login** **Page** is **accessible** via the **User** **Main** **Menu** which leads to such **arrangement**. The **valid** **number** of **admins** that are **eligible** for **registration** is only **10** which is **done** to **prevent** **having** **too** **many** **staffs** or **admins** having **access** to the **system** which can lead to **waste** of **space** and to **ensure** that the **security** of the **D2M** is **secured**. This **panel’s** **output** is the **insertion** of **Admin** **Information** (*which* ***includes*** *the* ***registered******Username****,* ***Password*** *and* ***Name***) which is **inputted** into the **text** **fields** **provided** as shown in **Figure** **10** **below** into the **Admin.txt** **file** after clicking the “Register button” which will firstly display a “Successful Registration” message as shown in **Figures** **11** and **12** **below**. In addition, below the “***Register***” button is a **label** which when **upon** **clicking** will **lead** to an **output** of **displaying** of the “***Edit*** ***Password***” **panel**.

*Graphical user interface, application

Description automatically generated*

***Figure******10*** *shows the* ***Admin/******Staff******Registration******panel*** *with* ***User******Inputs****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******11*** *shows the* ***Successful******Registration******Message****.*

*Graphical user interface, application

Description automatically generated*

***Figure******12*** *shows the* ***Registered******User****, “****John****” in the* ***Admin.txt******file****.*

In addition to that, **other** **outputs** also **include** the **error** **messages** as shown in **Figures** **13**, **14**, **15** and **16** that will be **displayed** when **certain** **conditions** (*such as* ***empty******fields****,* ***username******characters*** *are* ***less******than******3******characters*** *or* ***more******than******5******characters****, if* ***Username******inputted*** *already* ***existed*** *or if the* ***Maximum******Number*** *of* ***Admin******Slot*** *has* ***reached***) are **met**.

Graphical user interface, website

Description automatically generated

***Figure******13*** *shows the* ***Error******Message******displayed*** *after* ***empty******fields*** *are* ***detected****.*

Graphical user interface, website

Description automatically generated

***Figure******14*** *shows the* ***Error******Message******displayed*** *after the* ***Username*** *is* ***less*** *than* ***3******characters*** *or* ***more*** *than* ***5******characters****.*

Graphical user interface, website

Description automatically generated

***Figure******15*** *shows the* ***Error******Message******displayed*** *after the* ***Username******inputted*** *already* ***existed****.*

*Graphical user interface, text, website

Description automatically generated*

***Figure******16*** *shows the* ***Error******Message******displayed*** *after* ***Max******Number*** *of* ***Admin******Slot*** *has* ***reached****.*

#### **2.2.1.3 Add Item**

Graphical user interface, website

Description automatically generated

***Figure******17*** *shows the* ***Add******Item******panel****.*

**Figure** **17** shows the **Add** **Item** **panel** in which **existing** **Admin** or **Staff** with a **valid** **account** can **add** **items** into the **vending** **machine** by **filling** **up** the **text** **fields** **required**. The **maximum** **number** of **items** that can be **inserted** is **6.** After making sure that the there is still **empty** **slot** in the **D2M** to be **filled**, **filling** **up** the **required** **fields** and **clicking** on the “**Add** **Item**” **button**, **one** of the **many** **outputs** which is the **insertion** of the **details** into the **itemlist.txt** **file** onto the **lines** which are **empty** after the **Successful** **Added** **Item** **message** is **displayed** as shown in **Figures** **18**, **19** and **20**. Whereas, in **Figure** **21** after **redirecting** **back** to the **User** **Main** **Menu**, the just **added** **item’s** **information** are **displayed** in one of the **card-like** **panels**.

Graphical user interface, website

Description automatically generated

***Figure******18*** *shows the* ***Add******Item******panel*** *with* ***user******inputs****.*

*Graphical user interface, application, website

Description automatically generated*

***Figure******19*** *shows the* ***Add******Item******Successful******message****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******20*** *shows the* ***itemlist.txt*** *with the* ***new******item******inputs******added****.*

*Graphical user interface, application

Description automatically generated*

***Figure******21*** *shows the* ***User******Main******Menu*** *with the* ***Added******Item******displayed*** *on the* ***Admin******Main******Menu****.*

Other **outputs** also **include** the **error** **message** as shown in **Figures** **22**, **23**, **24** and **25** that will be **displayed** after **certain** **conditions** (*such as* ***empty******fields****,* ***Item******Name*** *is* ***less*** *than* ***4******characters*** *or* ***more*** *than* ***10******characters****,* ***Item******Price*** *is not in the “****0.00****” or “****00.00****”* ***format*** *and if all the* ***6******item******slots*** *are already* ***filled******up***) are **met**.

Graphical user interface, website

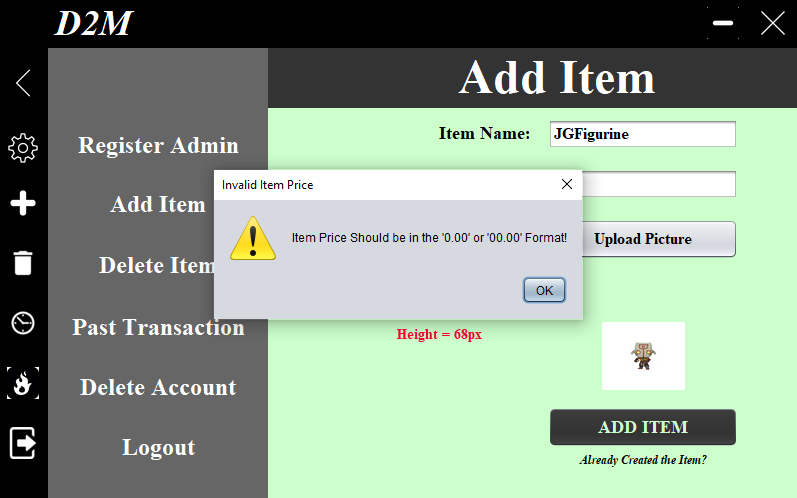
Description automatically generated

***Figure******22*** *shows the* ***Error******Message******displayed*** *due to* ***empty******fields****.*

*Graphical user interface, website

Description automatically generated*

***Figure******23*** *shows the* ***Error******Message******displayed*** *due to* ***Item******Name******less******than******5******characters*** *or* ***more******than******10******characters****.*

**

***Figure******24*** *shows the* ***Error******Message*** *due to* ***Item******Price******not******following*** *the “****0.00****” or “****00.00****”* ***format****.*

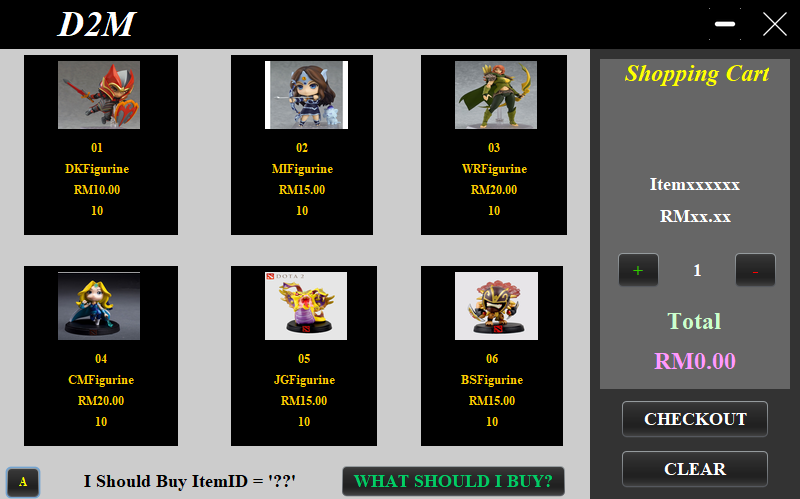
*Graphical user interface, text, website

Description automatically generated*

***Figure******25*** *shows the* ***Error******Message******displayed*** *due to* ***Maximum******Item******Slot******reached****.*

### **2.2.2 Customer**

#### **2.2.2.1 User Main Menu**



***Figure******26*** *shows the* ***User******Main******Menu****.*

**Figure** **26** shows the **User** **Main** **Menu** where the **customer** can **choose** the **item** which they would like to **purchase** by **clicking** on the **card-like panels**. After **clicking** on the **item**, the **item** **details** (*such as the* ***item******name****,* ***item******price*** *and* ***total******price******calculated***) would be **displayed** on the **Shopping** **Cart** **section** as shown in **Figure** **27** **below**. The “***Plus***” and “***Minus***” **icon** is used to **add** or **reduce** the **quantity** of the **item** **selected** which will also **lead** to the **reduction** or **increase** of the **calculated** **total** **price** as shown in **Figure** **28**. After confirming the **item** **selected** and its **quantity** is **correct**, the customer just has to **click** on the “**CheckOut**” **button** to **direct** to the **Check** **Out** **page** as shown in **Figure** **29**. In addition to that, a “***A***” **button** which when **clicked** will **lead** to the **Admin** **Login** **page** as shown in **Figures 28** and **30.**

Graphical user interface

Description automatically generated

***Figure******27*** *shows the* ***Item******01****, “****DKFigurine****”****’s******information*** *which is* ***selected*** *being* ***displayed*** *in the* ***Shopping******Cart******Section****.*

*Graphical user interface

Description automatically generated*

***Figure******28*** *shows the* ***calculated******total******price******increases******linearly*** *to the* ***increase*** *in* ***quantity*** *of the* ***item****.*

*Graphical user interface, application

Description automatically generated*

***Figure******29*** *shows the* ***Check******Out******page****.*

*Graphical user interface, website

Description automatically generated*

***Figure******30*** *shows the* ***Admin******Login******page****.*

Other **outputs** (aside than **redirecting** to **different** **other** **pages**, **retrieving** **item’s** **information** into the **shopping** **cart**, **getting** **suggestion** for **item** to **purchase,** and **getting** the **total** **price**) also includes that **error** **message** that will be **displayed** when the **condition** (which is when the **shopping cart** is **empty**, **item** **selected** is **unavailable** and if the **item** **selected’s** **quantity** is “**out** **of** **stock**”) is met as shown in **Figures** **31**, **32 and 33**.

Graphical user interface, website

Description automatically generated

***Figure******31*** *shows the* ***Error******Message******displayed*** *due to* ***Shopping******Cart*** *being* ***empty****.*

*Graphical user interface, website

Description automatically generated*

***Figure******32*** *shows* ***Error******Message******displayed*** *due to* ***Selecting*** *an* ***Item*** *that is an* ***empty******slot****.*

*Graphical user interface

Description automatically generated*

***Figure******33*** *shows* ***Error******Message*** *being* ***displayed*** *due to* ***Selecting*** *an* ***Item*** *that is* ***Out*** *of* ***Stock****.*

#### **2.2.2.2 Check Out**

Graphical user interface

Description automatically generated with medium confidence

***Figure******34*** *shows the* ***User******Main******Menu******page*** *with an* ***Item******Selected****.*

*Graphical user interface, application

Description automatically generated*

***Figure******35*** *shows the* ***Check******Out******page*** *with the* ***Item******Selected******Information******displayed****.*

**Figure** **34** shows the **User** **Main** **Menu** which **contains** the **Selected** **Item’s** **information** being **displayed** in the **Shopping** **Cart** **section** in which after the **customer** **confirm** **it’s** **correct**, the **customer** will then **click** on the “**CheckOut**” **button** to **direct** to the **Check** **Out** **page** as shown in **Figure** **35** **above** in which all the **previous** **item’s** **information** will be **displayed** in the **Check** **Out** **page**. All the **customer** has to do now is **input** the **amount** that they would like to make into the **pay text field** as shown in **Figure 36**. After **inputting,** all the **customer** has to do is **click** on anywhere on the **page** or the “***Click Here For Fun***” **picture** as shown in **Figure** **36** to get to **know** how much their **change** would be. Finally, to **get** their **item** to be **dispensed**, all they have to do is click on the “***Confirm******&******Pay***” **button** which was **previously** **disabled** as shown in **Figure** **35** and is now **enabled** as shown in **Figure** **36** to **finalize** the **payment** and **confirm** their **purchase**. The **output** in this case is the **Change** **calculated** by **subtracting** the **input** in the **pay** **text** **field** with the **input** in the **total** **text** **field** which will be **displayed** in an **informative** **message** **form** as shown in **Figure** **37**. In addition to that, after the **payment**, a **successful payment** and **dispensing** **item** **message** will be **displayed** as shown in **Figure** **38** and after the **user** **pressed** “***ok***”, another **successful** **message** about **item** **dispensing** will be **displayed** and after the **customer** **pressed** “***ok***” again, the **customer** will be **redirected** **back** to the **User** **Main** **Menu** as shown in **Figure** **39**.

A screenshot of a computer

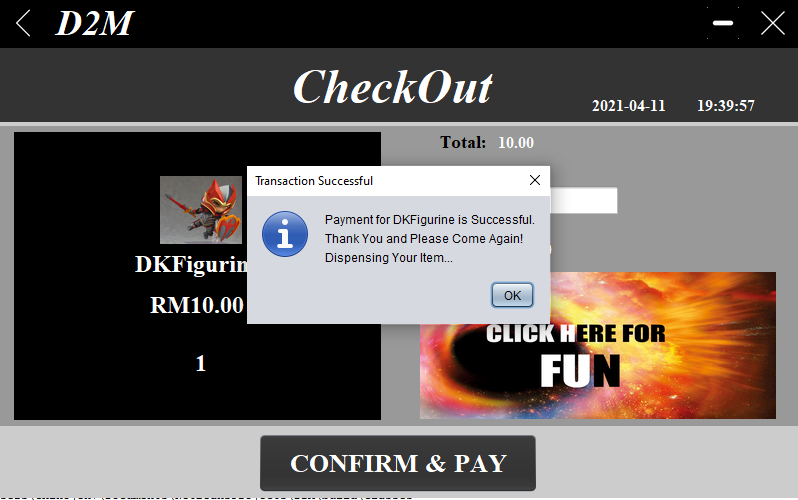
Description automatically generated with medium confidence

***Figure******36*** *shows the* ***Check******Out******page*** *with the* ***pay******text******field******inputted*** *and the “****Confirm******&******Pay****”* ***button******enabled****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******37*** *shows the* ***Transaction******Change******displayed*** *in an* ***Informative******Message******form****.*

**

***Figure******38*** *shows the* ***Transaction******Successful******Message****.*

*A screenshot of a computer

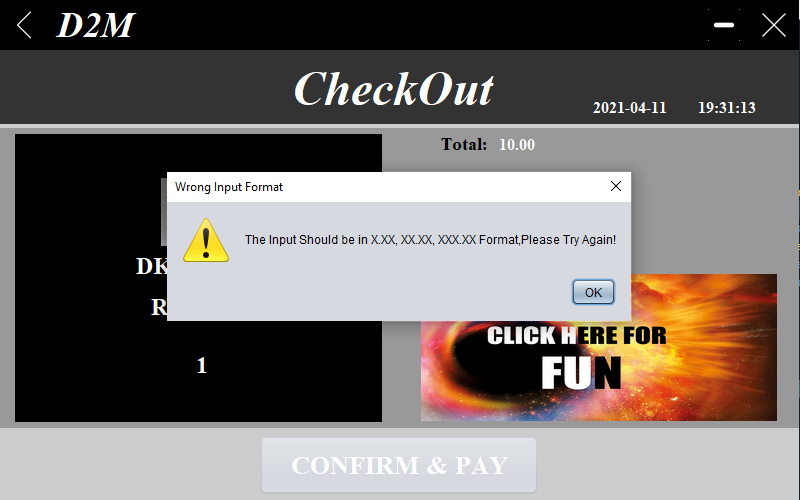
Description automatically generated with medium confidence*

***Figure******39*** *shows the* ***Item******Dispense******Successful*** *and* ***Redirecting******message****.*

Other **outputs** here also **include** the **error** **messages** that will be **displayed** **upon** **meeting** **certain** **conditions** (*such as* ***empty******field,*** *the* ***input*** *of* ***pay******text******field*** *is* ***not*** *in the “****0.00****”, “****00.00****” or “****000.00****”* ***format*** *and* ***Value*** *in the* ***pay******text******field*** *is* ***lower*** *than of the* ***value*** *in the* ***total******text******field***) as shown in **Figure**s **40**, **41** and **42**.



***Figure******40*** *shows* ***Error******Message******displayed*** *due to* ***empty******fields****.*

**

***Figure******41*** *shows* ***Error******Message******displayed*** *due to* ***input*** *in* ***pay******text******field*** *not in the “****0.00****”, “****00.00****” or “****000.00****”* ***format****.*

*Graphical user interface, text, application, chat or text message

Description automatically generated*

***Figure******42*** *shows* ***Error******Message******displayed*** *due to* ***input*** *in* ***Total*** *is* ***higher*** *than in* ***Pay******text******field****.*

## **2.3 Discussion of Object-Oriented (OO) Concepts & Java Features**

### **2.3.1 Class**

A **Class** is an **Object** **Constructor** or a **blueprint** that **contains** **variables** (*or* ***also******known*** *as* ***attributes***), **methods** and many more which is **utilized** in the **creation** of **objects** (*Gibbs, 2018 and w3schools, n.d.*). Take the **code** **shown** in **Figure** **43** down below as an **example**, the **object** here is **CheckOut** which is **related** to **payment** **matters**, but to **structure** the **object** to **accommodate** **payment** **matters**, a **class** **called**, “**CheckOut**” is **created**. As can be seen, the class created **contains** many other **variables** **declared** as shown in **Figure** **43** which are “***saveDir***”, “***SelectedItemID***”, “***DisplayTotalPrice***” and many more. Other **examples** of **methods** that can be **observed** from the **CheckOut** **page** are “***DisplayTotalPrice()***”, “***DisplayDate()***”, “***DisplayTime()***” and “***CalculateChange()***” which are as shown in **Figure** **44** **down** **below**.

A screenshot of a computer

Description automatically generated with medium confidence

***Figure******43*** *shows some* ***examples*** *of* ***CheckOut******class’s******variables****.*

*Text

Description automatically generated*

***Figure******44*** *shows some* ***examples*** *of* ***CheckOut******class’s******methods****.*

### **2.3.2 Creation of Objects**

**Creation** of **Objects** **refers** to the **creating** of **objects** which **serve** as the **basic** **building** **blocks** of an **OOP** **Language** and is **important** due to the fact that **without** **it**, **program** is **inexecutable** (*javaTpoint, n.d.*). There are **five** **ways** of **creating** **an** **object** which are “***new Keyword***”, “***clone() method***”, “***newInstance() method of Class class***”, “***newInstance() method of Constructor class***” and “***Deserialization***” (*javaTpoint, n.d.*). Though only **one** **method** is **used** in the **D2M** which is the **first** **method**, “***new******Keyword***”. Take the **code** **shown** in **Figure** **45** as an **example**, **many** **new** **objects** are being **created** which are “***file***” (*which* ***represents*** *the* ***object****,* ***File***), “***fr***” (*which* ***represents*** *the* ***object****,* ***FileReader***) and “***br***” (*which* ***represents*** *the* ***object****,* ***BufferedReader***). It is also worth noting that this is **commonly** **implemented** **throughout** the **codes** of **other** **classes** as shown in **Figures** **46’s** **getRandomItemID()** **method’s object**, “***Random***” and **47’s** **declaration** of **new** **object**, “***DecimalFormat***”.

**Text

Description automatically generated**

***Figure******45*** *shows the* ***EmptySlotWarning()******method*** *of the* ***User******Main******Menu’s******code****.*

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure******46*** *shows the* ***getRandomItemID()******method*** *of the* ***User******Main******Menu’s******code****.*

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure******47*** *shows the* ***UserMainMenu’s******class’s******declaration*** *of* ***variables’ code****.*

### **2.3.3 Constructor**

A **Constructor** is a **unique** **method** which is **utilized** for the **initialization** of **Objects** and is **commonly** **called** when an **object** of a **class** is **created** and can be used to **set** the **initial** **values** for **variables** (*w3school, n.d.)*. It is also **worth** **noting** that **Constructor** has the **same** **name** as the **class** (as shown in **Figure** **48** where the **AdminMainMenu** **Class’s** **Constructor** has the **same** **name** as the **Class’s**) and is **syntactically** **similar** to a **method**, **except** for the **fact** that it has **no** **explicit** **return** **type** (tutorialpoints, n.d.). In addition to that, as can be **observed** in **Figure** **48,** **positioning** the **JFrame** to a **certain** **location** (*which in* ***this case****, “****this.setLocationRelativeTo(null)****”* ***sets*** *the* ***location*** *of the* ***JFrame*** *to the* ***centre*** *of the* ***screen***), **changing properties** ofa **Component** (*which in* ***this case*** *is* ***setting*** *the* ***JPanels******visibility******property*** *to* ***false***)or **methods** (*which in* ***this case*** *is* ***getItemData()*** *and* ***GetUsername()*** ) that are **required** to be **executed** **during** the **initialization** of the **application** can be **done** so by **adding** **it** **within** the **Constructor**.

Text

Description automatically generated

***Figure******48*** *shows the* ***AdminMainMenu******Class’s******code****.*

### **2.3.4 Access Modifier**

**Access** **Modifier** **specifies** the **accessibility** or **scope** of a **field**, **constructor**, **method**, or **class** and the **four** **access** **modifiers available** are “***Private***” (*which* ***access******level*** *is only* ***within*** *the* ***class*** *and* ***cannot*** *be accessed from* ***outside***), “***Public***” (*which* ***access******level*** *is everywhere, can be* ***accessed*** *from* ***within*** *and* ***outside*** *the* ***class*** *and* ***package***) “***Protected***” (*which* ***access******level*** *is* ***within*** *and* ***outside*** *the* ***package*** *through* ***child******class***) and “***Default***” (*which* ***access******level*** *is* ***only******within*** *the* ***package***) (*javaTpoint, n.d. and GeeksforGeeks, 2017*). Take the **code** in **Figure** **49** as an **example** where the **access** **modifier** for the **UserMainMenu** **class** is “**Public**” . Thus, the **class** is **accessible** **within** and **outside** the **package** and the **class**. Though it is also **worth** **noting** that **almost** **all** the **methods** **within** the **class** are using a “***Private***” **access** **modifier** which makes those **methods** **inaccessible** **outside** **package** or the **class** despite the **fact** that the **UserMainMenu** **class** is **set** to “***Public***”.

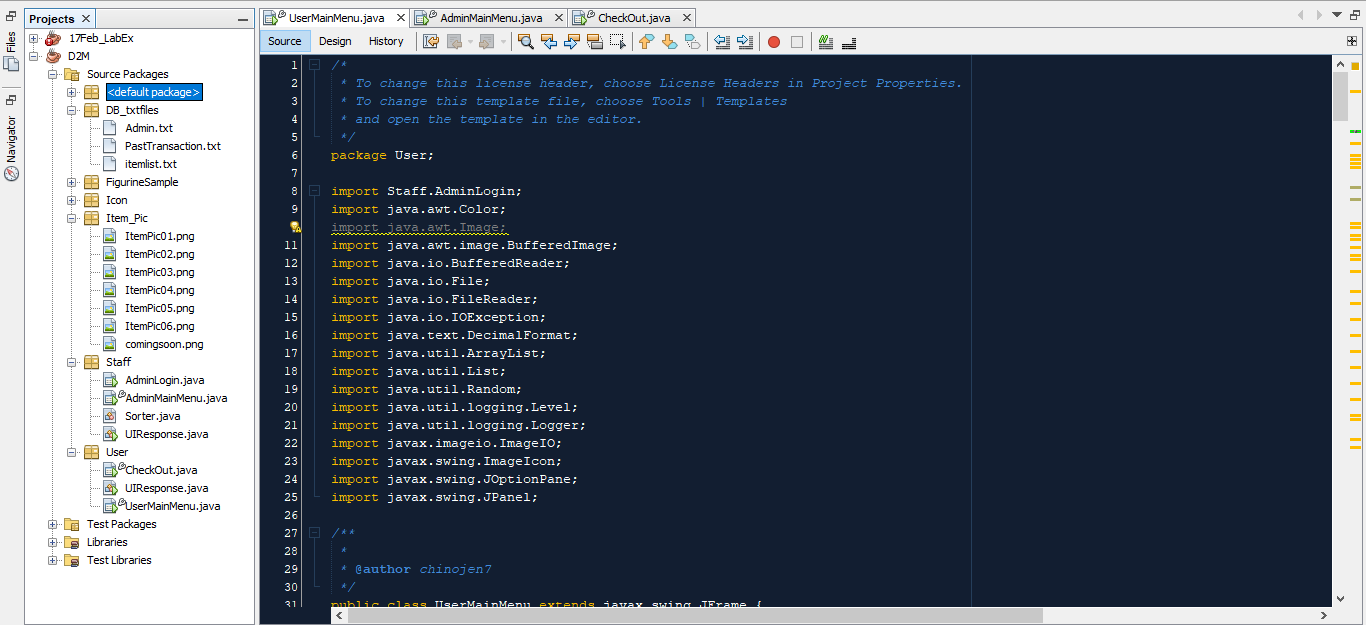
Text

Description automatically generated

***Figure******49*** *shows the* ***UserMainMenu******class’s******code****.*

### **2.3.5 Package**

A **Package** is used for the **grouping** of **related** **classes, interfaces, annotations and enumerations** or simply to say, it is a **folder** in a **file** **directory** which **purpose** is to **avoid** **name** **conflicts**, allow **proper** **organization** of **classes** and to **allow** the **writing** of a **better** **maintainable** **code**. **Package** is **divided** into **two** **different** **categories** which are “***Built-in-Packages***” (*which is* ***packages*** *from the* ***Java******API***) and “***User****-****defined******Packages***” (*which is* ***created*** *by the* ***User***) (*w3schools, n.d. and tutorialpoints, n.d.*). As shown in **Figure** **50**, the **vending** **machine** **project**, **D2M** **contains** many **packages** for the **organization** of **different** **classes, txt files** and **picture files**, such as the “***DB\_txtfiles***” which stores all **txt** **files** that **contains** **item** and **admin** **data**, “***FigurineSample***”, “***Icon***” and “***Item\_Pic***” storing **useful** **picture** **files**, “***Staff***” storing **staff-related classes** and “***User***” storing **user-related classes**. By doing so, it also **gives** the **project** a **sense** of **proper** **structure** which is **easier** for **other** **developers** who are **not** **part** of the **project** to **easily** **interpret** and **understand** the **code**.



***Figure******50*** *shows the* ***Project******Directory*** *for* ***D2M*** *with its* ***Packages****.*

### **2.3.6 Exception Handling**

**Exception** **Handling** can be **defined** as one of the **powerful** **mechanisms** for the **handling** of **runtime** **errors** which **leads** to the **maintaining** of a **normal** **flow** of the **application**. Examples of **Exception** **Handling** are “***ClassNotFoundException***”, “***RemoteException***”, “***IOException***”, “***ArrayIndexOutOfBoundsException***” and many more. In addition to that, there are **many** **statements** that can are **used** for **Exception** **Handling** which are the “***try-catch***” **statement** (which the “***try***” **allows** for the **testing** of a **block** of **code** for **error** **during** **execution** and the “***catch***” **allows** for the **execution** of a **block** of **code** if an **error** **occurs** in the “***try***” **block** as shown in **Figure** **51** **down** **below**) and “***throw***” **statement** (which allows for the **creation** of a **custom** **error** as shown in **Figure** **52** **down** **below**) (*w3schools, n.d. and javaTpoint, n.d.*). As can be **observed** in **Figure** **51**, the **try** **statement** would **firstly** **execute** the **block** of **code** as **highlighted** in the **red** **rectangle** and if an **error** **occurred**, the **block** of **code** **highlighted** in the **green** **rectangle** would be **executed**. Whereas in **Figure** **52**, it can be observed that **few** **custom** **errors** have been **created** under the **RegistrationEmptyField() method** which are “***Empty New Admin Username!***”, “***Empty New Admin Password Value!***” and “***Empty New Admin Real Name!***”.

Text

Description automatically generated

***Figure******51*** *shows the* ***AdminMainMenu’s******endSession()******method’s******code*** *with the “****try-catch****”* ***statement******used****.*

*Text

Description automatically generated*

***Figure******52*** *shows the* ***AdminMainMenu’s******RegistrationEmptyField()******method*** *with the “****throw****”* ***statement******used****.*

### **2.3.7 Event Handling**

**Event** **Handling** is **consisted** of **Event** (*which is the* ***change*** *of the* ***state*** *of an* ***object****, such as* ***clicking******on******button****,* ***dragging******mouse****,* ***lost******focus*** *and etc*) and **Listener** which are **provided** by the **java.awt.event package** in the **form** of **Event** **Classes** and **Listener** **Interfaces** as shown in **Figure** **53** down **below** (*Mathur, 2020 and javaTpoint, n.d.*). Take **Figure** **54’s** **code** as an **example** in which a **MouseClicked**, an **event** **part** of the **Event** **Class**, “**MouseEvent**” is used to **set** the **JPanels’** **visibility** to “**true**” and “**false**”. If there was **no** **MouseClicked** **event** **detected**, the **block** of **codes** in the **event** would **not** be **triggered.**

Table

Description automatically generated

***Figure******53*** *shows the* ***Event******Classes*** *and* ***Listener******Interfaces*** *of* ***Event******Handling*** *(javaTpoint, n.d.).*

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure******54*** *shows the* ***lblEditPasswordMouseClicked******event*** *in the* ***AdminMainMenu******class****.*

The **code** shown in **Figure** **55** **represents** an **example** of the **Event** **Class**, “**FocusEvent**” and the **event** that is being **detected** here is when the **customer** **lost** **focus** from the **tfPay** or **Pay** **text** **field** which will then **initiate** a **method** called, “**CalculateChange()”.** **Whereas** if **Customer** **remains** in the **text** **field**, the **method** would **not** be **initiated** as the **event**, “**Focus** **Lost**” is **not** **detected**.

*Graphical user interface, text

Description automatically generated with medium confidence*

***Figure******55*** *shows the* ***tfPayFocusLost******event*** *in the* ***CheckOut******class****.*

## **2.4 Additional Features Added**

### **2.4.1 Admin**

#### **2.4.1.1 Login**

Graphical user interface, website

Description automatically generated

***Figure******56*** *shows the* ***Admin******Login******page****.*

**Figure** **56** shows the **Admin** (*or* ***Staff***) **Login** **page** in which **Admins** with a **valid** **account** and **correct** **credentials** can **login** to by **inputting** their **username** and **password** into the **text** **fields** **provided** and **clicking** on the “***Login***” **button** to **validate** the **inputs** **provided**. If the **login** **credential** **provided** is **correct**, the user will be **firstly** **greeted** with a **welcome** **back** **message** as shown in ***Figure 57*** and then **later** be **redirected** to the **Admin** **Main** **Menu** as shown in ***Figure 58***. Whereas if the **login** **credential** **provided** is **false**, **error** **messages** will be **displayed** **depending** on **certain** **conditions** which are **empty** **fields** as shown in ***Figure 59*** and **wrong** **username** and **password** as shown in ***Figure 60***.

Graphical user interface

Description automatically generated

***Figure******57*** *shows the* ***Welcome******Back******message****.*

*Graphical user interface, application

Description automatically generated*

***Figure******58*** *shows the* ***Admin******Main******Menu****.*

*Graphical user interface, application

Description automatically generated*

***Figure******59*** *shows* ***Error******Message******displayed*** *due to* ***empty******fields****.*

Graphical user interface, application

Description automatically generated

***Figure******60*** *shows the* ***Error******Message******displayed*** *due to* ***Wrong******Username*** *or* ***Password****.*

#### **2.4.1.2 Logout**

Graphical user interface, application

Description automatically generated

***Figure******61*** *shows the* ***Admin******Main******Menu*** *and the* ***Logout******Message******prompt****.*

**Figure** **61** shows the **Admin** **Main** **Menu** and the **logout** **message** **prompt** which is **displayed** after the **user** **clicked** on the “***Logout***” **button**. The **prompt** would then **require** the **user** to **click** on the “***Yes***” option if they **would** **like** to **logout** and “***No***” if they **would** **not** **like** **to**. The **prompt** is **implemented** to **prevent** **Admins** from **accidentally** **pressing** the **Logout** **button**. If the **Admin** **chose** “***Yes***”, the **Admin** will be **redirected** back to the **Admin** **Login** **page** with a **Successful** **Logout** **message** being **displayed** **beforehand** as shown in **Figure** **62**. Whereas if the **Admin** **chose** “***No***”, the **prompt** will be **cleared** and the **user** would still **remain** in the **Admin** **Main** **Menu**. Besides than **clicking** on the “**Logout**” **button** to **logout,** the **Admin** can just **click** on the “**Exit**” **button** at the **top** **right** which will **start** with **displaying** the **same** **prompt** as shown in **Figure** **61**. Upon **clicking** “**Yes**”, **instead** of **redirecting** the **Admin** **back** to the **Admin** **Login** **page**, the **application** will **close** with another **Goodbye** **prompt** **displayed** **beforehand** as shown in **Figure** **63**.

Graphical user interface, application

Description automatically generated

***Figure******62*** *shows the* ***First******Version*** *of* ***Logout******Successful******Message******prompt****.*

*Graphical user interface, application

Description automatically generated*

***Figure******63*** *shows the* ***Second******Version*** *of* ***Logout******Successful******Message******prompt****.*

#### **2.4.1.3 Delete Account**

Graphical user interface, website

Description automatically generated

***Figure******64*** *shows the* ***Admin******Main******Menu*** *with the* ***Delete******Account******panel******displayed****.*

**Figure** **64** shows the **Delete** **Account** **panel** which can be **accessed** by **clicking** on the “**Delete** **Account**” **icon** at the **left** **side** as shown in **Figure 64**. This **panel** is for **Admin** who **wants** to **delete** their **account** or other **admins’** **accounts** can be done by **inputting** the **correct** **ID**, **username**, **password** and then **clicking** on the “***Submit***” **button** for **validation** as shown in **Figure** **64**. If the **inputs** **provided** is **true**, the **account** will be **deleted** and a **successful** **delete** **message** will be **displayed** as shown in **Figure** **65**. Whereas if the **inputs** **provided** is **false**, an **error** **message** will be **displayed** as shown in **Figure** **66**. In addition to that, if the **text** **fields** are **empty**, an **error** **message** will be **displayed** as shown in **Figure** **67**. In addition to that, if **Admin inputted** “**\***” into the, **ID**, **Username** and **Password** **text** **fields** which are **illegal**, an **error** **message** will be **displayed** as shown in **Figure** **68**. If **Admin** try to **login** into their **account** using the **deleted** **account’s** **credentials**, the **Admin** **will not** be able to **access** as shown in **Figure** **69**. As shown in **Figure** **70**, the **inputted** **account** for **deletion** has been **removed** and **cannot** be **found** in the **Admin.txt** **file**.

*Graphical user interface, text, application, chat or text message

Description automatically generated*

***Figure******65*** *shows the* ***Successful******Delete******Account******message****.*

*Graphical user interface, text, chat or text message

Description automatically generated*

***Figure******66*** *shows the* ***Error******Message******displayed*** *due to* ***wrong******username*** *or* ***password****.*

*Graphical user interface, text, application, chat or text message

Description automatically generated*

***Figure******67*** *shows* ***Error******Message******displayed*** *due to* ***empty******fields****.*

*Graphical user interface, text, chat or text message

Description automatically generated*

***Figure******68*** *shows the* ***Error******Message*** *when “****\*****” is* ***inputted****.*

*Graphical user interface, application

Description automatically generated*

***Figure******69*** *shows* ***Error******Message******displayed*** *when trying to* ***login*** *by* ***using*** *the* ***deleted******account******credentials****.*

*Graphical user interface, application

Description automatically generated*

***Figure******70*** *shows the* ***updated******Admin.txt*** *with the* ***inputted******account*** *for* ***deletion******deleted****.*

#### **2.4.1.4 Edit Password**

Graphical user interface, text, website

Description automatically generated

***Figure******70*** *shows the* ***Edit******Password******panel****.*

As shown in **Figure** **71** is an **Edit** **Password** **panel** in which **Admin** who wants to **edit** the **password** for their **account** can **do** it by **accessing** the **panel** by **clicking** on the **edit** **password** **label** under the **Register** **Admin** **panel** as shown in **Figure** **72**. After **reaching** the **panel**, the **Admin** is **required** to **fill** **up** the **Username**, **Old** **Password** and **New** **Password** **Text** **fields**, and after that all is **required** is **clicking** on the “**Edit**” **button** which will **firstly** **validate** the **username** and **old** **password**. If the inputs are true, the **new** **password** will be **updated** and **replace** the **old** **password** in the **txt** **file** and a **successful** **edit** **password** **message** will be **displayed** as shown in **Figure** **73**. **Whereas** if the **inputs** are **false**, the **new** **password** will not be **updated** and an **error** **message** will be **displayed** as shown in **Figure** **74**. In addition to that, if the **empty** **fields** are **empty**, an **error** **message** as shown in **Figure** **75** will be **displayed**. Whereas if the **old** and **new** **edit** **password** is the **same**, an **error** **message** will be **displayed** as shown in **Figure** **76**. As can be seen in **Figure** **77**, the **password** for the **account**, **“*John*”** **which** **password** **previously** was **4** **characters** is now **5** **characters**.

Graphical user interface, text, application, website

Description automatically generated

***Figure******72*** *shows the* ***Admin****/* ***Staff******Registration******panel****.*

Graphical user interface, text, application, chat or text message

Description automatically generated

***Figure******73*** *shows the* ***Successful******Edit******Password******message****.*

*Graphical user interface, text, application, chat or text message

Description automatically generated*

***Figure******74*** *shows the* ***Error******Message******displayed*** *due to* ***wrong******Username*** *or* ***Old******Admin******Password****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******75*** *shows the* ***Error******Message******displayed*** *due to* ***Empty******Fields****.*

*Graphical user interface, text, application

Description automatically generated*

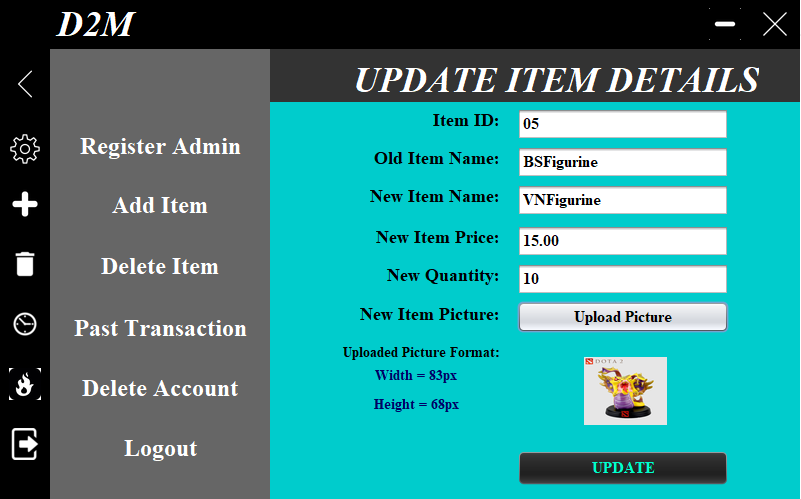
***Figure******76*** *shows the* ***Error******Message******displayed*** *due to* ***Old Edit Password*** *same as* ***New******Edit******Password****.*

*Graphical user interface, application

Description automatically generated*

***Figure******77*** *shows the* ***Admin.txt******file*** *with the* ***Account’s******Password*** *for “****John****”* ***changed*** *to* ***John1*** *which is* ***5******characters****.*

#### **2.4.1.5 Update Item**



***Figure******78*** *shows the* ***Update******Item******Details******panel****.*

As shown in **Figure** **78** is the **Update** **Item** **Details** **panel** which can be **accessed** by **clicking** on the **labels** below the “***Add*** ***Item***” **button** in the **Add** **Item** **panel** as **shown** in **Figure** **79** **down** **below**. The **purpose** of this **panel** is for the **Admin** (or **Staff**) can **update** the **Item** **Details** by **inputting** the **required** **details** into the **text** **fields** **provided** as **shown** **above** in **Figure** **78**. Once the **inputs** have been **inputted**, the **Admin** just has to **click** on the “**Update**” **button** to **validate** the **inputs**. If **all** the **inputs** are **correct**, a **successful** **update** **message** will be **displayed** as shown in **Figure** **80**. Whereas if the **inputs** **do** **not** **meet** **certain** **conditions** (*which are* ***empty******fields****,* ***new******Item******Name*** *is* ***less*** *than* ***5******characters*** *and* ***more*** *than* ***10******characters****,* ***Old******Item******Name*** *and* ***Item******ID******cannot*** *be* ***found*** *in the* ***database****,* ***New******Quantity******cannot******exceed******10*** *and* ***new******Item******Price*** *must* ***follow*** *the “****0.00****”, “****00.00****”* ***price******format***), **error** **messages** will be **displayed** as shown in **Figure** **81**, **82**, **83**, **84** and **85**. As can be **seen** in **Figure** **86**, the **item** **name** for **ID**, “***05***” in the **itemlist.txt** has been **changed** to “***VNFigurine***”.

Graphical user interface

Description automatically generated

***Figure******79*** *shows the* ***Add******Item******panel****.*

*Graphical user interface, website

Description automatically generated*

***Figure******80*** *shows the* ***Successful******Updated******Item******Message****.*

*Graphical user interface, website

Description automatically generated*

***Figure******81*** *shows* ***Error******Message******displayed*** *due to* ***empty******fields****.*

*Graphical user interface, website

Description automatically generated*

***Figure******82*** *shows* ***Error******Message******displayed*** *due to* ***New******Item******Name******less*** *than* ***5******characters*** *and* ***more*** *than* ***10******characters****.*

*Graphical user interface, text, website

Description automatically generated*

***Figure******83*** *shows* ***Error******Message******displayed*** *due to* ***Old******Item******Name*** *or* ***Item******ID******provided*** *is* ***false.***

*Graphical user interface, website

Description automatically generated****Figure******84*** *shows* ***Error******Message******displayed*** *due to* ***New******Item******Quantity******exceeding******10****.*

******

***Figure 85*** *shows the* ***Error******Message******displayed*** *due to* ***wrong******price******format****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******86*** *shows the* ***itemlist.txt*** *with the* ***item*** *of* ***ID****, “****05****”* ***changed*** *to* ***VNFigurine****.*

#### **2.4.1.6 Delete Item**

Graphical user interface, text, website

Description automatically generated

***Figure******87*** *shows the* ***Delete******Item******panel****.*

**Figure** **87** shows the **Delete** **Item** **panel** which can be **accessed** by **clicking** on the **trash** **icon** as shown in **Figure** **87**. The **purpose** of the **Delete** **Item** **panel** is for **Admin** to **delete** **item** by **firstly** **inputting** the **correct** **inputs** for the **Item** **ID** and **Item** **Name**, and then **clicking** on the “**Delete**” **button** to **validate** the **inputs**. If the inputs provided is **true**, the **Item’s** **information** will be **deleted** from the **itemlist.txt** **file** and a **successful** **delete** **item** **message** will be **displayed** as shown in **Figure** **88**. Whereas if the input is **false**, an **error** **message** will be **displayed** as shown in **Figure** **89** and for the **other** **condition** which is **empty** **field**, an **error** **message** will also be **displayed** as shown in **Figure** **90**. As can be seen in **Figure** **91** is the **updated** **itemlist.txt** **file** after the **item** “***JGFigurine***” is **deleted**.

Graphical user interface, application

Description automatically generated

***Figure******88*** *shows the* ***Successful******Delete******Item******message****.*

*Graphical user interface, website

Description automatically generated*

***Figure******89*** *shows* ***Error******Message******displayed*** *due to* ***inputs******being******false****.*

*Graphical user interface, application, website

Description automatically generated*

***Figure******90*** *shows* ***Error******Message******displayed*** *due to* ***empty******fields****.*

*Graphical user interface, text, application

Description automatically generated*

***Figure******91*** *shows* ***itemlist.txt*** *with the* ***item******inputted*** *for* ***Deletion*** *is* ***deleted****.*

#### **2.4.1.7 View Past Transaction**

Graphical user interface

Description automatically generated

***Figure******92*** *shows the* ***View******Past******Transaction******panel****.*

As shown in **Figure** **92** is the **View** **Transaction** **panel** which **Admin** can **view** **past** **transaction** by firstly **inputting** the **Transaction** **ID** in the **text** **field** **provided** and **clicking** on the “***Search***” **button**. Once that is **accomplished**, the **information** in which the **transaction** **ID** is **provided** will be **displayed** at the **labels** as can be seen in **Figure** **94** if the **input** is **true** with a **successful** **fetched** **Transaction** **ID** **message** **displayed** **beforehand** as shown in **Figure** **93**. Though if the **input** **provided** is **false**, an **error** **message** will be **displayed** as shown in **Figure** **95**. In addition to that, an **error** **message** will also be **displayed** if there is **empty** **fields** **present** as shown in **Figure** **96**.

Graphical user interface

Description automatically generated

***Figure******93*** *shows the* ***successful******fetched******transaction******ID******message****.*

*Graphical user interface, website

Description automatically generated*

***Figure******94*** *shows* ***Transaction******ID****,* ***0000001’s******Transaction******Details****.*

*Graphical user interface, website

Description automatically generated*

***Figure******95*** *shows* ***Error******Message******displayed*** *due to* ***wrong******Transaction******ID******provided****.*

*Graphical user interface, website

Description automatically generated*

***Figure******96*** *shows the* ***Error******Message******displayed*** *due to* ***empty******fields****.*

### **2.4.2 Customer**

#### **2.4.2.1 Get Item Suggestion**

Graphical user interface

Description automatically generated with low confidence

***Figure*** ***97*** *shows the* ***User*** ***Main*** ***Menu***.

As shown in **Figure** **97** is the **User** **Main** **Menu** where the **User** can also get **suggestion** on **items** to **purchase** by **clicking** on the “***WHAT SHOULD I BUY?***” **button** which will then **select** **one** of the **6** **item** **IDs** **available** and **display** it to the **message** on the **right** which **says**, “***I Should Buy ItemID = ’??’***”**label** as shown in **Figure** **97.** **Figure** **98** shows an **Item** **ID, “*05*”** which is **generated** after **clicking** on the **button**.

Graphical user interface

Description automatically generated

***Figure******98*** *shows the* ***User******Main******Menu*** *with an* ***item******ID******generated****.*

## **2.5 Assumptions**

### **2.5.1 System-Oriented**

1. The system only caters to a single user type login which is the “Admin or Staff”.
2. The system management is decentralized as any Admin can assume administrative privilege of adding, updating, deleting Admin and Item related information, and viewing Past Transactions.
3. The system is lacking a centralized administration system for admin accounts management due to absence of a role higher than Admin.
4. The system is displayed in a fixed-size view in all of its forms in which resizing it is impossible.
5. The system allows for the logging off of admin via the close button and the logout button provided.
6. The system has no countermeasure or steps to prevent accidental application termination which may possibly lead to loss of information from the text files.
7. The system has no backup measures in place for the text files.

### **2.5.2 Business Model**

1. The vending machine only allows one type of item without any restriction on its quantity to be purchased per transaction.
2. An item purchased is not refundable upon finalizing transaction.
3. An item is only dispensed once payment has been made and transaction is finalized.
4. An item which quantity is out of stock is not purchasable by the customer.
5. Payment must be paid in full for item to be dispensed.
6. Multiple quantity of the same item is allowed per transaction.

### **2.5.3 Miscellaneous**

1. An admin may delete an item or admin account at any time, but the consequence of such action is irreversible without explicit access to the text files.
2. An admin must be aware of the system’s limitation when dealing with management of items and other admin accounts.
3. A deleted item’s data or empty slots’ information will not be displayed in any of the main menus and the picture shall be replace with “comingsoon.png”.
4. A deleted admin’s data will prevent the admin from being able to login into the Admin Main Menu.

# **3.0 Reference**

1. anubhava, 2015. *StackOverflow.* [Online]   
   Available at: https://stackoverflow.com/questions/33208761/java-regex-validate-username-length  
   [Accessed 2 April 2021].
2. arshajii, 2013. *StackOverflow.* [Online]   
   Available at: https://stackoverflow.com/questions/18259644/how-to-check-if-a-string-matches-a-specific-format  
   [Accessed 1 April 2021].
3. Developer, C., 2020. *YouTube.* [Online]   
   Available at: https://www.youtube.com/watch?v=wB9\_NVpMgY8  
   [Accessed 1 April 2021].
4. geeksforgeeks, 2018. *geeksforgeeks.* [Online]   
   Available at: https://www.geeksforgeeks.org/java-program-delete-certain-text-file/  
   [Accessed 8 April 2021].
5. geeksforgeeks, 2021. *geeksforgeeks.* [Online]   
   Available at: https://www.geeksforgeeks.org/access-modifiers-java/  
   [Accessed 11 April 2021].
6. Gibbs, M., 2018. *Study.* [Online]   
   Available at: https://study.com/academy/lesson/what-is-a-class-in-java-definition-examples.html  
   [Accessed 9 April 2021].
7. greg-449, 2014. *StackOverflow.* [Online]   
   Available at: https://stackoverflow.com/questions/24473427/clear-textfield-using-button-java  
   [Accessed 1 April 2021].
8. howtodoinjava, n.d.. *howtodinjava.* [Online]   
   Available at: https://howtodoinjava.com/java/string/convert-int-to-string/  
   [Accessed 6 April 2021].
9. javaExamples, 2019. *javaExamples.* [Online]   
   Available at: https://www.java-examples.com/disable-button-example  
   [Accessed 5 April 2021].
10. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/java-string-to-int  
    [Accessed 6 April 2021].
11. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/java-jfilechooser  
    [Accessed 6 April 2021].
12. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/java-bufferedreader-class  
    [Accessed 7 April 2021].
13. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/how-to-create-object-in-java  
    [Accessed 10 April 2021].
14. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/access-modifiers  
    [Accessed 11 April 2021].
15. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/exception-handling-in-java  
    [Accessed 12 April 2021].
16. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/event-handling-in-java  
    [Accessed 12 April 2021].
17. javatpoint, n.d.. *javatpoint.* [Online]   
    Available at: https://www.javatpoint.com/event-handling-in-java  
    [Accessed 12 April 2021].
18. Jens, 2015. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/32943897/java-ioexception-stream-closed  
    [Accessed 3 April 2021].
19. kevmo314, 2009. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/731365/reading-and-displaying-data-from-a-txt-file  
    [Accessed 2 April 2021].
20. Lacquement, K., 2011. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/5507168/java-lang-numberformatexception-for-input-string  
    [Accessed 5 April 2021].
21. Lord, J. i., 2016. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/36492044/how-to-create-regex-for-00-00  
    [Accessed 3 April 2021].
22. Oracle, n.d.. *Oracle.* [Online]   
    Available at: https://docs.oracle.com/javase/7/docs/api/javax/swing/filechooser/FileNameExtensionFilter.html  
    [Accessed 6 April 2021].
23. question, a., 2013. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/19207082/refresh-same-image-in-a-jlabel  
    [Accessed 9 April 2021].
24. Ritchie, C., 2007. *Code Ranch.* [Online]   
    Available at: https://coderanch.com/t/344481/java/clear-JPasswordField  
    [Accessed 1 April 2021].
25. Skills, I. Y. P., 2016. *YouTube.* [Online]   
    Available at: https://www.youtube.com/watch?v=n5HFF1JqnzU  
    [Accessed 3 April 2021].
26. tutorialspoint, n.d.. *tutorialspoint.* [Online]   
    Available at: https://www.tutorialspoint.com/java/java\_filewriter\_class.htm  
    [Accessed 8 April 2021].
27. tutorialspoint, n.d.. *tutorialspoint.* [Online]   
    Available at: https://www.tutorialspoint.com/java-program-to-delete-certain-text-from-a-file  
    [Accessed 8 April 2021].
28. tutorialspoint, n.d.. *tutorialspoint.* [Online]   
    Available at: https://www.tutorialspoint.com/java/java\_constructors.htm  
    [Accessed 10 April 2021].
29. tutorialspoint, n.d.. *tutorialspoint.* [Online]   
    Available at: https://www.tutorialspoint.com/java/java\_packages.htm  
    [Accessed 12 April 2021].
30. user225269, 2010. *StackOverflow.* [Online]   
    Available at: https://stackoverflow.com/questions/2937513/new-line-in-java  
    [Accessed 1 April 2021].
31. w3schools, n.d.. *w3schools.* [Online]   
    Available at: https://www.w3schools.com/java/java\_try\_catch.asp  
    [Accessed 8 April 2021].
32. w3schools, n.d.. *w3schools.* [Online]   
    Available at: https://www.w3schools.com/java/java\_classes.asp  
    [Accessed 9 April 2021].
33. w3schools, n.d.. *w3schools.* [Online]   
    Available at: https://www.w3schools.com/java/java\_constructors.asp  
    [Accessed 11 April 2021].
34. w3schools, n.d.. *w3schools.* [Online]   
    Available at: https://www.w3schools.com/java/java\_packages.asp  
    [Accessed 12 April 2021].
35. w3schools, n.d.. *w3schools.* [Online]   
    Available at: https://www.w3schools.com/java/java\_try\_catch.asp  
    [Accessed 12 April 2021].