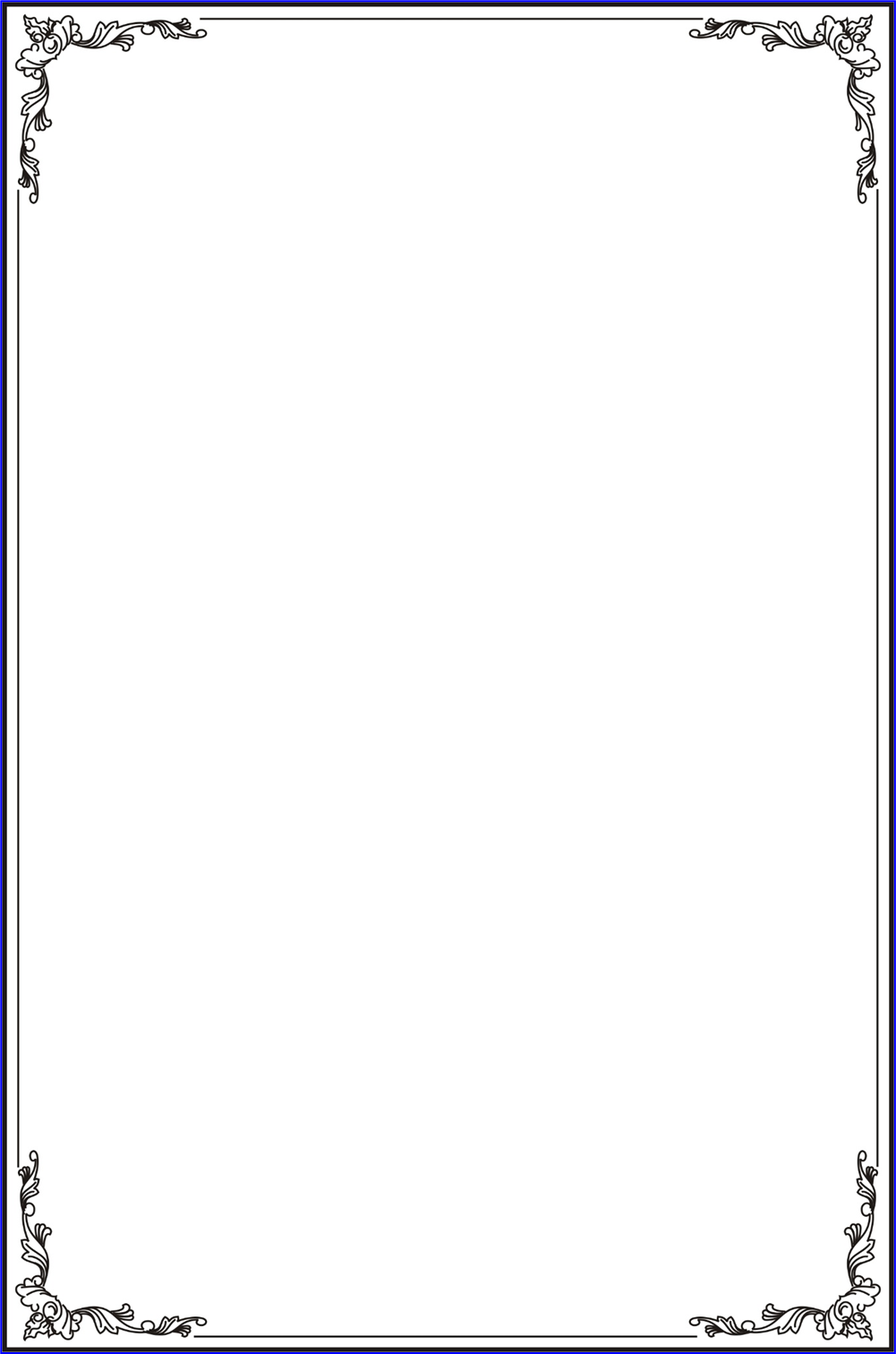
**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**

**🙠🙟🕮🙝🙢**



**SOFTWARE TESTING**

**Building a website for selling smartphones**

**Advisor: Nguyễn Trần Thi Văn, MSc**

**Group: 01**

**Student Name: Student ID**

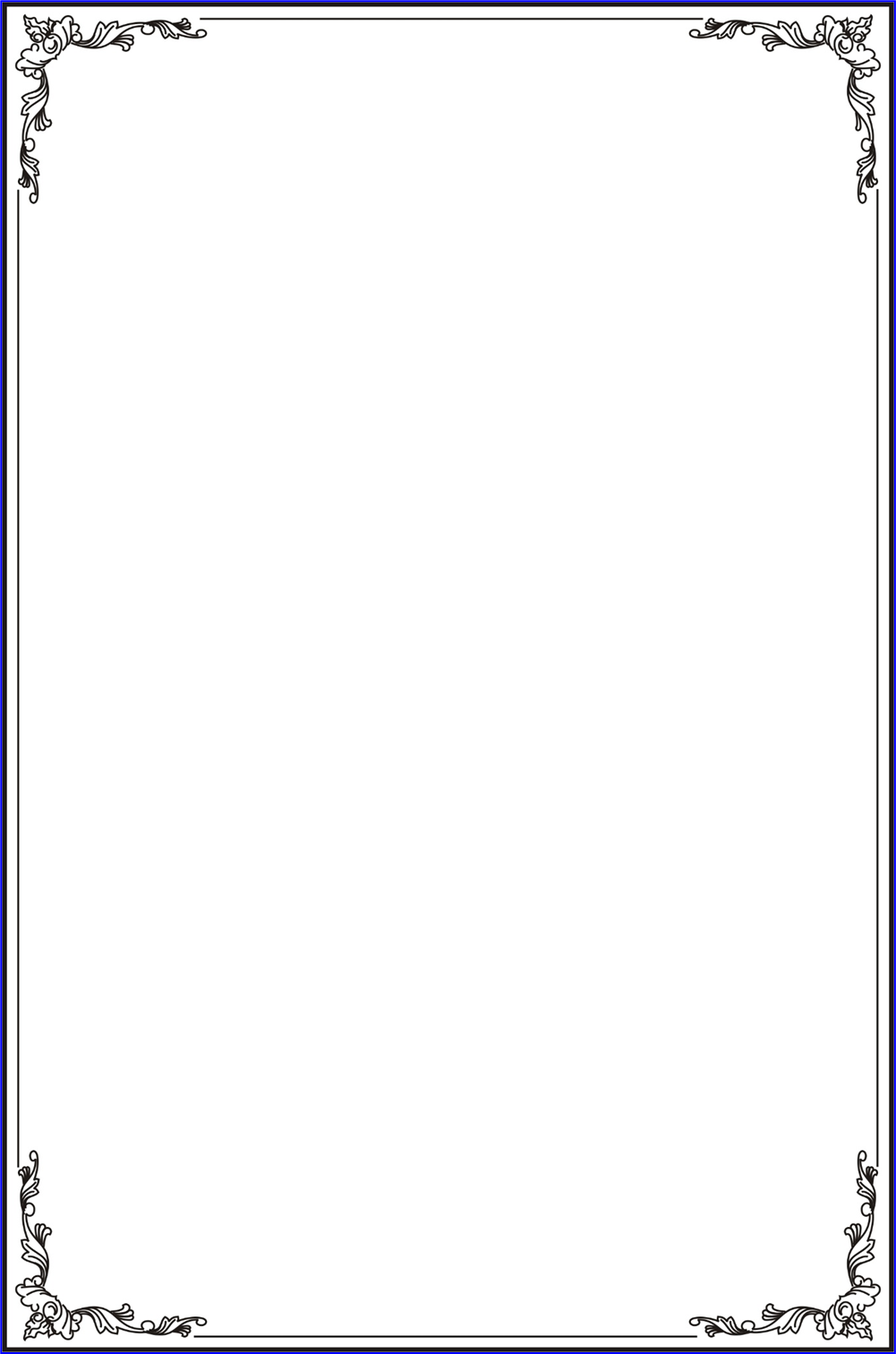
**Võ Minh Hưng 20110426**

**Trương Chí Kiên 20110376**

**Nguyễn Hoàng Nhân 20110010**

**Lê Hoàng Lâm 20110030**

**Ho Chi Minh, December 2023**

**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**

**🙠🙟🕮🙝🙢**



**SOFTWARE TESTING**

**Building a website for selling smartphones**

**Advisor: Nguyễn Trần Thi Văn, MSc**

**Group: 01**

**Student Name: Student ID**

**Võ Minh Hưng 20110426**

**Trương Chí Kiên 20110376**

**Nguyễn Hoàng Nhân 20110010**

**Lê Hoàng Lâm 20110030**

**Ho Chi Minh, December 2023**

**INSTRUCTOR’S COMMENTS**

**ACKNOWLEDGEGEMENT**

The success of a student in the field of software testing is greatly influenced by the guidance and support of the instructor. We would like to express our sincere gratitude to Mr. Nguyen Tran Thi Van, who provided direct assistance to our team throughout the software testing course. His valuable suggestions, comments, and advice played a crucial role in helping us excel in the project.

Although the project duration was approximately three months, some unexpected challenges emerged, resulting in certain errors and shortcomings. However, we are confident that these issues serve as learning opportunities and will further enhance our understanding of software testing principles. Throughout the course, we were able to create a stimulating learning environment and acquire the necessary skills to develop a real-world software product. This experience not only enhanced our individual capabilities but also instilled a strong work ethic within our team.

We sincerely appreciate your support and encouragement, and we eagerly anticipate receiving your feedback to further refine and improve our software testing skills. Once again, we extend our heartfelt gratitude to Mr. Nguyen Tran Thi Van for his invaluable guidance.

Best regards, The Software Testing Course Team.

*Ho Chi Minh City, Dec 2023*

*Students*

*(Student group implements)*

**TABLE OF CONTENTS**

[CHAPTER 1 SYSTEM SPECIFICATIONS 1](#_Toc153613218)

[1.1 SYSTEM SPECIFICATIONS 1](#_Toc153613219)

[1.2 SYSTEM REQUIREMENTS 1](#_Toc153613220)

[1.2.1 Functional requirements 1](#_Toc153613221)

[1.2.2 User requirements 2](#_Toc153613222)

[1.2.3 Business requirements 3](#_Toc153613223)

[1.2.4 Quality requirements 3](#_Toc153613224)

[CHAPTER 2 CURRENT STATUS SURVEY AND REQUIREMENT MODELING 5](#_Toc153613225)

[2.1. CURRENT STATUS SURVEY 5](#_Toc153613226)

[2.2.1 CellphoneS 5](#_Toc153613227)

[2.2.2 Hoàng Hà mobile 6](#_Toc153613228)

[2.2.3 Thế giới di động 7](#_Toc153613229)

[2.2. MODELING REQUIREMENTS 8](#_Toc153613230)

[2.2.1 USE CASE DIAGRAM 8](#_Toc153613231)

[2.2.2 CONCEPTUAL DIAGRAM 10](#_Toc153613232)

[2.2.3 USE CASE SCENARIO 10](#_Toc153613233)

[2.2.3.1. Use Case “Login” 10](#_Toc153613234)

[2.2.3.2. Use Case “Register Account” 11](#_Toc153613235)

[2.2.3.3. Use Case “Logout” 12](#_Toc153613236)

[2.2.3.4. Use Case “Forgot Password” 13](#_Toc153613237)

[2.2.3.5. Use Case “View Product” 14](#_Toc153613238)

[2.2.3.6. Use Case “Find Product” 15](#_Toc153613239)

[2.2.3.7. Use Case “Filter Product” 16](#_Toc153613240)

[2.2.3.8. Use Case “Manage Cart” 17](#_Toc153613241)

[2.2.3.9. Use Case “Add Product To Cart” 18](#_Toc153613242)

[2.2.3.10. Use Case “Update Cart Quantity” 19](#_Toc153613243)

[2.2.3.11. Use Case “Delete Cart Products” 20](#_Toc153613244)

[2.2.3.12. Use Case “Buy Product” 21](#_Toc153613245)

[2.2.3.13. Use Case “Manage Account” 22](#_Toc153613246)

[2.2.3.14. Use Case “Manage Personal account” 23](#_Toc153613247)

[2.2.3.15. Use Case “Manage Product” 24](#_Toc153613248)

[2.2.3.16. Use Case “Manage Order” 27](#_Toc153613249)

[2.2.3.17. Use Case “View Order History” 28](#_Toc153613250)

[2.2.3.18. Use Case “View Order Details” 29](#_Toc153613251)

[CHAPTER 3 TEST PLAN 31](#_Toc153613252)

[3.1. SCOPE 31](#_Toc153613254)

[3.1.1. In Scope 31](#_Toc153613255)

[3.1.2. Out of Scope 31](#_Toc153613256)

[3.2. QUALITY OBJECTIVE 31](#_Toc153613257)

[3.2.1. Primary Objective 31](#_Toc153613258)

[3.2.2. Secondary Objective 31](#_Toc153613259)

[3.3. ROLES AND RESPONSIBILITIES 31](#_Toc153613260)

[3.4. ASSUMPTIONS FOR TEST EXECUTION 32](#_Toc153613261)

[3.5 TESTING METHODS 32](#_Toc153613262)

[3.5.1. Overview 32](#_Toc153613263)

[3.5.2. Boundary Value Analysis 33](#_Toc153613264)

[3.5.3. Decision Table 33](#_Toc153613265)

[3.5.4. State Transition 34](#_Toc153613266)

[3.5.5. Use Case 35](#_Toc153613267)

[3.6. TESTING COMPLETENESS 35](#_Toc153613268)

[3.7. TEST LEVELS 35](#_Toc153613269)

[3.7.1. Build Tests 35](#_Toc153613270)

[3.7.2. Milestone Tests 36](#_Toc153613271)

[3.7.3. Release Tests 36](#_Toc153613272)

[3.8. TESTING RISKS AND MITIGATION FACTORS 37](#_Toc153613273)

[CHAPTER 4 TEST PLAN AND PROCESS 40](#_Toc153613274)

[4.1. TESTING PROCESS 40](#_Toc153613279)

[4.1.1. Building a test plan 40](#_Toc153613280)

[4.1.2. Test Analysis and Design 40](#_Toc153613281)

[4.1.3. Set up the necessary environment as required to run the program 40](#_Toc153613282)

[4.1.4. Evaluate test results and report findings 41](#_Toc153613283)

[4.1.5. Closing the Testing Activities 41](#_Toc153613284)

[4.2. TEST PLAN 41](#_Toc153613285)

[4.2.1. Introduction 41](#_Toc153613286)

[4.2.1.1. Purpose 41](#_Toc153613287)

[4.2.1.2. Overview 41](#_Toc153613288)

[4.2.1.3. Scope 42](#_Toc153613289)

[4.2.1.4. Document Users 42](#_Toc153613290)

[4.2.1.5. References 42](#_Toc153613291)

[4.2.2. Work schedule 43](#_Toc153613292)

[4.2.3. Resource requirements 43](#_Toc153613293)

[4.2.3.1. Hardware 43](#_Toc153613294)

[4.2.3.2. Software 43](#_Toc153613295)

[4.2.3.3. Testing tools 43](#_Toc153613296)

[4.2.3.4. Testing Environment 43](#_Toc153613297)

[4.2.3.5. Test role 44](#_Toc153613298)

[4.2.4 Testing strategy 44](#_Toc153613299)

[4.2.5. Acceptance criteria 45](#_Toc153613300)

[CHAPTER 5 ACTUAL TEST 46](#_Toc153613301)

[5.1 WHITEBOX TESTING 46](#_Toc153613302)

[5.1.1 Create Product Function 46](#_Toc153613303)

[5.1.2 Update Product Function 51](#_Toc153613304)

[5.1.3 Get Details Product Function 55](#_Toc153613305)

[5.1.4 Delete Product Function 58](#_Toc153613306)

[5.1.5 Delete Many Product Function 61](#_Toc153613307)

[5.1.6 Get All Type Product Function 63](#_Toc153613308)

[5.1.7 Create User Function 66](#_Toc153613309)

[5.1.8 Delete User Function 71](#_Toc153613310)

[CHAPTER 6 BLACKBOX TESTING 75](#_Toc153613311)

[6.1 BUG MANAGEMENT SYSTEM 75](#_Toc153613312)

[6.1.1 Introduction of MantisBT 75](#_Toc153613313)

[6.1.2 Advantages of MantisBT 75](#_Toc153613314)

[6.1.3 Setting up and configuration 76](#_Toc153613315)

[6.2 BUG REPORTS 77](#_Toc153613316)

[6.2.1 Bug report [LAM\_002] 77](#_Toc153613317)

[6.2.2 Bug report [HNG\_001] 79](#_Toc153613318)

[6.2.3 Bug report [HNG\_002] 80](#_Toc153613319)

[6.2.4 Bug report [KIN\_001] 81](#_Toc153613320)

[6.2.5 Bug report [KIN\_002] 82](#_Toc153613321)

[6.2.6 Bug report [KIN\_003] 83](#_Toc153613322)

[6.2.7 Bug report [NHN\_002] 84](#_Toc153613323)

[6.2.8 Bug report [LAM\_004] 85](#_Toc153613324)

[6.2.9 Bug report [LAM\_006] 86](#_Toc153613325)

[6.2.10 Bug report [LAM\_008] 87](#_Toc153613326)

[6.2.11 Bug report [LAM\_011] 88](#_Toc153613327)

[6.2.12 Bug report [LAM\_014] 89](#_Toc153613328)

[6.2.13 Bug report [NHN\_009] 90](#_Toc153613329)

[CHAPTER 7 CONCLUSION 91](#_Toc153613330)

[7.1 ARCHIEVEMENT 91](#_Toc153613331)

[7.2 FAVORABLE 91](#_Toc153613332)

[7.3 SHORTCOMING 91](#_Toc153613333)

[7.4 GENERAL KNOWLEDGE 91](#_Toc153613334)

[7.5 FUTURE DEVELOPMENT 91](#_Toc153613335)

[REFERENCES 92](#_Toc153613336)

[TASK DISTRIBUTION 93](#_Toc153613337)

**LIST OF FIGURES**

[Figure 1 Home Page “CellphoneS” 5](#_Toc153613338)

[Figure 2 Home Page “hoanghamobile.com” 6](#_Toc153613339)

[Figure 3 Home Page “thegioididong” 7](#_Toc153613340)

[Figure 4 Use Case “User” Diagram 8](#_Toc153613341)

[Figure 5 Use Case Diagram “Admin” 9](#_Toc153613342)

[Figure 6 Testing strategy 44](#_Toc153613343)

[Figure 7 Create Product Data Flow Graph 47](#_Toc153613344)

[Figure 8 Create Product Data Flow Graph 49](#_Toc153613345)

[Figure 9 Update Product Control Flow Graph 52](#_Toc153613346)

[Figure 10 Update Product Data Flow Graph 53](#_Toc153613347)

[Figure 11 Get Details Product Control Flow Graph 56](#_Toc153613348)

[Figure 12 Get Details Product Data Flow Graph 57](#_Toc153613349)

[Figure 13 Delete Product Control Flow Graph 59](#_Toc153613350)

[Figure 14 Delete Product Data Flow Graph 60](#_Toc153613351)

[Figure 15 Delete Many Product Control Flow Graph 61](#_Toc153613352)

[Figure 16 Delete Many Product Data Flow Graph 62](#_Toc153613353)

[Figure 17 Get All Types Control Flow Graph 64](#_Toc153613354)

[Figure 18 Get All Type Data Flow Graph 65](#_Toc153613355)

[Figure 19 Create User Control Flow Graph 67](#_Toc153613356)

[Figure 20 Create User Data Flow Graph 68](#_Toc153613357)

[Figure 21 Delete User Control Flow Graph 72](#_Toc153613358)

[Figure 22 Delete User Data Flow Graph 73](#_Toc153613359)

[Figure 23 Bug report [LAM\_002] 77](#_Toc153613360)

[Figure 24 Activities in [LAM\_002] 77](#_Toc153613361)

[Figure 25 Bug report [HNG\_001] 79](#_Toc153613362)

[Figure 26 Activities [HNG\_001] 79](#_Toc153613363)

[Figure 27 Bug report [HNG\_002] 80](#_Toc153613364)

[Figure 28 Activities in [HNG\_002] 80](#_Toc153613365)

[Figure 29 Bug report [KIN\_001] 81](#_Toc153613366)

[Figure 30 Activities in [KIN\_001] 81](#_Toc153613367)

[Figure 31 Bug report [KIN\_002] 82](#_Toc153613368)

[Figure 32 Activities in [KIN\_002] 82](#_Toc153613369)

[Figure 33 Bug report [KIN\_003] 83](#_Toc153613370)

[Figure 34 Activities in [KIN\_003] 83](#_Toc153613371)

[Figure 35 Bug report [NHN\_002] 84](#_Toc153613372)

[Figure 36 Activities in [NHN\_002] 84](#_Toc153613373)

[Figure 37 Bug report [LAM\_004] 85](#_Toc153613374)

[Figure 38 Activities in [LAM\_004] 85](#_Toc153613375)

[Figure 39 Bug report [LAM\_006] 86](#_Toc153613376)

[Figure 40 Activities in [LAM\_006] 86](#_Toc153613377)

[Figure 41 Bug report [LAM\_008] 87](#_Toc153613378)

[Figure 42 Activities in [LAM\_008] 87](#_Toc153613379)

[Figure 43 Bug report [LAM\_011] 88](#_Toc153613380)

[Figure 44 Activities in [LAM\_011] 88](#_Toc153613381)

[Figure 45 Bug report [LAM\_014] 89](#_Toc153613382)

[Figure 46 Activities in [LAM\_014] 89](#_Toc153613383)

[Figure 47 Bug report [NHN\_009] 90](#_Toc153613384)

[Figure 48 Activities in [NHN\_009] 90](#_Toc153613385)

**LIST OF TABLES**

[Table 1 Functional requirements 2](#_Toc153613386)

[Table 2 User requirements 3](#_Toc153613387)

[Table 3 Use Case “Login” 10](#_Toc153613388)

[Table 4 Use Case "Register Account" 11](#_Toc153613389)

[Table 5 Use Case “Logout” 12](#_Toc153613390)

[Table 6 Use Case "Forgot Password" 13](#_Toc153613391)

[Table 7 Use Case “View Product” 14](#_Toc153613392)

[Table 8 Use Case “Find Product” 15](#_Toc153613393)

[Table 9 Use Case “Filter Product” 16](#_Toc153613394)

[Table 10 Use Case “Manage Cart” 17](#_Toc153613395)

[Table 11 Use Case “Add Product To Cart” 18](#_Toc153613396)

[Table 12 Use Case “Update Cart Quantity” 19](#_Toc153613397)

[Table 13 Use Case “Delete Cart Products” 20](#_Toc153613398)

[Table 14 Use Case “Buy Product” 21](#_Toc153613399)

[Table 15 Use Case “Manage Account” 22](#_Toc153613400)

[Table 16 Use Case “Manage Personal account” 23](#_Toc153613401)

[Table 17 Use Case “Manage Product” 24](#_Toc153613402)

[Table 18 Use Case “Manage Order” 27](#_Toc153613403)

[Table 19 Use Case “View Order History” 28](#_Toc153613404)

[Table 20 Use Case “View Order Details” 29](#_Toc153613405)

[Table 21 Work Schedule 43](#_Toc153613406)

[Table 22 Test Role 44](#_Toc153613407)

[Table 23 Functional testing 45](#_Toc153613408)

[Table 24 Create Product Test Cases 48](#_Toc153613409)

[Table 25 Create Product Variable Lifecycle 50](#_Toc153613410)

[Table 26 Update Product Test Cases 53](#_Toc153613411)

[Table 27 Update Product Variables Lifecycle 54](#_Toc153613412)

[Table 28 Get Details Product Test Cases 56](#_Toc153613413)

[Table 29 Get Details Product Variable Lifecycle 58](#_Toc153613414)

[Table 30 Delete Product Test Cases 59](#_Toc153613415)

[Table 31 Delete Product Variables Lifecycle 60](#_Toc153613416)

[Table 32 Delete Many Product Test Cases 62](#_Toc153613417)

[Table 33 Delete Many Product Variables Lifecycle 63](#_Toc153613418)

[Table 34 Get All Type Test Cases 64](#_Toc153613419)

[Table 35 Get All Type Variables Lifecycle 65](#_Toc153613420)

[Table 36 Create User Test Cases 67](#_Toc153613421)

[Table 37 Create User Variables Lifecycle 69](#_Toc153613422)

[Table 38 Delete User Test Cases 72](#_Toc153613423)

[Table 39 Delete User Variables Lifecycle 73](#_Toc153613424)

# CHAPTER 1 SYSTEM SPECIFICATIONS

## 1.1 SYSTEM SPECIFICATIONS

As the society is increasingly developed, people's living standards are improved, economic income is higher and higher, the mobile phone is no longer far away from everyone, it has become a An indispensable item for every citizen today. Most people equip themselves with a phone that suits their needs and pocket. However, with today's increasingly busy life, to want to buy a favorite phone, consumers have to go to the store to choose, so it will take a lot of time and effort.

Along with the above reasons, through my research and understanding, I have built a website system to sell mobile phones online to make buying and selling phones easier, helping customers reduce costs. Time and effort do not have to go to the store to buy goods. If you want to choose a suitable phone for yourself, customers just need to sit next to a computer with an internet connection to be able to buy their favorite items.

The store system that sells mobile phones includes many other complex and difficult jobs. Therefore this topic building a sales website only focus on sales and stages such as management: human resources, order processing.

## 1.2 SYSTEM REQUIREMENTS

### 1.2.1 Functional requirements

* User management.
* Product management.
* Income management.
* User personal information modification.
* Providing solutions for user’s forgetting password.
* Searching and filtering for products.
* Recommending hottest products.
* View and purchase products.
* Shopping cart and check out features.

Table 1 Functional requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Requirement** | **Type of requirement** | **Note** |
| 1 | Product management | Storage | Product information includes product name, price, brand,…  Information at the store includes Phone information, brand, configuration,… |
| 2 | Manage customer information | Storage | Personal information such as full name, location, contact address, email, account, phone number,... |
| 3 | Manage cart | Storage | Selected product, product quantity, .. |
| 4 | Store information, transaction history | Storage | Personal information such as full name, location, contact address, email, account, phone number,... |
| 5 | Find products | Search | By name, by type |
| 6 | Search for customer’s information | Search | Manage by admin |
| 7 | Calculate revenue | Calculation | Total revenue by day, month, year. |
| 8 | Sales Statistics | Report | Day, week, month. |

### 1.2.2 User requirements

* Friendly UI/UX.
* Simple, easy-to-interact, easy-to-access website.

Table 2 User requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Content** | **Criteria** | **Description** | **Note** |
| Software can be updated and modified to suit the requirements of the user as well as the management requirements of the shop. | Resilience. |  |  |
| Graphical user interface:  - User-friendly.  - Easy operation.  - The function buttons are simply  - Functional windows can interact with each other, bringing efficiency to the user. | Convenience |  |  |
| - Database is secure and easy to access.  - The system works stably, reliably and responds immediately. | Convenience |  |  |
| - Deliver effective user experience | Compatibility |  |  |
| - The software can fulfill user requirements without doing a silo effect. | Compatibility |  |  |

### 1.2.3 Business requirements

Product ordered must be followed by below steps:

* Sign in
* Product availability
* Order product
* Confirm order
* Choose payment method
* Delivery product ordered

### 1.2.4 Quality requirements

* Meet the needs of customers to use service and features of the website.
* Providing effective features and satisfaction to customers.
* Brings an easy experience in making phone purchases.

# CHAPTER 2 CURRENT STATUS SURVEY AND REQUIREMENT MODELING

## 2.1. CURRENT STATUS SURVEY

- Below are some famous websites for selling phones in Vietnam and their main functions

### 2.2.1 CellphoneS

- Link: <https://cellphones.com.vn/>



Figure 1 Home Page “CellphoneS”

* **Main function of Cellphones:**
* View, search, filter products
* Manage the product in the cart
* Add the product to the Cart
* Checkout
* Track Order
* Chat box
* **Strengths:**

+ Products are classified in detail

+ Diverse payment methods (COD, PayPal, ,,,)

+ Develop related technology products (Điện thoại, Tablet, laptop, Âm thanh …)

+ View details information of product (configuration, manufacturer,…)

* **Weaknesses:**

+ Delivery time and shipping costs sometimes also become disadvantages of sales websites.

### 2.2.2 Hoàng Hà mobile

- Link: https://hoanghamobile.com

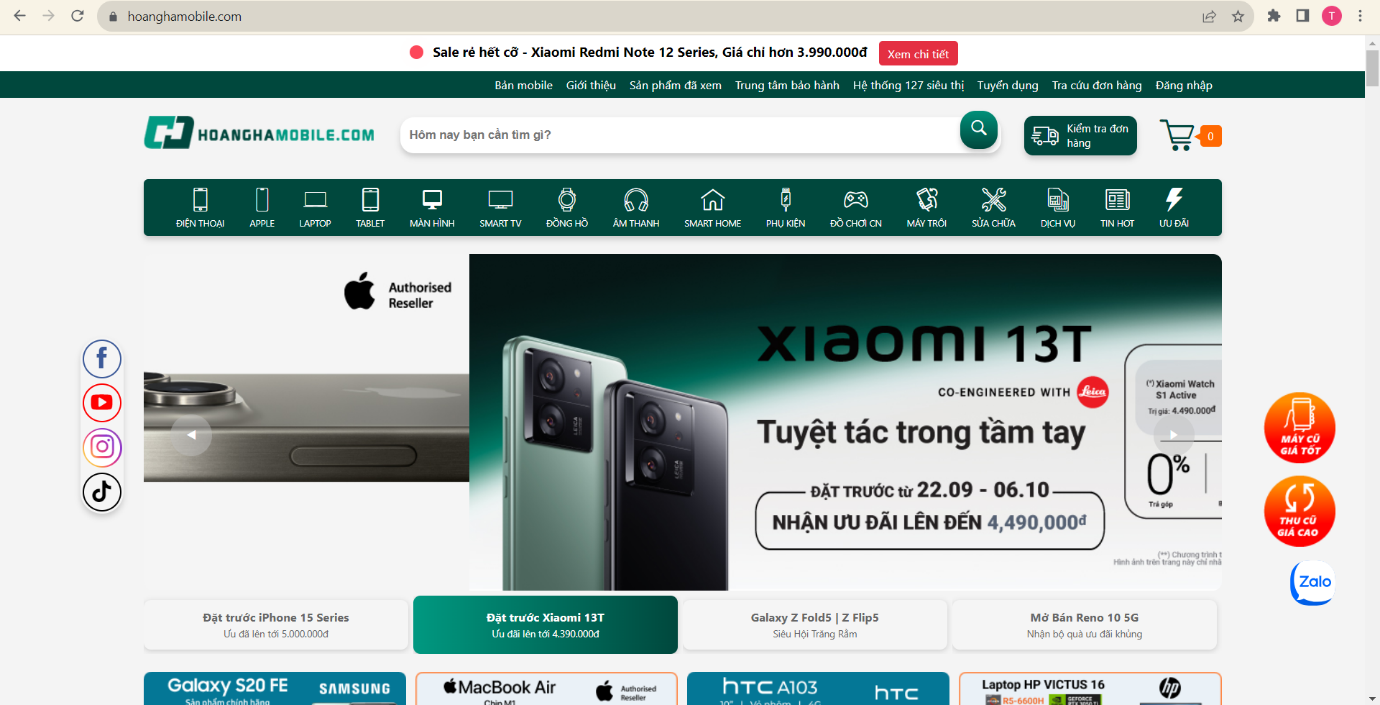


Figure 2 Home Page “hoanghamobile.com”

* **Main function of Hoàng Hà Mobile :**
* View, search, filter products
* Manage the product in the cart
* Add the product to the Cart
* Checkout
* Track Order
* Reserve phone ( reserve iphone 15, …)
* **Strengths:**

+ Products are classified in detail

+ Diverse payment methods (COD, PayPal, Visa,…)

+ View details information of product (configuration, manufacturer,…)

* **Weaknesses:**

+ Delivery time and shipping costs sometimes also become disadvantages of sales websites.

### 2.2.3 Thế giới di động

- Link: <https://www.thegioididong.com/>



Figure 3 Home Page “thegioididong”

* **Main function of TheGioiDiDong :**
* View, search, filter products
* Manage the product in the cart
* Add the product to the Cart
* Checkout

+ Reserve phone ( reserve Samsung Galasy Z flip 5, …)

+ Harvest and innovate phones

* **Strengths**:

+ Products are classified in detail

+ Diverse payment methods (COD, PayPal, Visa,…)

+ View details information of product (configuration, manufacturer,…)

* **Weaknesses:**

+ Delivery time and shipping costs sometimes also become disadvantages of sales websites.

## 2.2. MODELING REQUIREMENTS

### 2.2.1 USE CASE DIAGRAM

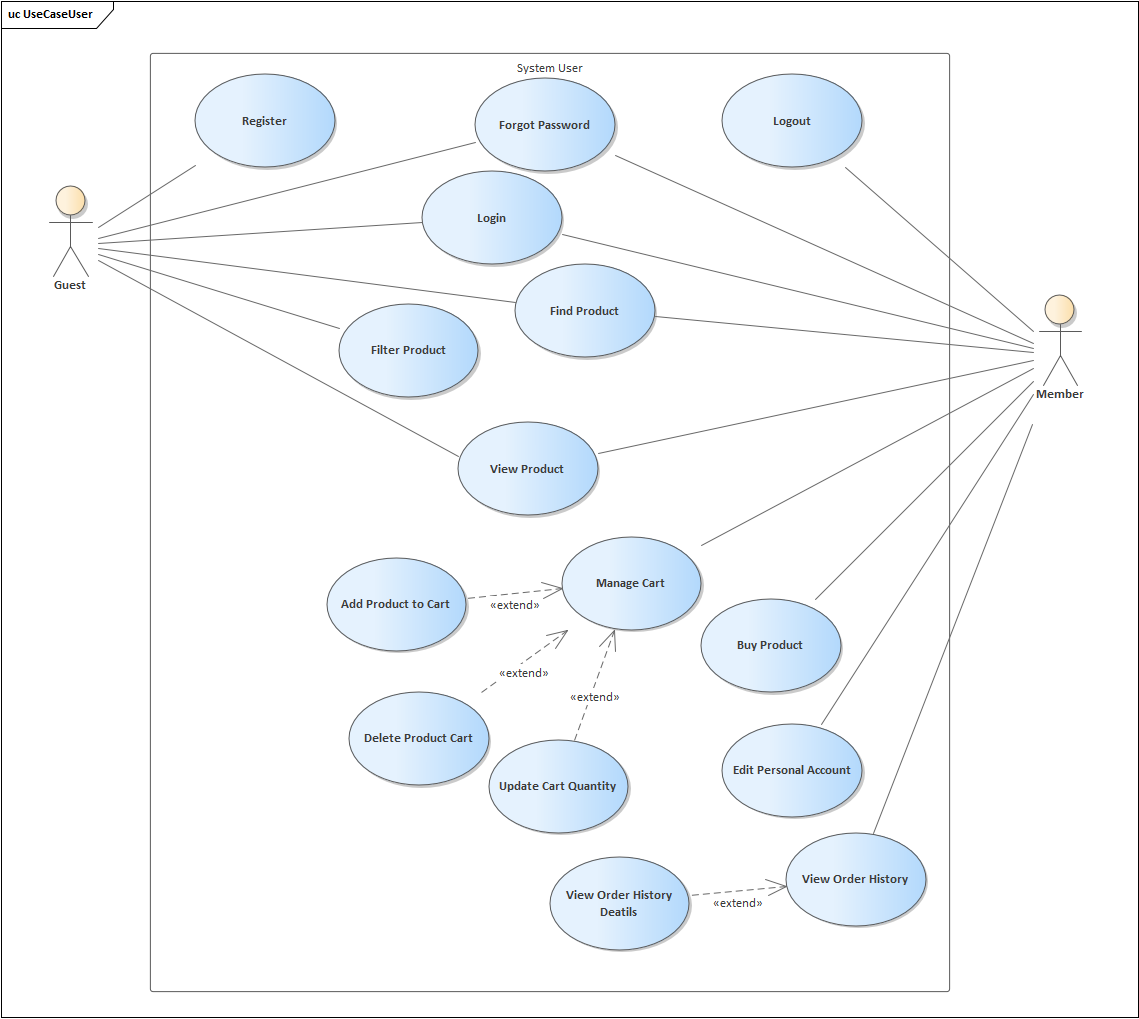


Figure 4 Use Case “User” Diagram

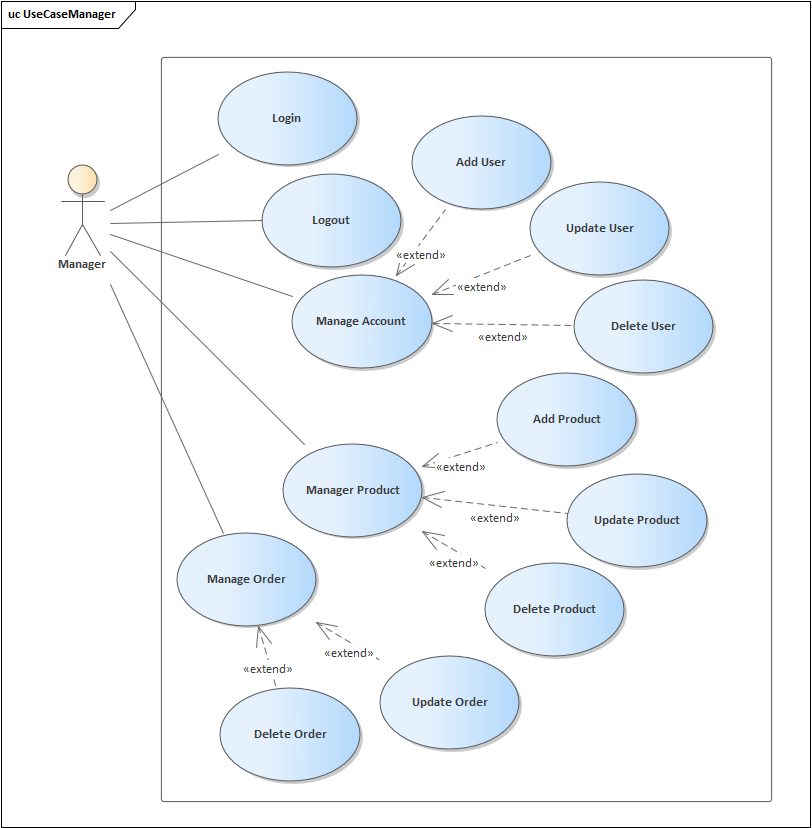


Figure 5 Use Case Diagram “Admin”

### 2.2.2 CONCEPTUAL DIAGRAM

### 2.2.3 USE CASE SCENARIO

#### 2.2.3.1. Use Case “Login”

Table 3 Use Case “Login”

|  |  |
| --- | --- |
| Use Case ID | UC\_LG |
| Use Case Name | Login |
| Description | This Use Case allows the Actor to login to the website |
| Actor | Member |
| Pre-  Condition(s) | The Actor on the login page |
| Post-  Condition(s) | Successful login as Member account |
| Basic Flow | 1. The actor clicks the “Đăng ký/ Đăng nhập” button. 2. The system displays the Sign In page. 3. Actors enter the required fields (email, password) to login [E1] 4. The Actor clicks the “Login” button. 5. The system verifies the registration information [E2] 6. The system redirects to the Home Page. |
| Alternative Flow |  |
| Exception Flow | [E1] If Actors do not enter email or password, the system display error message “The input is required”. If, the format of email is wrong, the system display error message “The input is not format of an email!” |

#### 2.2.3.2. Use Case “Register Account”

Table 4 Use Case "Register Account"

|  |  |
| --- | --- |
| Use Case ID | UC\_RA |
| Use Case | Register Account |
| Short Description | This Use Case allows the Actor to register a personal account to  log in to the website. |
| Actor | Guest |
| Pre-  Condition(s) | Actor on login page |
| Post-Condition(s) | Successful account registration |
| Basic Flow | 1.The actor clicks the “Đăng nhập/Đăng kí” button.  2. The system displays the Sign In page.  3. Click "Tạo tài khoản" under the "Đăng nhập" button.  4. The system displays the Sign Up page.  5. Actors enter the required fields (email, password , confirm password) to register [E1]  6. The Actor clicks the “Đăng ký” button.  7. The system verifies the registration information [E2]  8. The system redirects to the Login Page. |
| Alternative Flow |  |
| Exception Flow | [E1] If Actors do not enter email or password or confirm password, the system display error message “The input is required”. If, the format of email is wrong, the system display error message “The input is not format of an email!”  [E2] If the email already exists, the system displays an error message and requests the Actor to enter a different email. |

#### 2.2.3.3. Use Case “Logout”

Table 5 Use Case “Logout”

|  |  |
| --- | --- |
| Use Case ID | UC\_LO |
| Use Case Name | Logout |
| Description | Use case allows Actor to log out Actor's account from the website |
| Actor | Manager, Member |
| Pre-Condition(s) | Actor has logged into the system. |
| Post-  Condition(s) | Actor is logged out of the account on the website. |
| Basic Flow | 1. Actor presses the “Log Out” function. 2. The system deletes the user’s session 3. The system redirects the actor to the homepage. |
| Alternative Flow |  |
| Exception Flow |  |

#### 2.2.3.4. Use Case “Forgot Password”

Table 6 Use Case "Forgot Password"

|  |  |
| --- | --- |
| Use Case ID | UC\_FP |
| Use Case Name | Use Case Forget Password. |
| Description | This Use Case allows the User to create a new password. |
| Actor | Guest |
| Pre-  Condition(s) | The Actor account is already created. |
| Post-  Condition(s) | The Actor has registered an account. |
| Basic Flow | 1. The Actor accesses the login page.  2. Actor clicks "Forgot password".  3. The system displays a Forgot Password page requesting the Actor to enter the email address associated with the account.  4. The Actor enters the email address.  5. Actor clicks "Get New Password”.  6. The system checks the email address. [E1]  7. The system sends an email to recover the password.  8. The Actor accesses the page to reset the password  9. The actor enters the new password and confirms the new password and clicks "Reset."  10. The system verifies the validity of the new password. [E2]  11. The system saves the new password 12. The system redirects the actor to the login page |
| Alternative Flow |  |
| Exception Flow | [E1]  Actor enters an invalid email, the system will display an error message and request the Actor to enter a valid email.  [E2] Actor enters a mismatched confirmation password, the system will notify and request the Actor to re-enter the confirmation password. |

#### 2.2.3.5. Use Case “View Product”

Table 7 Use Case “View Product”

|  |  |
| --- | --- |
| Use Case ID | UC\_VP |
| Use Case Name | View product |
| Description | Allow the actor to view product details when the actor clicks on the product they want to view |
| Actor | Guest, Member |
| Pre-  Condition(s) | Choose one product |
| Post-Condition(s) | The 'Products screen' shows the details of that product such as description, price, quantity, rate,… |
| Basic Flow | 1. The actor clicks the product they want to view information.  2. The system displays the product detail page. |
| Alternate Flow |  |
| Exception Flow |  |

#### 2.2.3.6. Use Case “Find Product”

Table 8 Use Case “Find Product”

|  |  |
| --- | --- |
| Use Case ID | UC\_FP |
| Use Case Name | Find product |
| Description | The actor can find product based on specified filtering criteria |
| Actor | Guest, Member |
| Pre-Condition(s) |  |
| Post-Condition(s) | The system displays products that match the actor's specified  filtering criteria. |
| Basic Flow | 1. Actor enters the keyword or name of the product they want to find for in the search bar on the homepage or navigation bar.  2. Actor clicks on the search icon to initiate the find process.  3. The system searches within the product catalog and matches the products with similar names or keywords to the user's search keyword. [E1]  4. The system displays a list of products that match the find criteria. |
| Alternate Flow |  |
| Exception Flow | [E1] If there are no products that match the find criteria, the  system displays a message "No results found" |

#### 2.2.3.7. Use Case “Filter Product”

Table 9 Use Case “Filter Product”

|  |  |
| --- | --- |
| Use Case ID | UC\_FP |
| Use Case Name | Filter Product |
| Description | This use case allows the actor to filter products based on criteria such as price, brand, rating, etc |
| Actor | Guest, Member |
| Pre-  Condition(s) |  |
| Post-  Condition(s) | The system displays products that match the actor's filtering criteria |
| Basic Flow | 1. Actor clicks “Shop” on the navigation bar.  2. The system displays filtering criteria (e.g., price range,  brand, category)  3. Actor selects the filtering criteria they want to apply.  4. The system filters products based on the actor's criteria. [E1]  5. The system displays the filtered products list. |
| Alternative Flow |  |
| Exception Flow | [E1] If there are no products that match the filtering criteria, display "No result is found". |

#### 2.2.3.8. Use Case “Manage Cart”

Table 10 Use Case “Manage Cart”

|  |  |
| --- | --- |
| Use Case ID | UC\_MC |
| Use Case Name | Manage Cart |
| Description | This use case allows Actor to manage their shopping cart, including adding products to the cart, editing the quantity of products in the cart, removing products from the cart. |
| Actor | Member |
| Pre-  Condition(s) | Actor has logged in to the website. |
| Post-  Condition(s) | Actor actor has logged in to the website. |
| Basic Flow | 1. Actor clicks on the "Cart Icon" on the navigation bar of the website.  2. The system displays Actor's shopping cart information (including the list of products, quantity of each product, the price of each product, and the total price of the cart, product payment button).[E1]  3. Actor manages the shopping cart (Actor can add, remove, or edit the number of products in the cart). [A1]  4. The system updates the changes and saves them to the  database. |
| Alternative Flow | [A1] 3.1. If Actor clicks on the "Checkout" function, the system will transfer to the “Purchase” Use Case |
| Exception Flow | [E1] 2.2. If the shopping cart has no products, the system will display the message “Your shopping cart is empty”. |

#### 2.2.3.9. Use Case “Add Product To Cart”

Table 11 Use Case “Add Product To Cart”

|  |  |
| --- | --- |
| Use Case ID | UC\_APTC |
| Use Case Name | Add Product To Cart |
| Description | This use case allows Actor to add product to cart. |
| Actor | Member |
| Pre- Condition(s) | The actor has logged in to the website. |
| Post- Condition(s) | The actor successfully add the product to the shopping cart. |
| Basic Flow | 1. The actor selects the product which want to add to the cart. 2. Click ‘Add product ’ 3. The system updates the changes about the cart. |
| Alternative Flow |  |
| Exception Flow |  |

#### 2.2.3.10. Use Case “Update Cart Quantity”

Table 12 Use Case “Update Cart Quantity”

|  |  |
| --- | --- |
| Use Case ID | UC\_UCQ |
| Use Case Name | Update Cart Quantity |
| Description | This use case allows the Actor to increase or decrease the quantity of a product or products in the cart. |
| Actor | Member |
| Pre- Condition(s) | The actor has logged in to the website.  The actor need to have one or more products in their cart. |
| Post- Condition(s) | Actor updates the shopping cart successfully |
| Basic Flow | 1. Actor clicks the "Cart Icon" on the website's navigation bar. 2. The system displays the Actor's shopping cart with a list of quantities of one or more products. 3. The actor can edit the quantity of 1 or more products in the shopping cart to increase or decrease and cannot reduce the quantity to 0. 4. The system updates changes to the shopping cart. |
| Alternative Flow |  |
| Exception Flow |  |

#### 2.2.3.11. Use Case “Delete Cart Products”

Table 13 Use Case “Delete Cart Products”

|  |  |
| --- | --- |
| Use Case ID | UC\_DCP |
| Use Case Name | Delete Cart Products |
| Description | This use case allows Actor to remove products from cart. |
| Actor | Member |
| Pre- Condition(s) | The actor has logged in to the website.  Actors need to have one or more products in their cart. |
| Post- Condition(s) | The actor successfully removes the product from the shopping cart. |
| Basic Flow | 1. Actor clicks on the "Cart Icon" on the navigation bar of the website. 2. The system displays Actor's shopping cart the list of products 3. Actor clicks on the trash icon right at the product he wants to delete to remove the product in the shopping cart. 4. The system updates the changes about the cart. |
| Alternative Flow |  |
| Exception Flow |  |

#### 2.2.3.12. Use Case “Buy Product”

Table 14 Use Case “Buy Product”

|  |  |
| --- | --- |
| Use Case ID | UC\_BP |
| Use Case Name | Buy Product |
| Description | This use case allows the Actor to order the selected product |
| Actor | Member |
| Pre-  Condition(s) | The actor has logged in to the website. |
| Post-  Condition(s) | The actor has placed a successful order. |
| Basic Flow | 1. Actor accesses the shopping cart on the website.  2. The system displays the list of products in the shopping cart.  3. Actor confirms the product information and quantity.  4. Actor clicks Checkout  5. The system displays the shipping detail on page  6. The actor fills in shipping information  7. The actor clicks Order  8. The system checks order information [E1]  9. The system displays “Order successfully!” |
| Alternative Flow |  |
| Exception Flow | [E1] If the Actor does not enter complete or valid information, the system displays an error message and requests the Actor to correct the information. |

#### 2.2.3.13. Use Case “Manage Account”

Table 15 Use Case “Manage Account”

|  |  |
| --- | --- |
| Use Case ID | UC\_MA |
| Use Case Name | Use Case Manage Account |
| Description | This use case allows the Manager to manage member accounts |
| Actor | Manager |
| Pre-  Condition(s) | A manager has logged in to the system. |
| Post-  Condition(s) |  |
| Basic Flow | 1. The manager assesses the admin page  2. The manager chooses the user management.  3. The system displays a list of user accounts.  4. The manager selects the account and clicks the edit icon. [A1]  5. The system displays detailed information about the account.  6. Manager enters the updated information for the account.  7. Manager clicks the "Update User Now" button to confirm the changes.  8. The system verifies the updated information. [E1]  9. The system saves the changes |
| Alternative Flow | [A1].1 Manager selects the Delete account  [A1].1.1 The system deletes the account from the database.  [A1].2 Manager selects Create account  [A1].2.1 The system displays the Create Account page  [A1].2.2 Actor enters account info and clicks Create Account  [A1].2.3 The system verifies the account info. [E1]  [A1].2.4 The system saves the changes |
| Exception Flow | [E1] Manager does not enter complete or valid information, the system displays an error message and requests the Manager to correct the information |

#### 2.2.3.14. Use Case “Manage Personal account”

Table 16 Use Case “Manage Personal account”

|  |  |
| --- | --- |
| Use Case ID | UC\_MPA |
| Use Case Name | Manage Personal Account |
| Description | This use case allows the Actor to manage personal account |
| Actor | Member |
| Pre-  Condition(s) | Actor has logged into the system by member account. |
| Post-  Condition(s) |  |
| Basic Flow | 1. Actor accesses the home page.  2. Actor clicks on the Avatar icon and chooses "Quản lý thông tin” to view the information.  3. The system displays a information of actor.  4. Actor clicks on the "chỉnh sửa thông tin”. [A1]  5. The system saves the changes and updates the Information |
| Alternative Flow | [A1].1. Actor proceed edit their information . |
| Exception Flow |  |

#### 2.2.3.15. Use Case “Manage Product”

Table 17 Use Case “Manage Product”

|  |  |
| --- | --- |
| Use Case ID | UC\_MP |
| Use Case Name | Manage product |
| Description | Allows Actors to manage products including viewing information, adding, editing information and deleting products. |
| Actor | Manager |
| Pre-Condition(s) | Actor has logged in to the system. |
| Post-Condition(s) | The product information has been successfully updated. |
| Basic Flow | 1. Actor accesses the admin page  2. Actor chooses the product management.  3. The system displays the list of products currently available on the website.  4. Actor selects a product and clicks the edit icon. [A1]  5. The system displays detailed information about the product.  6. Actor enters the updated information for the product.  7. Actor clicks the “Update Product Now” button to confirm the changes.  8. The system verifies the updated information.[E1]  9. The system saves the changes |
| Alternate Flow | [A1].1 Actor selects the "Delete account icon"  [A1].1.1 The system deletes the account from the database.  [A1].2 Actor selects "Create account icon"  [A1].2.1 The system displays the Create account page  [A1].2.2 Actor enters account info and clicks Create Account  [A1].2.3 The system verifies the account info.  [A1].2.4 The system saves the changes |
| Exception Flow | [E1] If Actor does not enter complete or valid information, the  system displays an error message and requests Actor to correct the  information. |

#### 2.2.3.16. Use Case “Manage Order”

Table 18 Use Case “Manage Order”

|  |  |
| --- | --- |
| Use Case ID | UC\_OM |
| Use Case Name | Manage Order |
| Description | This use case allows the Actor to manage the shipping status of the orders |
| Actor | Manager |
| Pre-  Condition(s) | Actor has logged into the system by manager account. |
| Post-  Condition(s) | The order's shipping status has been successfully updated |
| Basic Flow | 1. Actor accesses the management page.  2. Actor clicks on the "Select Management" bar and chooses "Order Management”.  3. The system displays a list of orders and their details. [E1]  4. Actor clicks on the "Order is placed”. [A1]  5. The system saves the changes and updates the status bar to "Order is in process." |
| Alternative Flow | [A1].1 Actor repeats step 4 and clicks on "Order is on process".  [A1].1.1 The system saves the changes and update status bar to “Order is transcended to the Shipping Agency”.  [A1].2. The Actor repeats step [A1].1 and clicks on "Order is transcended to the Shipping Agency".  [A1].2.1 The system saves the changes and update status bar "Order on transit".  [A1].3 The Actor repeats step [A1].2 and clicks on "Order on transit".  [A1].3.1 The system saves the changes and update status bar “Delivered”. |
| Exception Flow | [E1] If there are no orders in the order list, the system displays the message "No order is found". |

#### 2.2.3.17. Use Case “View Order History”

Table 19 Use Case “View Order History”

|  |  |
| --- | --- |
| Use Case ID | UC\_VOH |
| Use Case Name | View Oder History |
| Description | This use case allows the Actor to manage the status of his order, including reviewing order details. |
| Actor | Member |
| Pre- Condition(s) | The actor has logged in to the website.  Actors must ensure that they have an order that has just been placed. |
| Post- Condition(s) | Successful actor reviews recent product transaction history in the shopping cart. |
| Basic Flow | 1. The actor clicks "User Icon" on the site's navigation bar. 2. The system displays 3 items: user information, my orders, and log out. 3. Actor clicks "My Orders". 4. The system displays order history (including order status, order cancellation, view order details) |
| Alternative Flow |  |
| Exception Flow |  |

#### 2.2.3.18. Use Case “View Order Details”

Table 20 Use Case “View Order Details”

|  |  |
| --- | --- |
| Use Case ID | UC\_VOD |
| Use Case Name | View Oder Details |
| Description | This use case allows the Actor to review order details. |
| Actor | Member |
| Pre- Condition(s) | Actors must ensure that they have an order that has just been placed. |
| Post- Condition(s) | The actor successfully reviews the details of the order just placed |
| Basic Flow | 1. Actor clicks on the "User Icon" on the website's navigation bar. 2. The system displays 3 items: user information, my orders, and log out. 3. Actor clicks "My Orders". 4. The system displays order history (including order status, order cancellation, view order details) 5. Actor clicks on the “View order detail” for the order the actor wants to view |
| Alternative Flow |  |
| Exception Flow |  |

# CHAPTER 3 TEST PLAN



## 3.1. SCOPE

### 3.1.1. In Scope

The following items are included in the scope of the test:

- Some functional, application performance, security, and use case requirements are being tested.

- In our system, we have quality standards and fit metrics.

- End-to-end testing of all systems that interact with the system, as well as testing of their interfaces.

### 3.1.2. Out of Scope

The following one is considered out of scope for student management systemtest plan and testing scope: Functional requirements testing for systems outside the system.

## 3.2. QUALITY OBJECTIVE

### 3.2.1. Primary Objective

The system satisfies all of the requirements, containing quality and non-functional requirements as well as fit metrics for each quality requirement.

The system meets the use case scenarios and maintains the product's quality.

The user should find that the project has met or surpassed all of their expectations as outlined in the requirements at the end of the project development cycle.

### 3.2.2. Secondary Objective

* Identify and uncover any concerns and hazards that are related with them.
* Inform the project team of all known difficulties.
* Before releasing, make sure that all concerns have been resolved properly.

## 3.3. ROLES AND RESPONSIBILITIES

**- Developers:** Responsible to:

+ Develop the application.

+ Develop use cases and requirements.

+ Perform unit, system, regression, and integration testing.

+ Assist with user acceptability testing.

**- Management Team:**

+ Maintain testing integrity and assist with testing operations.

+ Organize events among cancer centers.

+ Add extra as needed to the testing scope.

## 3.4. ASSUMPTIONS FOR TEST EXECUTION

- The Developer team has completed unit, system, and integration testing and has satisfied all of the requirements for User Acceptance testing.

- End-users will undertake user acceptance testing.

- The results of the tests will be disclosed on a daily basis. Failed scripts and a defect list with supporting evidence will be delivered straight to the developer.

- For User Acceptance testing, use scenarios have been created. The test lead approves use cases.

- Scripts for tests are written and authorized.

- Developers will be supported and given necessary direction by the Test Team while they do test.

- Following the testing kick-off meeting, major dependencies should be reported.

## 3.5 TESTING METHODS

### 3.5.1. Overview

- Define testing methodologies for each region and sub-area, taking into account both functional and non-functional needs.

- Establish mechanisms for bug tracking.

- Identify the dangers of testing.

- Identify the resources and information that are necessary.

- Provide a schedule for testing.

### 3.5.2. Boundary Value Analysis

Programming iterative algorithms tells us that the error usually lies in boundary (beginning or end) of some continuous interval (corresponding equivalence class). Therefore, we will focus on creating test cases corresponding to the prices value at this boundary.

The idea of testing techniques based on boundary values is prescriptive meaning of test cases corresponds to values right on or near the boundary of each equivalence class. Therefore this technique is just suitable with correspondence classes defined by continuous values (number integer, real number), but it is not suitable for equivalence classes is determined by the enumeration values without interrelationships together.

Specific process for performing testing based on boundary values:

1) Identify equivalence classes based on the software component functional requirements specification.

2) Identify the two boundaries of each corresponding class

3) Create test cases for each boundary of each corresponding class:

a. 1 testcase for boundary value

b. 1 testcase below the boundary

c. 1 testcase above the boundary

### 3.5.3. Decision Table

A decision table is a very useful tool for specifying software requirements or for specifying a software system design. It describes the complex business rules that the software must have done in an easy-to-read and control form.

Specific process to perform testing using decision table:

1) Find the decision sheet from the software component functional requirements specification or from the software component design sheet. If it does not exist, build it based on the specification of functional requirements or based on the software component design table.

2) From the decision table, it is converted into a table of test cases in which each column describing a rule is converted into 1 to n columns describing test cases corresponding to that rule.

a. If the input condition is a logical value, each rule column is converted into a testcase column.

b. If the input condition is an equivalence class (many continuous values), then each rule column is converted into multiple test cases based on the equivalence class technique or boundary value technique.

### 3.5.4. State Transition

Instead of describing complex business rules that the software must implement in an easy-to-read and control form such as decision tables, state transition maps record events that occur, and are then systematized. processing system as well as the system's responses. The state transition diagram is composed of three basic components: begin state, intermediate state, end.

While the state transition diagram is an easy-to-understand and easy-to-read way of describing the behavior of the TPPM, another form - the state transition table - can describe the behavior of the system TPPM more easily and is easier to handle automatically.

The state transition table includes 4 columns: current state, event occurred, action to be taken/result obtained, next state.

Based on the state transition diagram, we can easily define test cases:

1) Coverage level 1: create test cases so that each state occurs at least once

2) Coverage level 2: create test cases so that each event occurs at least once

3) Coverage level 3: create test cases so that all transition paths are tested. A transition path is a defined state transition path, starting from the input state and ending at the end state.

4) Coverage level 4: create test cases so that each linear transition path occurs at least once

### 3.5.5. Use Case

The testing technique that uses information in use-cases is the technique of defining test cases based on use case implementation scenarios.

Each use case diagram represents a small part of the software, it includes many functions and which actors these functions interact with.

Based on the specification of the main scenario and the extensions of the scenario, we will design test cases according to the following ideas:

1) At least 1 testcase to test the main scenario.

2) At least 1 test case for each possible extension.

3) If the main script or an extension is looped, there is no need to test the loop again.

## 3.6. TESTING COMPLETENESS

* The app has passed all of its tests, is stable, and satisfies all of its functional criteria.
* Automated test cases have passed in all areas.
* The Test Lead has signed off on each test area as finished.

## 3.7. TEST LEVELS

Testing of an application can be broken down into three primary categories and several sub-levels. The three primary categories include tests conducted every build (Build Tests), tests conducted every major milestone (Milestone Tests), and tests conducted at least once every project release cycle (Release Tests). The test categories and test levels are defined below:

### 3.7.1. Build Tests

***Level 1 - Build Acceptance Tests***

Build Acceptance Tests should take less than 2-3 hours to complete (15 minutes is typical). These test cases simply ensure that the application can be built and installed successfully. Other related test cases ensure that adopters received the proper Development Release Document plus other build related information (drop point, etc.). The objective is to determine if further testing is possible. If any Level 1 test case fails, the build is returned to developers un-tested.

***Level 2 - Smoke Tests***

Smoke Tests should be automated and take less than 2-3 hours (20 minutes is typical). These tests cases verify the major functionality a high level.

The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Level 2 test case fails, the build is returned to developers un-tested.

***Level 2a - Bug Regression Testing***

Every bug that was “Open” during the previous build, but marked as “Fixed, Needs Re-Testing” for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

### 3.7.2. Milestone Tests

***Level 3 - Critical Path Tests***

Critical Path test cases are targeted on features and functionality that the user will see and use every day.

Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

### 3.7.3. Release Tests

***Level 4 - Standard Tests***

Test Cases that need to be run at least once during the entire test cycle for this release. These cases are run once, not repeated as are the test cases in previous levels. Functional Testing and Detailed Design Testing (Functional Spec and Design Spec Test Cases, respectively). These can be tested multiple times for each Milestone Test Cycle (Iteration, Final Release, etc.).

Standard test cases usually include Installation, Data, GUI, and other test areas.

***Level 5 - Suggested Test***

These are Test Cases that would be nice to execute but may be omitted due to time constraints.

Most Performance and Stress Test Cases are classic examples of Suggested test cases (although some should be considered standard test cases). Other examples of suggested test cases include WAN, LAN, Network, and Load testing.

## 3.8. TESTING RISKS AND MITIGATION FACTORS

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Mitigation Plan** |
| **Schedule**  The testing timetable is constrained. The test cannot be prolonged beyond the UAT scheduled start date if the start of the testing is delayed owing to design work. | High | High | The testing team has control over the preparatory activities and early contact with all parties involved.  A contingency buffer has been included to the timetable, but not as much as best practices recommend. |
| **Resources**  There aren't enough resources, and the ones that are available are outdated (process takes around 15 days). | Medium | High | Holidays and vacations have been predicted and factored into the timetable; any variations from this estimate may result in testing delays. |
| **Defects**  Flaws are detected late in the cycle or towards the end of the cycle; defects discovered late are almost always due to ambiguous requirements and take a long time to address. | Medium | High | A defect management plan is in place to ensure that issues are communicated and fixed as soon as possible. |
| **Scope**  Scope completely demonstrated | Medium | Medium | The scope is clear, but the modifications to the functionality have not yet been decided or are still evolving. |
| Natural disasters | Low | Medium | Teams and duties have been distributed among two geographical zones. In the case of a catastrophic occurrence in one of the areas, resources will be available in the other areas to continue testing. |
| Inaccessibility and non-availability of an independent testing environment | Medium | High | The timetable is influenced by the lack of availability of the environment, resulting in a delay in the commencement of Test execution. |
| Testing has been postponed due to new issues. | Medium | High | There's a strong likelihood that some "new" problems will be discovered during testing and will become a problem that will take time to remedy. Because of the ambiguous document specification, problems might arise during testing. These flaws may result in a problem that will take long to remedy. If these concerns become show-stoppers, they will have a significant influence on the project's overall timetable. If new faults are detected, the defect management and issue management procedures are in place to give a resolution as soon as possible. |

# CHAPTER 4 TEST PLAN AND PROCESS



## 4.1. TESTING PROCESS

- The team applies a testing process consisting of 5 steps:

+ Test planning

+ Analysis and design of test scenarios

+ Setting up the environment and executing test scenarios.

+ Evaluating test results and reporting.

+ Closing the testing activity.

### 4.1.1. Building a test plan

- Define the scope of testing.

- Define the testing strategies.

- Identify risks and unexpected factors.

- Identify which testing activities are manual and which are automated.

- Estimate the cost of testing and create a testing schedule.

- Identify the testing environment.

### 4.1.2. Test Analysis and Design

- Design test cases from functional requirements and non-functional requirements of the software.

- The test cases should cover all aspects of testing for each software requirement.

- The test cases should cover all requirements in the testing strategies.

- If automated testing is needed, the Test Designer will build scripts based on the test cases/procedures.

### 4.1.3. Set up the necessary environment as required to run the program

- Set up the necessary environmental requirements to run the software.

- Execute the testing based on each test case.

- Perform exploratory testing that is not defined in the test cases.

- Retest the issues that have been fixed.

- The tester will create reports on the errors encountered during the testing process and track them until they have been resolved.

### 4.1.4. Evaluate test results and report findings

- Generate defect reports.

- Evaluate test results, and record changes to requirements.

- Measure and distribute test performance metrics.

- Prepare a summary table to evaluate the testing activity.

- Determine if success criteria have been met and if testing is complete.

### 4.1.5. Closing the Testing Activities

When all test cases have passed, no more test cases are generated, and the software meets the requirements, the testing process ends. At this point, the software is ready to be delivered to the customer.

## 4.2. TEST PLAN

### 4.2.1. Introduction

#### 4.2.1.1. Purpose

This test plan document sets out the following purposes:

- Identify basic information about the project and tested and untested functional components.

- List the requirements for testing (Test Requirements).

- The testing strategies should be used.

- The documents are prepared after the completion of the test.

- Describe the scope of testing, testing methods as well as the approach and application of each method to the system of “Accompany with Parent”.

#### 4.2.1.2. Overview

As the society is increasingly developed, people's living standards are improved, economic income is higher and higher, the mobile phone is no longer far away from everyone, it has become a An indispensable item for every citizen today. Most people equip themselves with a phone that suits their needs and pocket. However, with today's increasingly busy life, to want to buy a favorite phone, consumers have to go to the store to choose, so it will take a lot of time and effort.

The store system that sells mobile phones includes many other complex and difficult jobs. Therefore this topic building a sales website only focus on sales and stages such as management: human resources, order processing.

#### 4.2.1.3. Scope

Verify that all essential functions of the smartphones selling website are working as intended:

* Authentication.
* View products.
* Add products to cart.
* Modify quantity of products purchased.
* Purchase products.
* Manage user’s information
* Manage orders.

Verify all the management functions of the smartphones selling website:

* Manage products: add, edit, remove products.
* Manage users: edit, remove users.
* Export products / users statistics.

Verify the website's interaction:

* Navigation.
* Search functionality.
* Product listings.
* Product filtering.
* Spelling of website’s contents.

#### 4.2.1.4. Document Users

This document is used for Test Manager, Test Designer and Tester.

#### 4.2.1.5. References

- A large project for students in theSoftware Testing course

- Sample documents for test plan

- Sample documents for test plan

- Lecture slides for Software Testing course

### 4.2.2. Work schedule

Table 21 Work Schedule

|  |  |  |
| --- | --- | --- |
| **Task** | **Document** | **Duration** |
| Test planning | Test Plan Documentation | 1 days |
| Review the documents | Test Plan Documentation | 1 days |
| Design test cases | Testcase Documentation | 1 days |
| Write test cases | Testcase Documentation | 2 days |
| Review test cases, set up test environment | Testcase Documentation | 1 days |
| Execute test cases | Testcase Documentation | 2 days |
| Record and evaluate test results | Testcase Documentation | 2 days |

### 4.2.3. Resource requirements

#### 4.2.3.1. Hardware

Personal computer with Internet connection.

#### 4.2.3.2. Software

- Postman

- Visual Studio Code

#### 4.2.3.3. Testing tools

- Test Case Management

- Configuration Management

- Defect Tracking

#### 4.2.3.4. Testing Environment

Personal computer with Internet connection to access the website.

#### 4.2.3.5. Test role

Table 22 Test Role

|  |  |
| --- | --- |
| **Member** | **Role** |
| Trương Chí Kiên | Test Designer / Tester: Design and write test cases, execute test cases for functionality on user module, review Test Plan |
| Nguyễn Hoàng Nhân | Test Manager / Test Designer / Tester: Make a test plan, manage the progress of test activities, design additional test cases and execute test cases on the user module, review the Test Plan |
| Lê Hoàng Lâm | Test Designer / Tester: Design and write test cases, execute test cases for functionality on admin module, review Test Plan |
| Võ Minh Hưng | Test Designer / Tester: Design and write test cases, execute test cases for functionality on admin module, review Test Plan |

### 4.2.4 Testing strategy

- The General Testing Process Approach will be used to perform testing.

Diagram

Description automatically generated

Figure 6 Testing strategy

- Functional testing.

Table 23 Functional testing

|  |  |
| --- | --- |
| Purpose of testing | Ensure that the functions are tested and operating accurately according to the required specifications. |
| Technology | Execute all possible test cases for each functional group, using both valid and invalid data, to determine:   * The expected results when valid data is used. * Appropriate warnings are displayed when invalid data is used. |
| Stopping criterion | All designed test cases have been executed.  All found errors have been recorded with clear reasons to assist developers in fixing them. |
| Responsibility for testing. | Test Designer / Tester |
| Testing approach | Manual testing is conducted sequentially following the steps defined in the test cases... |
| Exception handling | List all issues encountered during the testing process... |

### 4.2.5. Acceptance criteria

- Pass all defined test cases.

- The system runs stably on various mobile device.

# CHAPTER 5 ACTUAL TEST

## 5.1 WHITEBOX TESTING

### 5.1.1 Create Product Function

const createProduct = (newProduct) => { (1)

    return new Promise(async (resolve, reject) => { (2)

        const { name, image, type, price, countInStock, rating, description } = newProduct (3)

        try { (4)

            const checkProduct = await Product.findOne({

                name: name

            }) (5)

            if (checkProduct !== null) { (6)

                resolve({ (7)

                    status: 'ERR',

                    message: 'The name of product is already have'

                })

            }

            const newProduct = await Product.create({

                name, image, type, price, countInStock, rating, description

            }) (8)

            if (newProduct) { (9)

                resolve({ (10)

                    status: 'OK',

                    message: 'SUCCESS',

                    data: newProduct

                })

            }

        } catch (e) { (11)

            reject(e) (12)

        }

    })

}

A diagram of a diagram

Description automatically generated

Figure 7 Create Product Data Flow Graph

Cyclomatic complexity = P(decision nodes) + 1 = 2 + 1 = 3

Table 24 Create Product Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-5-6-7 | * name = Iphone 15 Promax 512G Tím * image = iphone15.png * type = phone * price = 35000000 * countInStock = 50 * rating = 5 * description = Iphone 15 | Status: ‘ERR’  Message: ‘The name of product already have’ |
| 2 | 1-2-3-4-5-6-8-9-10 | * name = Laptop Apple Macbook Air M2 * image = macbookairm2.png * type = laptop * price = 26990000 * countInStock = 16 * rating = 5 * description = Màn hình 13.6-inch; 2560 x 1664… | Status: ‘OK’  Message: ‘SUCCESS’  Data: newProduct |
| 3 | 1-2-3-4-5-6-8-9-11-12 |  |  |

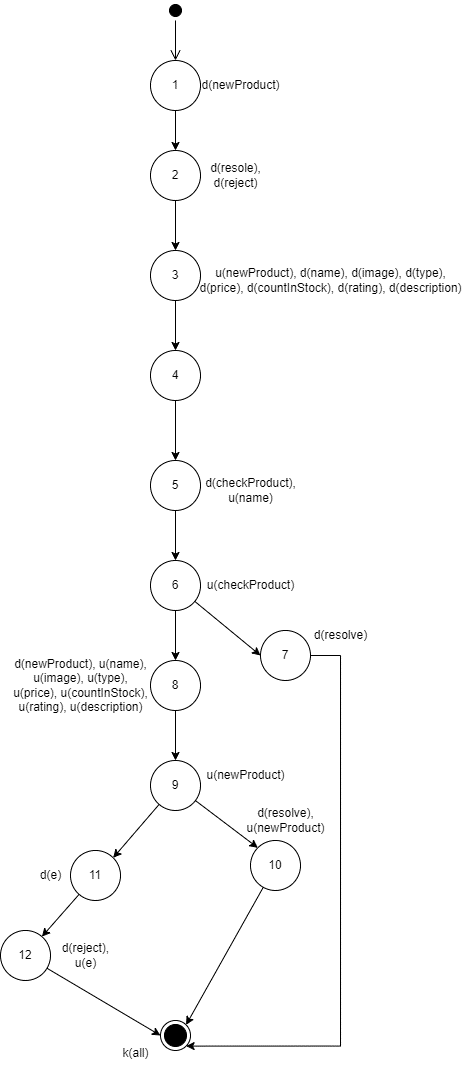


Figure 8 Create Product Data Flow Graph

Table 25 Create Product Variable Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | newProduct | 1. ~duk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~ddk  2. ~ddk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~dk  2. ~dk  3. ~ddk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 4 | name | 1. ~duk  2. ~duuk  3. ~duuk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 5 | image | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 6 | type | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 7 | price | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 8 | countInStock | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 9 | rating | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 10 | description | 1. ~dk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 11 | checkProduct | 1. ~duk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 12 | e | 1. ~duk | Scenario above does not contain any abnormal activity pairs. |

### 5.1.2 Update Product Function

const updateProduct = (id, data) => {                           //1

    return new Promise(async (resolve, reject) => {             //2

        try {

            const checkProduct = await Product.findById({       //3

                \_id: id

            })

            if (checkProduct === null) {                        //4

                resolve({                                       //5

                    status: 'ERR',

                    message: 'The product is not defined'

                })

            }

            // console.log('checkProduct', checkProduct)

            const updateProduct = await Product.findByIdAndUpdate(id, data, {new : true})       //6

            resolve({                                           //7

                status: 'OK',

                message: 'SUCCESS',

                data: updateProduct

            })

        } catch (error) {                                       //8

            console.error('Error generating tokens:', error);

            reject({                                            //9

                status: 'ERR',

                message: 'Token generation failed'

            });

        }

    })

}

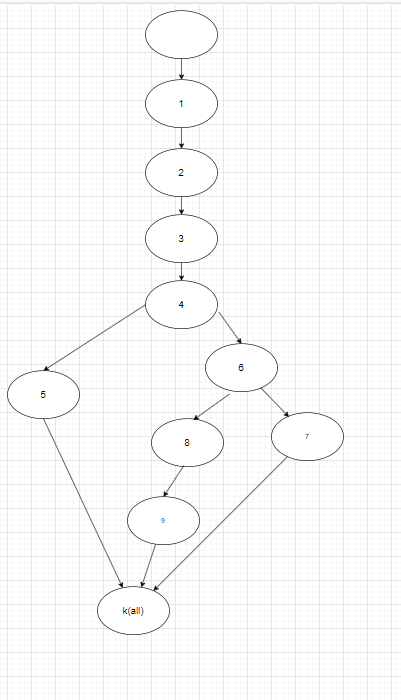


Figure 9 Update Product Control Flow Graph

Cyclomatic complexity = P(decision nodes) + 1 = 2 + 1 = 3

Table 26 Update Product Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-6-7 | id = 6549145b95e87cb95b25c6a0  data: discount = 2 | Status: ‘OK’  Message: ‘SUCCESS’  Data: updateProduct |
| 2 | 1-2-3-4-5 | id = 6549145b95e87cb95b25c6  data: discount = 2 | Status: ‘ERR’  Message = ‘The product is not defined’ |
| 3 | 1-2-3-4-6-8-9 |  |  |

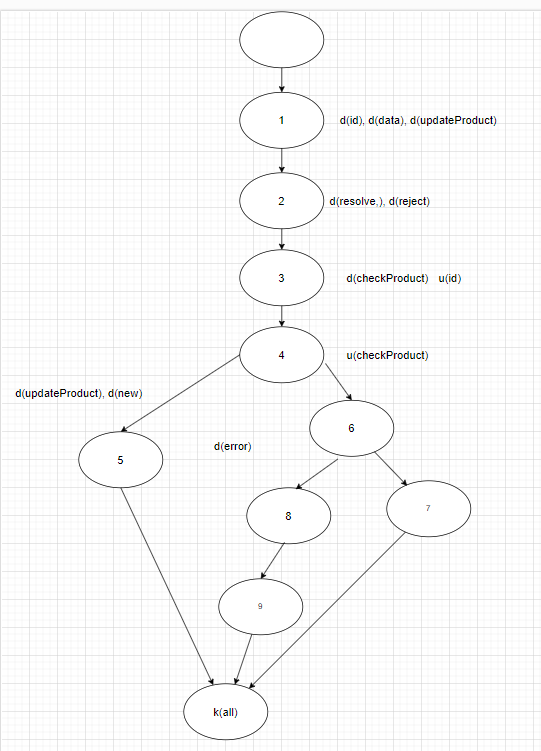


Figure 10 Update Product Data Flow Graph

Table 27 Update Product Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | id | 1. ~duk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | data | 1. ~dk  2. ~dk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 3 | updateProduct | 1. ~dk  2. ~ddk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 4 | resolve | 1. ~dk  2. ~dk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 5 | reject | 1. ~dk  2. ~dk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 6 | checkProduct | 1. ~duk  2. ~dk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 7 | error | 1. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 8 | new | 1. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |

### 5.1.3 Get Details Product Function

const getDetailsProduct = (id) => { 1 d(a)

    return new Promise(async (resolve, reject) => { 2 d(b), d(c)

        try { 3

            const product = await Product.findOne({

                \_id: id

            }) 4 d(d), u(a)

            if (product === null) { 5 u(d)

                resolve({ 6 d(b)

                    status: 'ERR',

                    message: 'The product is not defined'

                })

            }

            resolve({ 7 d(b), u(d)

                status: 'OK',

                message: 'Get product details SUCCESS',

                data: product

            })

        } catch (error) { 8 d(e)

            reject({ 9 d(c)

                status: 'ERR',

                message: 'Token generation failed'

            });

        }

    })

}

A diagram of a network

Description automatically generated

Figure 11 Get Details Product Control Flow Graph

Cyclomatic complexity = P(decision nodes) + 1 = 2 + 1 = 3

Table 28 Get Details Product Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-8-9 |  |  |
| 2 | 1-2-3-4-5-6 | id = 6549145b95e87cb95b25c6 | Status: ‘ERR’  Message: ‘The product is not defined’ |
| 3 | 1-2-3-4-5-7 | Id = 6549145b95e87cb95b25c6a0 | Status: ‘OK’  Message: ‘Get product details SUCCESS’  Data: product |

A diagram of a diagram

Description automatically generated

Figure 12 Get Details Product Data Flow Graph

Table 29 Get Details Product Variable Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | id | 1. ~duk  2. ~duk  3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~ddk  2. ~ddk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~dk  2. ~dk  3. ~ddk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 4 | product | 1. ~duk  2. ~duuk  3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 5 | error | 1. ~dk | Scenario above does not contain any abnormal activity pairs. |

### 5.1.4 Delete Product Function

const deleteProduct = (id) => { (1)

    return new Promise(async (resolve, reject) => { (2)

        try {

            const checkProduct = await Product.findById({ (3)

                \_id: id

            })

            if (checkProduct === null) { (4)

                resolve({ (5)

                    status: 'ERR',

                    message: 'The product is not defined'

                })

            }

            await Product.findByIdAndDelete(id) (6)

            resolve({ (7)

                status: 'OK',

                message: 'Delete product SUCCESS',

            })

        } catch (error) { (8)

            reject({ (9)

                status: 'ERR',

                message: 'Token generation failed'

            });

        }

    })

}

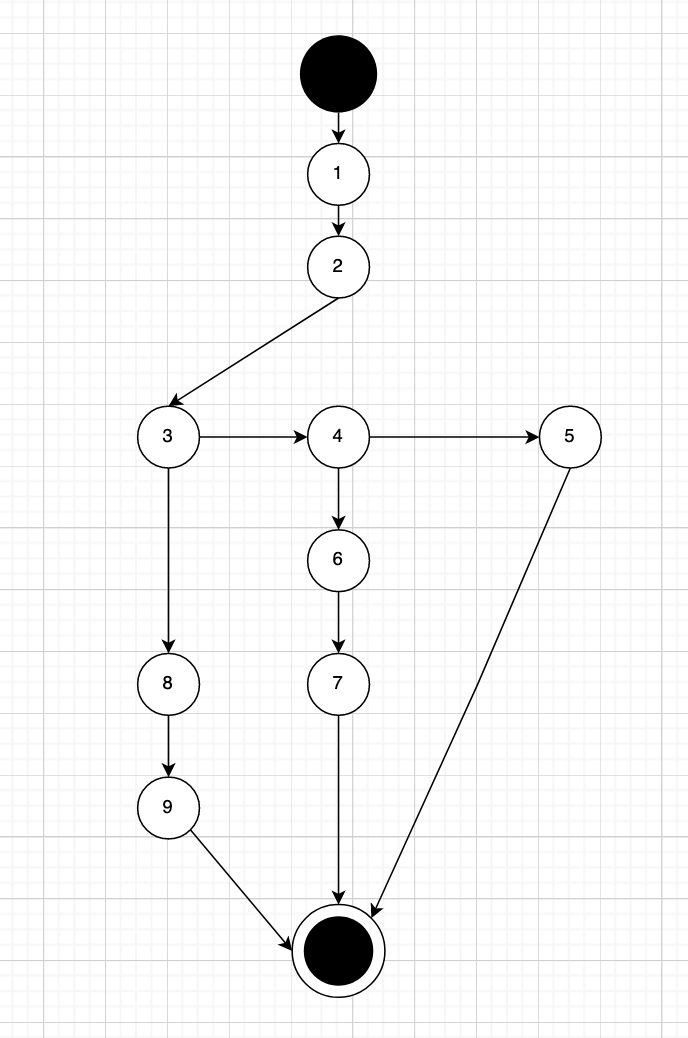


Figure 13 Delete Product Control Flow Graph

Cyclomatic complexity = P (decision nodes) + 1 = 2 + 1 = 3

Table 30 Delete Product Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-8-9 |  |  |
| 2 | 1-2-3-4-6-7 | id= 6549145b95e87cb95b25c6a0 | Status: ‘OK’  Message: ‘Delete product SUCCESS’ |
| 3 | 1-2-3-4-5 | id = 6549145b95e87cb95b25c6 | Status: ‘ERR’  Message: ‘The product is not defined’ |

A diagram of a diagram

Description automatically generated

Figure 14 Delete Product Data Flow Graph

Table 31 Delete Product Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | id | 1. ~duk 2. ~duuk 3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~dk 2. ~duk 3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~duk 2. ~dk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 4 | checkProduct | 1. ~duk 2. ~dk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |

### 5.1.5 Delete Many Product Function

const deleteManyProduct = (ids) => {                        (1)

    return new Promise(async (resolve, reject) => {         (2)

        try {                                               (3)

            await Product.deleteMany({ \_id: ids })          (4)

            resolve({

                status: 'OK',

                message: 'Delete product success',

            })                                              (5)

        } catch (e) {                                       (6)

            reject(e)                                       (7)

        }

    })

}

A diagram of a diagram

Description automatically generated

Figure 15 Delete Many Product Control Flow Graph

Cyclomatic complexity = P (decision nodes) + 1 = 1 + 1 = 2

Table 32 Delete Many Product Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-5 | Ids: [  id = 6549145b95e87cb95b25c6a0,  id = 65524c9efead3cc5bc9302ef] | Status: ‘OK’  Message: ‘Delete product success’ |
| 2 | 1-2-3-4-6-7 |  |  |

A diagram of a diagram

Description automatically generated

Figure 16 Delete Many Product Data Flow Graph

Table 33 Delete Many Product Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | ids | 1. ~duk 2. ~dk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~duk 2. ~dk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~dk 2. ~ddk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 4 | e | 1. ~duk | All 2 scenarios above do not contain any abnormal activity pairs. |

### 5.1.6 Get All Type Product Function

const getAllType = () => {

    return new Promise(async (resolve, reject) => {         (1)

        try {                                               (2)

            const allType = await Product.distinct('type')  (3)

            resolve({

                status: 'OK',

                message: 'Success',

                data: allType,

            })                                              (4)

        } catch (e) (5)

{

            reject(e)                                       (6)

        }

    })

}

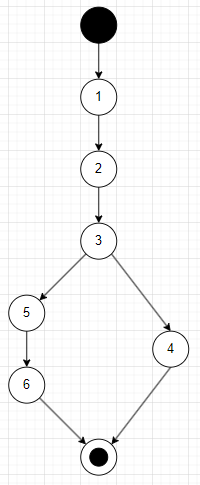


Figure 17 Get All Types Control Flow Graph

Cyclomatic complexity = P (decision nodes) + 1 = 1 + 1 = 2

Table 34 Get All Type Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Test Data** | **Expected Value** |
| 1 | 1-2-3-4 |  | status: ‘OK’  message: ‘Success’  data: allType |
| 2 | 1-2-3-5-6 |  |  |

A diagram of a diagram

Description automatically generated

Figure 18 Get All Type Data Flow Graph

Table 35 Get All Type Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | allType | 1. ~duk 2. ~dk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~duk 2. ~dk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~dk 2. ~ddk | All 2 scenarios above do not contain any abnormal activity pairs. |
| 4 | e | 1. ~duk | All 2 scenarios above do not contain any abnormal activity pairs. |

### 5.1.7 Create User Function

const createUser = (newUser) => { (1)

    return new Promise(async (resolve, reject) => { (2)

        const {name, email, password, confirmPassword, phone}=newUser (3)

        try {

            const checkUser = await User.findOne({ (4)

                email: email

            })

            if (checkUser !== null) { (5)

                resolve({ (6)

                    status: 'ERR',

                    message: 'The email is already have'

                })

            }

            const hash = bcrypt.hashSync(password, 10) (7)

            const createdUser = await User.create({ (8)

                name,

                email,

                password: hash,

                phone

            })

            if (createdUser) { (9)

                resolve({ (10)

                    status: 'OK',

                    message: 'SUCCESS',

                    data: createdUser

                })

            }

        } catch (e) { (11)

            reject(e) (12)

        }

    })

}

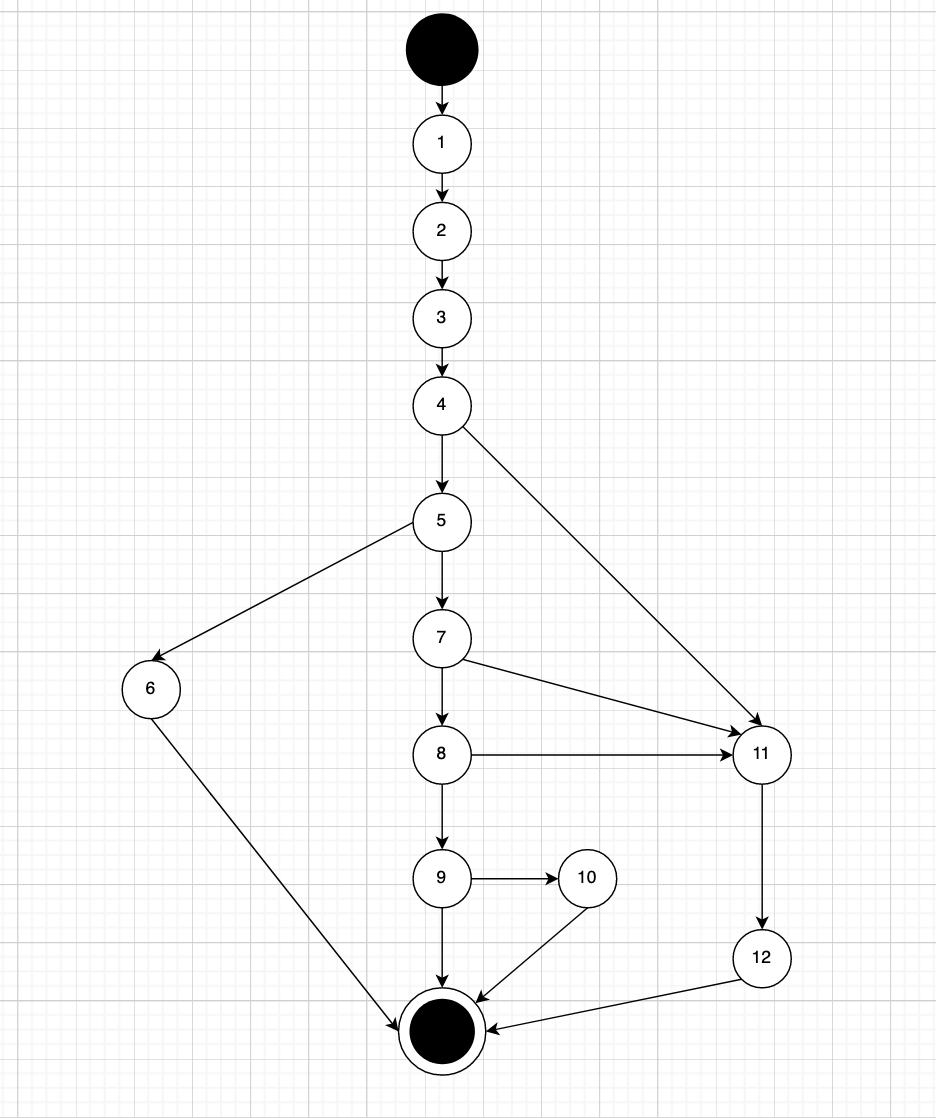


Figure 19 Create User Control Flow Graph

Cyclomatic complexity = P (decision nodes) + 1 = 5 + 1 = 6

Table 36 Create User Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-5-6 | * "email": "hung@gmail.com" * "password": "123" * "confirmPassword": "123" * “phone”: 0912345678 | Status: ‘ERR’  Message: ‘The email is already have’ |
| 2 | 1-2-3-4-5-7-8-9 |  |  |
| 3 | 1-2-3-4-5-7-8-9-10 | * "email": "hungnew@gmail.com" * "password": "123" * "confirmPassword": "123" * “phone”: 0912345678 | Status: ‘OK’  Message: ‘SUCCESS’ |
| 4 | 1-2-3-4-5-7-11-12 |  |  |
| 5 | 1-2-3-4-5-7-8-11-12 |  |  |
| 6 | 1-2-3-4-11-12 |  |  |

A diagram of a diagram

Description automatically generated

Figure 20 Create User Data Flow Graph

Table 37 Create User Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | newUser | 1. ~duk 2. ~duk 3. ~duk 4. ~duk 5. ~duk 6. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | resolve | 1. ~duk 2. ~dk 3. ~duk 4. ~dk 5. ~dk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 3 | reject | 1. ~duk 2. ~duk 3. ~duk 4. ~duk 5. ~duk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 4 | checkUser | 1. ~duk 2. ~dk 3. ~dk 4. ~duk 5. ~duk 6. ~duk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 5 | createdUser | 1. ~duk 2. ~duuk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 6 | hash | 1. ~dk 2. ~duk 3. ~duk 4. ~duk 5. ~duk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 7 | email | 1. ~duk 2. ~duuk 3. ~duuk 4. ~duuk 5. ~duk 6. ~duk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 8 | password | 1. ~dk 2. ~duuk 3. ~duuk 4. ~duuk 5. ~duuk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 9 | name | 1. ~dk 2. ~duk 3. ~duk 4. ~duk 5. ~dk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 10 | confirmedPassword | 1. ~dk 2. ~duk 3. ~duk 4. ~duk 5. ~dk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |
| 11 | phone | 1. ~dk 2. ~duk 3. ~duk 4. ~duk 5. ~dk 6. ~dk | All 6 scenarios above do not contain any abnormal activity pairs. |

### 5.1.8 Delete User Function

const deleteUser = (id) => {                                  //1

    return new Promise(async (resolve, reject) => {           //2

        try {

            const checkUser = await User.findById({           //3

                \_id: id

            })

            if (checkUser === null) {                         //4

                resolve({                                     //5

                    status: 'ERR',

                    message: 'The user is not defined'

                })

            }

            await User.findByIdAndDelete(id)                   //6

            resolve({                                          //7

                status: 'OK',

                message: 'Delete user SUCCESS',

            })

        } catch (error) {                                      //8

            console.error('Error generating tokens:', error);

            reject({                                           //9

                status: 'ERR',

                message: 'Token generation failed'

            });

        }

    })

}

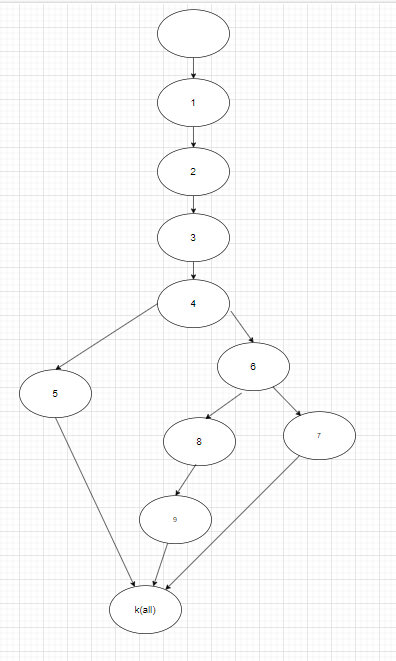


Figure 21 Delete User Control Flow Graph

Cyclomatic complexity = P (decision nodes) = 2 + 1 = 3

Table 38 Delete User Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Flow** | **Data** | **Result** |
| 1 | 1-2-3-4-5 | id: 65328668afe86c2fc0fce3 | Status: ‘ERR’  Message: ‘The user is not defined’ |
| 2 | 1-2-3-4-6-7 | Id: 65328668afe86c2fc0fce318 | Status: ‘OK’  Message: ‘Delete user SUCCESS’ |
| 3 | 1-2-3-4-6-8-9 |  |  |

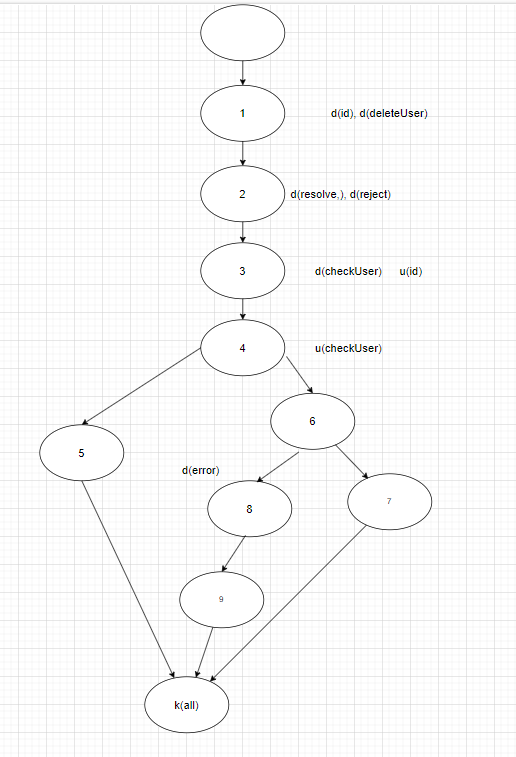


Figure 22 Delete User Data Flow Graph

Table 39 Delete User Variables Lifecycle

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Variable** | **Lifecycle** | **Conclusion** |
| 1 | id | 1. ~duk 2. ~duk 3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 2 | deleteUser | 1. ~dk 2. ~dk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 3 | resolve | 1. ~dk 2. ~dk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 4 | reject | 1. ~dk 2. ~dk 3. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 5 | checkUser | 1. ~duk 2. ~duk 3. ~duk | All 3 scenarios above do not contain any abnormal activity pairs. |
| 6 | err | 1. ~dk | All 3 scenarios above do not contain any abnormal activity pairs. |

# CHAPTER 6 BLACKBOX TESTING

## 6.1 BUG MANAGEMENT SYSTEM

### 6.1.1 Introduction of MantisBT

MantisBT, commonly known as Mantis, is an open-source web-based bug tracking system designed to help teams efficiently manage and track software defects throughout the development lifecycle. It provides a centralized platform for reporting, tracking, and resolving issues, making it an invaluable tool for quality assurance and testing processes.

### 6.1.2 Advantages of MantisBT

* Open Source: MantisBT is open-source software, meaning that it is freely available for use, modification, and distribution. This makes it accessible to a wide range of development and testing teams without incurring licensing costs.
* User-Friendly Interface: MantisBT offers a straightforward and user-friendly interface, making it easy for both technical and non-technical team members to create, track, and manage issues. This simplicity contributes to increased adoption and efficiency within the team.
* Customization: MantisBT allows teams to customize workflows, fields, and issue categories, tailoring the bug tracking process to match specific project requirements. This flexibility ensures that MantisBT can adapt to diverse development and testing workflows.
* Collaboration Features: MantisBT supports collaboration by enabling team members to add notes, comments, and attachments to issues. It also facilitates communication through email notifications and @mentions, ensuring that everyone involved is kept informed about issue updates.
* Integration Capabilities: MantisBT can be integrated with version control systems like Git and SVN, providing a seamless connection between bug tracking and source code repositories. This integration helps streamline the development and testing process.

### 6.1.3 Setting up and configuration

1. Download and Install:

* Download the latest version of MantisBT from the official website.
* Follow the installation instructions provided in the installation manual.

1. Access MantisBT:

* Once installed, access MantisBT through a web browser using the provided URL.
* Log in using the default administrator credentials.

1. User Management:

* Create user accounts for team members in the "Manage Users" section.
* Assign roles (e.g., developer, reporter) to users based on their responsibilities.

1. Project Configuration:

* Create a new project in the "Manage Projects" section.
* Configure project settings, including custom fields and workflows.

1. Issue Tracking:

* Start tracking issues by creating new issues through the "Report Issue" feature.
* Fill in issue details such as summary, description, and category.

1. Workflow Customization:

* Customize workflows and fields in the "Manage Custom Fields" and "Manage Workflows" sections to align MantisBT with your testing process.

1. Collaboration and Notifications:

* Enable and configure email notifications in the "Manage Configuration" section.
* Encourage team members to use notes, comments, and @mentions for effective collaboration.

1. Integration (Optional):

* Explore the "Manage Plugins" section to integrate MantisBT with version control systems if needed.

1. Documentation and Support:

* Refer to the official documentation for detailed guidance.
* Utilize the MantisBT community forums for support and discussions.

1. Regular Maintenance:

* Stay informed about updates and security patches.
* Refer to the upgrade guide when necessary.

## 6.2 BUG REPORTS

### 6.2.1 Bug report [LAM\_002]

A screenshot of a computer

Description automatically generated

Figure 23 Bug report [LAM\_002]

A screenshot of a computer

Description automatically generated

Figure 24 Activities in [LAM\_002]

### 6.2.2 Bug report [HNG\_001]

A screenshot of a computer

Description automatically generated

Figure 25 Bug report [HNG\_001]

A screenshot of a computer

Description automatically generated

Figure 26 Activities [HNG\_001]

### 6.2.3 Bug report [HNG\_002]

A screenshot of a computer

Description automatically generated

Figure 27 Bug report [HNG\_002]

A screenshot of a computer

Description automatically generated

Figure 28 Activities in [HNG\_002]

### 6.2.4 Bug report [KIN\_001]

A screenshot of a computer

Description automatically generated

Figure 29 Bug report [KIN\_001]

A screenshot of a computer

Description automatically generated

Figure 30 Activities in [KIN\_001]

### 6.2.5 Bug report [KIN\_002]

A screenshot of a computer

Description automatically generated

Figure 31 Bug report [KIN\_002]

A screenshot of a computer

Description automatically generated

Figure 32 Activities in [KIN\_002]

### 6.2.6 Bug report [KIN\_003]

A screenshot of a computer

Description automatically generated

Figure 33 Bug report [KIN\_003]

A screenshot of a computer

Description automatically generated

Figure 34 Activities in [KIN\_003]

### 6.2.7 Bug report [NHN\_002]

A screenshot of a computer

Description automatically generated

Figure 35 Bug report [NHN\_002]

A screenshot of a computer

Description automatically generated

Figure 36 Activities in [NHN\_002]

### 6.2.8 Bug report [LAM\_004]

A screenshot of a computer

Description automatically generated

Figure 37 Bug report [LAM\_004]

A screenshot of a computer

Description automatically generated

Figure 38 Activities in [LAM\_004]

### 6.2.9 Bug report [LAM\_006]

A screenshot of a computer

Description automatically generated

Figure 39 Bug report [LAM\_006]

A screenshot of a computer

Description automatically generated

Figure 40 Activities in [LAM\_006]

### 6.2.10 Bug report [LAM\_008]

A screenshot of a computer

Description automatically generated

Figure 41 Bug report [LAM\_008]

A screenshot of a computer

Description automatically generated

Figure 42 Activities in [LAM\_008]

### 6.2.11 Bug report [LAM\_011]

A screenshot of a computer

Description automatically generated

Figure 43 Bug report [LAM\_011]

A screenshot of a computer

Description automatically generated

Figure 44 Activities in [LAM\_011]

### 6.2.12 Bug report [LAM\_014]

A screenshot of a computer

Description automatically generated

Figure 45 Bug report [LAM\_014]

A screenshot of a computer

Description automatically generated

Figure 46 Activities in [LAM\_014]

### 6.2.13 Bug report [NHN\_009]

A screenshot of a computer

Description automatically generated

Figure 47 Bug report [NHN\_009]

A screenshot of a computer

Description automatically generated

Figure 48 Activities in [NHN\_009]

# CHAPTER 7 CONCLUSION

## 7.1 ARCHIEVEMENT

* Complete and capable of Software Testing
* Have the opportunity to test the information management function for administrators: product management, account management, order management, …
* Test majority of the business features and fix to appropriate with the description.

## 7.2 FAVORABLE

* Complete the assigned schedules.
* Try to stick to the set criteria.
* User-friendly application.

## 7.3 SHORTCOMING

* Have not fully tested the application.
* Some knowledge lack during working time, takes many time to review.

## 7.4 GENERAL KNOWLEDGE

* Learn more about building an app and testing an app.
* Learn more about testing tools, learn about new technologies and some other supporting software as Postman, MantisBT.
* Learn more about many aspects of error for preventing occurring again.

## 7.5 FUTURE DEVELOPMENT

* Step by step perfecting the application makes each part clear and friendly, easy for customers to see and use.
* Complete management system, easier for administrators.

# REFERENCES

1. *Carner, K. (n.d.). Lessons Learned in Software Testing.*

2. *Education, Đ. G. (n.d.). Retrieved from Youtube: https://www.youtube.com/@DucGiangTesterEducation*

3. *GeekforGeek, S. (n.d.). GeekforGeek, StackOverflow.*

4. *Hiệp, N. V. (n.d.). Lecturer Resources.*

5. *Tutorial, L. P. (n.d.). Retrieved from Youtube: https://www.youtube.com/@LearningProgrammingTutorial*

6. *Van, N. T. (n.d.). Course Resources.*

7. Wikipedia. (n.d.).

8. *Mantis Documentation:* [*https://mantisbt.org/documentation.php*](https://mantisbt.org/documentation.php)

# TASK DISTRIBUTION

|  |  |  |
| --- | --- | --- |
| **Name** | **Tasks** | **Complete** |
| Nguyễn Hoàng Nhân | * 1.2 System requirements * 4.2 Test Plan * 6.1 Bug management system * Chapter 7 Conclusion * Whitebox Tesing:   + 5.1.5 Delete Many Product   + 5.1.6 Get All Type Product * Blackbox Testing: * Test cases and Bug reports from Use case from 2.2.3.10 to 2.2.3.13 | 100% |
| Lê Hoàng Lâm | * Acknowledgement * Chapter 3 Test Plan * 4.1 Testing Process * Whitebox Tesing:   + 5.1.1 Create Product   + 5.1.3 Get Details Product * Blackbox Testing: * Test cases and Bug reports from Use case from 2.2.3.5 to 2.2.3.9 | 100% |
| Trương Chí Kiên | * 2.1 Current status survey * 2.2.1 Use case diagram * 2.2.3 Use case scenario from 2.2.3.1 to 2.2.3.9 * Whitebox Tesing:   + 5.1.2 Update Product   + 5.1.8 Delete User * Blackbox Testing: * Test cases and Bug reports from Use case 2.2.3.1 to 2.2.3.4 | 100% |
| Võ Minh Hưng | * 1.1 System specifications * 2.2.2 Conceptual diagram * 2.2.3 Use case scenario from 2.2.3.10 to 2.2.3.18 * Whitebox Tesing:   + 5.1.4 Delete Product   + 5.1.7 Create User * Blackbox Testing: * Test cases and Bug reports from Use case 2.2.3.14 to 2.2.3.18 | 100% |