Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | In charge | Description |
| 24-October 25 | V1.0 | Trần Trung Nghĩa | Update Section I, II and Appendix A |
| 31-October 25 | V1.1 | Trần Trung Nghĩa | Overloading Method |
| 31-October 25 | V1.2 | Âu Hoàng Minh Duy | Update Section I, II, III |
| 7-November 25 | V1.3 | Trần Trung Nghĩa  Âu Hoàng Minh Duy | Update Section III & IV |
| 10-November 25 | V1.4 | Âu Hoàng Minh Duy | Update Section V |
| 14-November 25 | V1.5 | Trần Trung Nghĩa | Update Section VI, VII |
| 21-November 25 | V1.6 | Âu Hoàng Minh Duy | Update Section VIII & Report |

Abbreviation

DMS Database Management System

… …

List of Figures

Figure 1: Use-Case Diagram………………………………………………………………7

Figure 2: GUI…………………………………………………………………………24-25

List of Tables

Table 1: User Role………………………………………………………………………6-7

Table 2: List of Objects……………………………………………………………………8

Table 3: List of Classes………………………………………………………………....8-9

Table 4: Details of Classes…………………………………………………………...10-13

Table 5: Data Access Control Table………………………………………………..……17

Table 6: Method Access Control Table………………………………………………17-18

Table 7: Databases……………………………………………………………………….23

Table of Contents

I. Introduction……………………………………………………………………………..6

II. Class Analysis………………………………………………………………………….8

III. Class Design………………………………………………………………………….10

IV. Packet Design………………………………………………………………………..16

V. Access Control………………………………………………………………………..17

VI. Encapsulation vs Inheritance vs Polymorphism……………………………………..19

VII. Experiment………………………………………………………………………….22

VIII. Conclusion…………………………………………………………………………26

IX. Duty Roster…………………………………………………………………………..28

X. References…………………………………………………………………………….29

1. **INTRODUCTION**

*Managing in general is a stressful task, which requires precision, carefulness and constant awareness of the things you are in charge of. With difficult assignment like it, there comes a need for the right tool to get things done perfectly. That is the reason why we are building up a* ***Pet Shop Management System*** *to give a helping hand to those people who are taking good care of our future furry (or non-furry) family members.*

*The System will provide users with the capabilities to:*

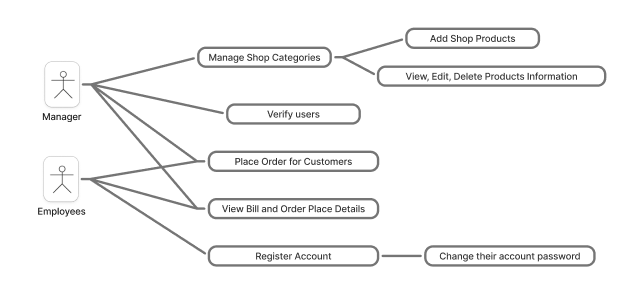
* *Manager*
  + *Mange Shop Categories*
  + *Add Shop Products*
  + *View, Edit, Delete Products information*
  + *Verify Users*
  + *Place Order for customers*
  + *View Bill and Order Placed Details*
* *Employees*
  + *Register, Change Password of their accounts.*
  + *Place Order for customers*
  + *View Bill and Order Placed Details*

*User role Table:*

|  |  |  |
| --- | --- | --- |
| ***Capabilities*** | ***Manager Role*** | ***Employees Role*** |
| *Mange Shop Categories*  *(Create/Edit/Delete Categories)* | *✅* |  |
| *Add Shop Products* | *✅* |  |
| *View, Edit, Delete Products information* | *✅* |  |
| *Verify Users*  *(Approve/Decline Registration Status)* | *✅* |  |
| *Register Account (Sign Up)* | *✅* | *✅* |
| *Change Password* | *✅* | *✅* |
| *Place Order for Customers* | *✅* | *✅* |
| *View Bill and Order Place Details* | *✅* | *✅* |

*Table 1: User Role*

*Use-Case Diagram:*

**

*Figure 1: Use-Case Diagram*

1. **CLASS ANALYSIS**
2. *Objects*

|  |  |  |  |
| --- | --- | --- | --- |
| *No* | *Object Name* | *State* | *Behaviors* |
| *1* | *User user* | *{id, name, email, phone, password, status}* | *getId, setId, getName, setName, getEmail, setEmail,...*  *(getters and setters)* |
| *2* | *Category category* | *{id, name}* |
| *3* | *Product product* | *{id, name, category, price}* |
| *4* | *Bill bill* | *{id, name, phone, email, date, total, createdBy}* |

*Table 2. List of Objects*

1. *Classes*

|  |  |  |
| --- | --- | --- |
| *Name* | *Attributes* | *Functions* |
| *User* | *{id, name, email, phone, password, status}* | *getters and setters* |
| *Category* | *{id, name}* |
| *Product* | *{id, name, category, price}* |
| *Bill* | *{id, name, phone, email, date, total, createdBy}* |
| *UserDao* |  | *save()*  *login()*  *updatePassword()*  *getAllRecords()*  *changeStatus()* |
| *CategoryDao* |  | *getAllRecords()*  *delete()* |
| *ProductDao* |  | *getAllRecords()*  *save()*  *update()*  *delete()*  *getAllRecordsByCategory()*  *filterProductByName()*  *getProductByName()* |
| *BillDao* |  | *getId()*  *save()*  *getAllRecordsByInc()*  *getAllRecordsByDesc()* |
| *DBOperations* |  | *setDataOrDelete()*  *getData()* |
| *ConnectionProvider* |  | *getCon()* |

*Table 3. List of Classes*

1. **CLASS DESIGN**
2. *Classes*

|  |  |  |  |
| --- | --- | --- | --- |
| *No* | *Class* | *Instance Variable* | *Methods* |
| *1* | *User* | *private int id;*  *private String fullName;*  *private String email;*  *private String phone;*  *private String password;*  *private String status;* | *Getters and Setters* |
| *2* | *Category* | *private int id;*  *private String name;* |
| *3* | *Product* | *private int id;*  *private String name;*  *private String category;*  *private String price;* |
| *4* | *Bill* | *private int id;*  *private String name;*  *private String phone;*  *private String email;*  *private String date;*  *private String total;*  *private String createdBy;* |
| *5* | *UserDAO* |  | *public static void save(user user)*  *public static user login(String phone, String password)*  *public static void updatePassword(String phone, String newPassword)*  *public static ArrayList<user> getAllRecords (String phone)*  *public static void changeStatus(String phone, String status)* |
| *6* | *CategoryDAO* |  | *public static ArrayList<categories> getAllRecords ()*  *public static void save(categories categories)*  *public static void delete(String id)* |
| *7* | *ProductsDAO* |  | *public static void save(Product product)*  *public static ArrayList<Product> getAllRecords()*  *public static void update(Product product)*  *public static void delete(String id)*  *public static ArrayList<Product> getAllRecordsByCategory(String category)*  *public static ArrayList<Product> filterProductByName(String name, String category)*  *public static Product getProductByName (String name)* |
| *8* | *BillDAO* |  | *public static String getId()*  *public static void save(Bill bill)*  *public static ArrayList<Bill> getAllRecordsByInc(String date)*  *public static ArrayList<Bill> getAllRecordsByDesc(String date)* |
| *9* | *DBOperations* |  | *public static void setDataOrDelete(String Query, String msg)*  *public static ResultSet getData(String query)* |
| *10* | *ConnectionProvider* |  | *public static Connection getCon()* |
| *11* | *Signup* | *public String emailPattern*  *public String phonePattern* | *public signup()*  *public void clear()*  *public void validateFields()* |
| *12* | *ManageCategory* |  | *public manageCategory()*  *public void validateField()* |
| *13* | *Login* | *public String emailPattern*  *public String phonePattern* | *public login()*  *public void clear()*  *public void validateFields()* |
| *14* | *Home* | *public String phone* | *public home()*  *public home(String userPhone)* |
| *15* | *ForgotPassWord* | *public String phonePattern* | *public forgotPassword()*  *public void clear()*  *public void validateFields()* |
| *16* | *ViewEditDeleteProduct* |  | *public ViewEditDeleteProduct()*  *public void validateField()* |
| *17* | *ViewBillsOrderPlacedDetails* | *public String userPhone* | *public ViewBillsOrderPlacedDetails()*  *public ViewBillsOrderPlacedDetails(String userPhone)*  *public void tableDetails()* |
| *18* | *VerifyUsers* |  | *public VerifyUsers()*  *public void getAllRecords (String phone)* |
| *19* | *PlaceOrder* | *public int billId*  *public int grandTotal*  *public int productPrice*  *public int productTotal*  *public String emailPattern*  *public String phonePattern*  *public String userPhone* | *public PlaceOrder()*  *public PlaceOrder(String phone)*  *public void productNameByCategory(String category)*  *public void filterProductByName(String name, String category)*  *public void clearProductFields()*  *public void validateField()* |
| *20* | *AddNewProduct* |  | *public AddNewProduct()*  *public void validateField()* |

*Table 4. Details of Classes*

***Overloading method:***

* *In class Home*

*public home() {*

*initComponents();*

*}*

*public home(String userPhone) {*

*initComponents();*

*phone = userPhone;*

*if(!userPhone.equals("0799755429")) {*

*manageCategoriesButton.setVisible(false);*

*newProductsButton.setVisible(false);*

*viewEditDeleteProductsButton.setVisible(false);*

*verifyUsersButton.setVisible(false);*

*}*

*}*

* *In class ViewBillsOrderPlacedDetails*

*public ViewBillsOrderPlacedDetails() {*

*initComponents();*

*}*

*public ViewBillsOrderPlacedDetails(String userPhone) {*

*initComponents();*

*this.userPhone = userPhone;*

*SimpleDateFormat dFormat = new SimpleDateFormat("dd-MM-yyyy");*

*Date date = new Date();*

*String todayDate = dFormat.format(date);*

*jTextField1.setText(todayDate);*

*}*

* *In class PlaceOrder*

*public PlaceOrder() {*

*initComponents();*

*}*

*public PlaceOrder(String phone) {*

*initComponents();*

*fieldProductName.setEditable(false);*

*fieldPrice.setEditable(false);*

*fieldTotal.setEditable(false);*

*buttonCart.setEnabled(false);*

*buttonGenerateBillPrint.setEnabled(false);*

*JFormattedTextField tf = ((JSpinner.DefaultEditor) jSpinner1.getEditor()).getTextField();*

*tf.setEditable(false);*

*userPhone = phone;*

*}*

1. **Package Design**

*com.nghiaduy.petshop* ***→ GUI / application layer***

*Login.java*

*Signup.java*

*ForgotPassword.java*

*Home.java*

*ManageCategory.java*

*ViewEditDeleteProduct.java*

*ViewBillsOrderPlacedDetails.java*

*VerifyUsers.java*

*PlaceOrder.java*

*AddNewProduct.java*

*com.nghiaduy.petshop.dao* ***→ data access layer***

*BillDAO.java*

*ProductDAO.java*

*CategoryDAO.java*

*UserDAO.java*

*DBOperations.java*

*ConnectionProvider.java*

*com.nghiaduy.petshop.model* ***→ model layer***

*Bill.java*

*Product.java*

*Category.java*

*User.java*

*common* ***→ shared utility***

*OpenPdf.java*

*src/main/resources* **→ *images***

1. **Access Control**

Analyse and discuss on the access control relating variables (data), method, classes and packages, interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Data | Class | Modifier | Description |
| 1 | private int id;  private String fullName;  private String email;  private String phone;  private String password;  private String status;  ... | Model Classes (User, Category, Product, Bill) | Public | * Attributes of Model Classes can only be accessed via **Getter or Setter** methods |
| 2 |  | DAO Classes (UserDAO, CategoryDAO, ProductDAO, BillDAO) | Public |  |

*Table 5: Data Access Control Table*

|  |  |  |  |
| --- | --- | --- | --- |
| No | Class | Methods | DESC |
| 1 |  |  |  |
| 2 |  |  |  |
| … |  |  |  |

*Table 6: Method Access Control Table*

1. **Encapsulation vs Inheritance vs Polymorphism**
2. **Encapsulation**

* In Model Classes: **Fields are private. Access only through public getters/setters.**

Public class user {

private int id;

private String fullName;

private String email;

private String phone;

private String password;

private String status;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

…

}

* In DAO Classes: (Example BillDAO)Methods are public static, and all DB details (ResultSet, SQL strings) stay inside the class. Other code just calls BillDAO.getAllRecordsByInc(date) and doesn’t see the DB internals.

Public static ArrayList<Bill> getAllRecordsByInc(String date) {

ArrayList<Bill> arrayList = new ArrayList<>();

try {

ResultSet rs = DBOperations.getData(“select \* from bills where date like ‘%” + date + “%’”);

while (rs.next()) {

Bill bill = new Bill();

bill.setId(rs.getInt(“id”));

bill.setName(rs.getString(“name”));

bill.setPhone(rs.getString(“phone”));

bill.setEmail(rs.getString(“email”));

bill.setDate(rs.getString(“date”));

bill.setTotal(rs.getString(“total”));

bill.setCreatedBy(rs.getString(“createdBy”));

arrayList.add(bill);

}

} catch (Exception e) {

JoptionPane.showMessageDialog(null, e);

}

return arrayList;

}

Other classes just call BillDAO.getAllRecordsByInc(date) and receive an ArrayList<Bill>; they never see ResultSet, SQL, or DBOperations details

1. **Polymorphism**

*In Home Class:*

*public home() {*

*initComponents();*

*}*

*public home(String userPhone) {*

*initComponents();*

*phone = userPhone;*

*if(!userPhone.equals(“0799755429”)) {*

*manageCategoriesButton.setVisible(false);*

*newProductsButton.setVisible(false);*

*viewEditDeleteProductsButton.setVisible(false);*

*verifyUsersButton.setVisible(false);*

*}*

*}*

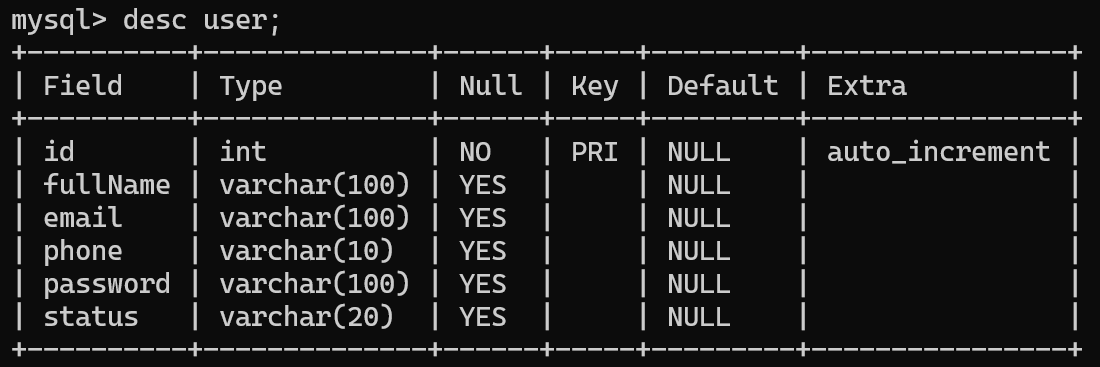
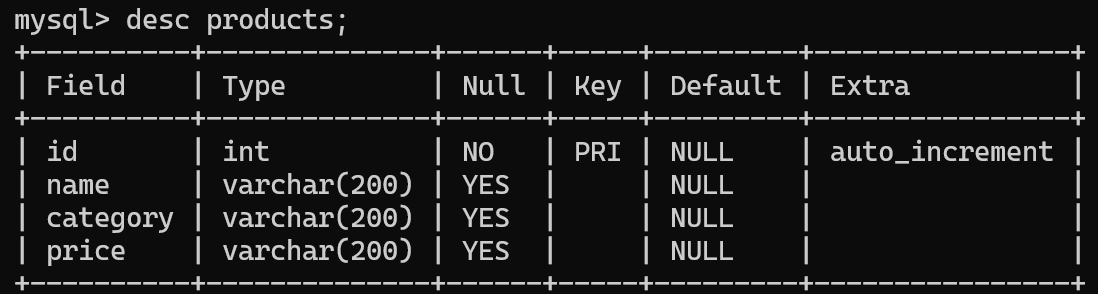
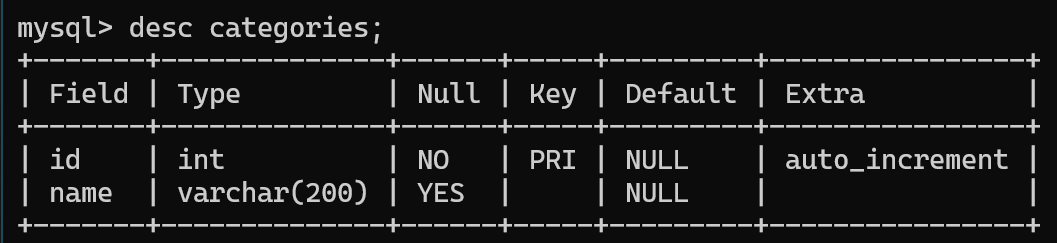
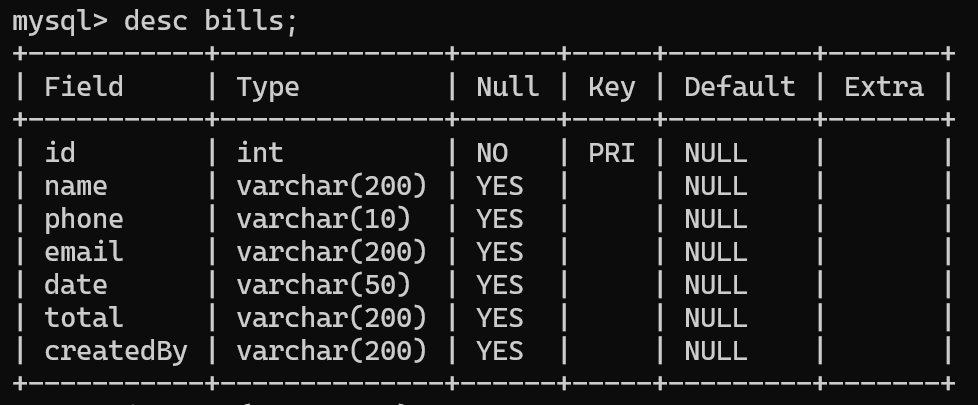
*public home(String userPhone)* ***overrides*** *public home()*

1. **Experiment**
2. **Environment and Tools**
3. **Environment:**
   * 2 PCs
4. **Tools**:
   * NetBeans IDE
   * Github
   * Java 25
   * itextpdf.jar ***→ Printing PDF***
   * MySQL Database and mysqlconnector.jar
5. **Project functions**

* *Mange Shop Categories*
* *Add Shop Products*
* *View, Edit, Delete Products information*
* *Verify Users*
* *Place Order for customers*
* *View Bill and Order Placed Details*
* *Register, Change Password of their accounts.*
* *Place Order for customers*
* *View Bill and Order Placed Details*

1. **Database (min. 4 tables)**

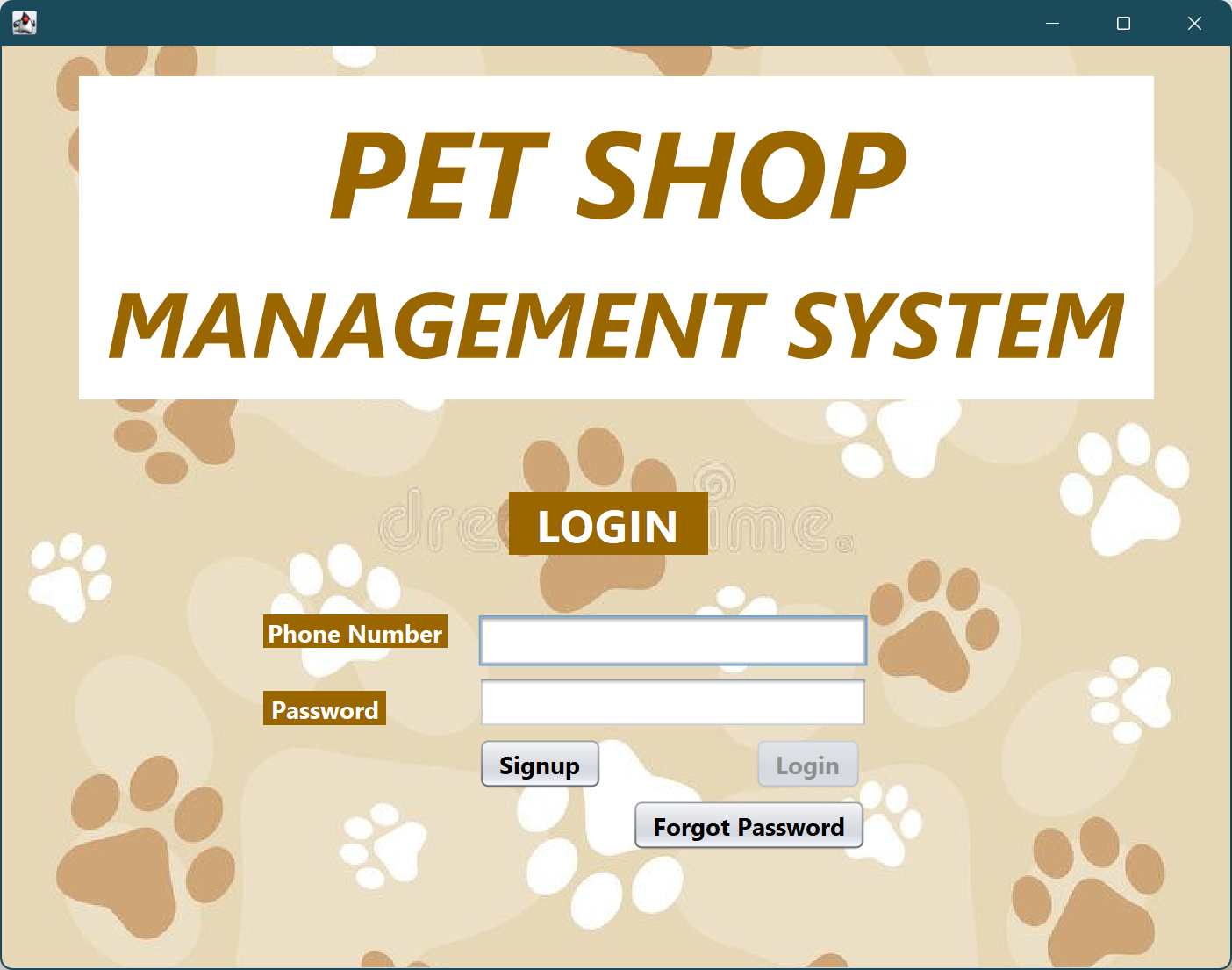
* Database: petshop
* Tables:



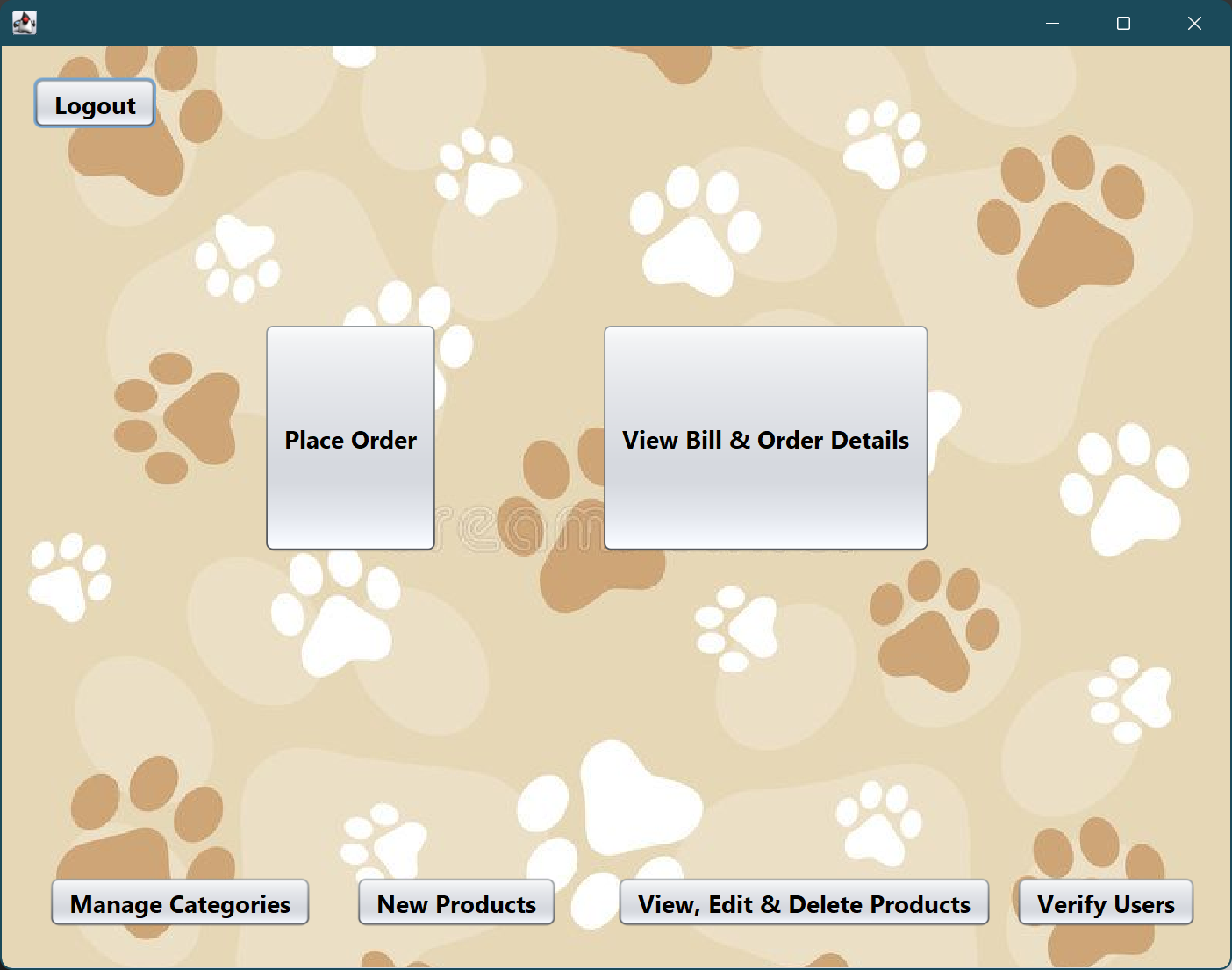
*Table 7: Databases*

1. **GUI**

(Login Button takes you to **Home Page**)



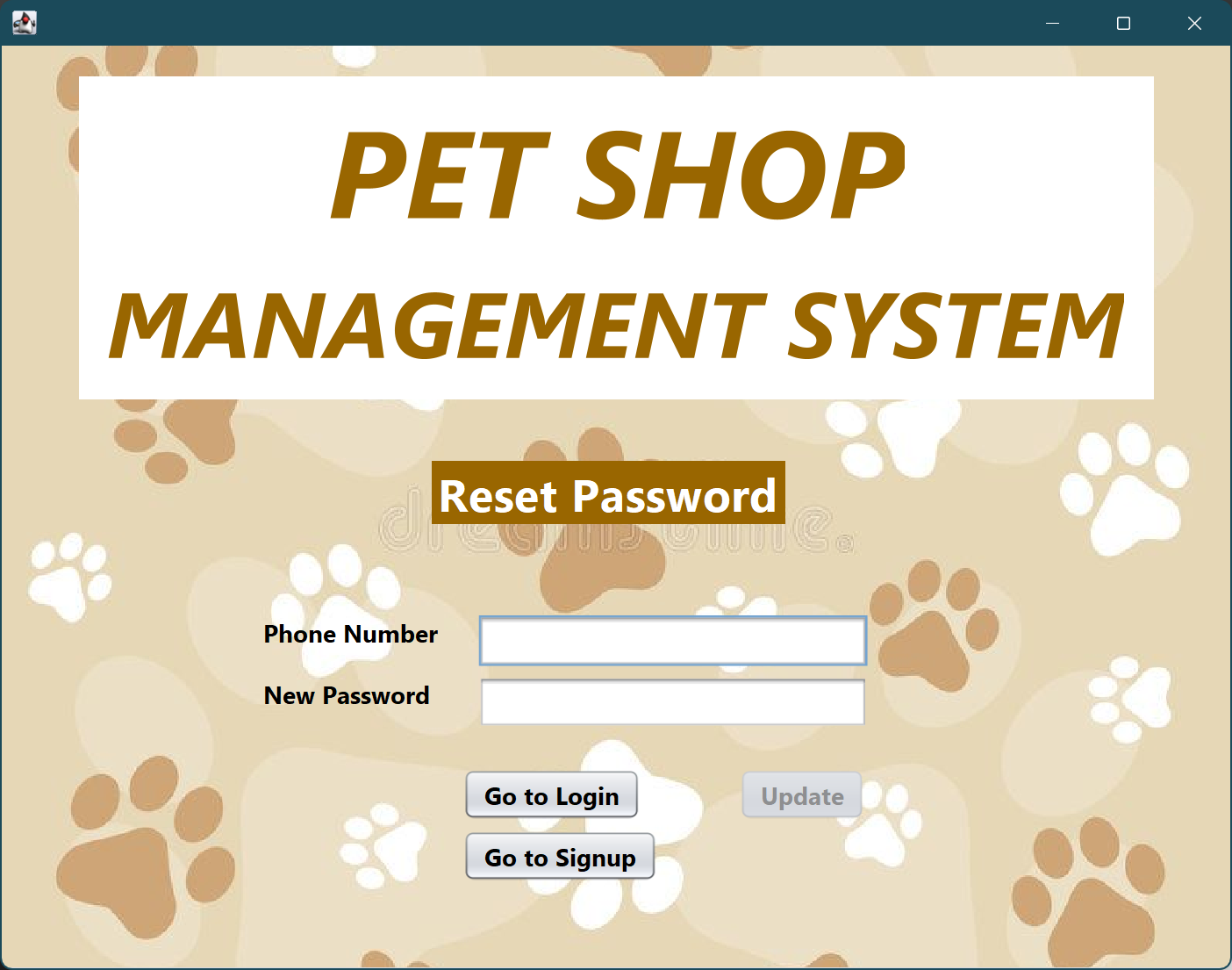
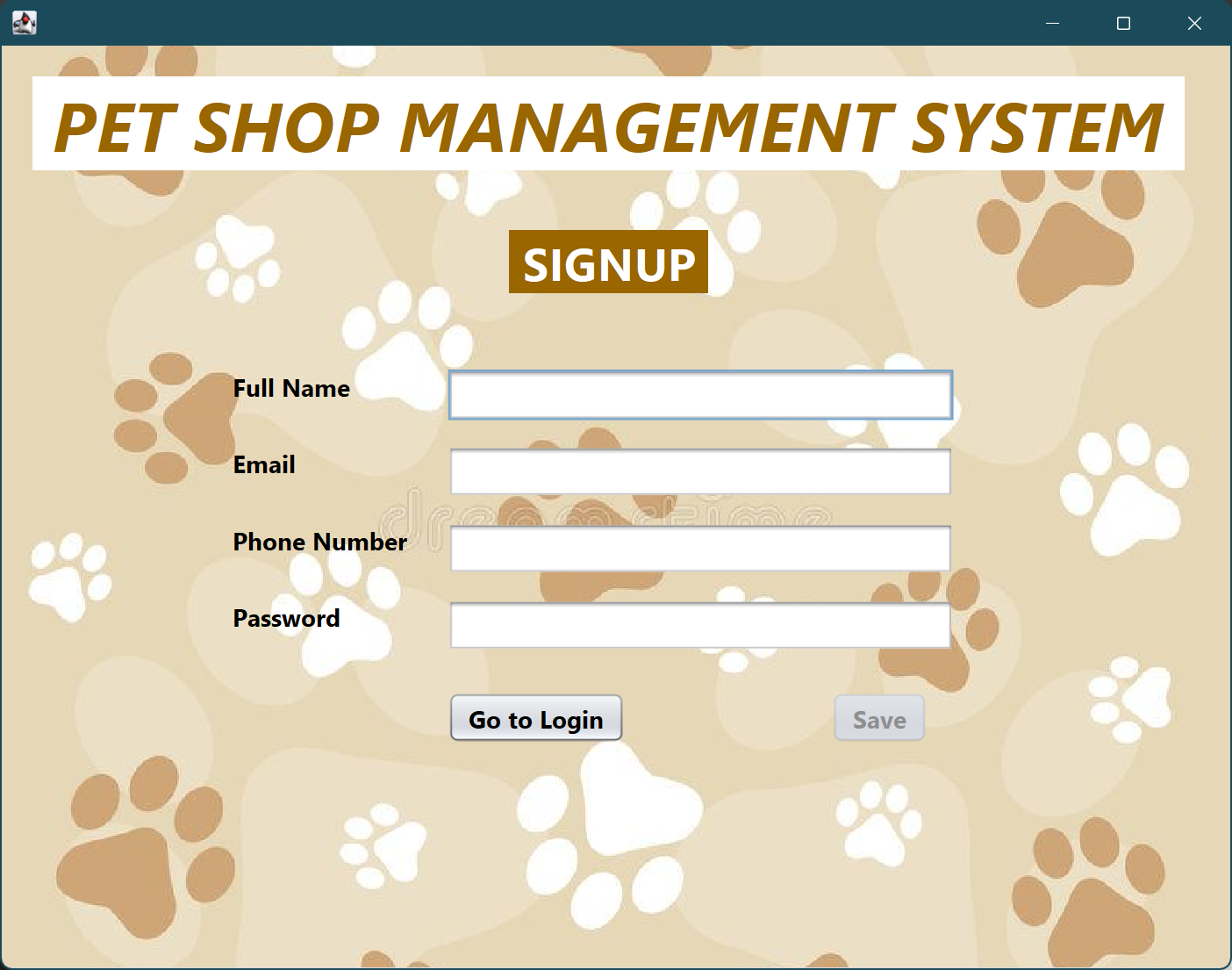
(**Home page** for **admin** users)



(**Home page** for **non-admin** users)



While **Sign Up** and **Forgot Password** take you to there respective pages.



*Figure 2: GUI*

1. **Conclusion**

**1. Project Assessment (Pros & Cons)**

**Pros:**

* **Clearly Role Segregation:** The system effectively implements Access Control, between the ‘Manager’ and ‘Employee’ roles. Which ensures that’s the functions like managing categories and verifying new users are assigned to the Manager. Leads to better protection and accountability.
* **Effective Use of the OOP**: The design in this project is exhibited strong OOP. With the use of Encapsulation in model classes and DAO classes ensure integrity and modularity.
* **Comprehensive Data Layer:** The DAO layer allows to separates my Pet Shop Management system logic from data persistence, which is easier for maintenance and scalability. Its also interacts with MySQL database structures with tables like user, categories, etc.
* **Essential Functionality:** We made sure all the core requirements have been implemented such as the product, category management, bill & order from the customers.

**Cons:**

* **Limited Future-Proofing:** Well, the current logic is hide the menu items away from the non-admin users which hardcoded to a specific phone number. Which means this is rigid and poor practice for the real-world deployment.
* **GUI is basic:** As our GUI screenshot suggests a functional but it just a simple interface. A modern one should be put into consideration next project.
* **Lack of Frontend Validation Detail:** the method **validateField()** is mentioned in classes like login, it does not explicitly confirm robust front-end user experience like send out error messages or guidance for the user.

**2. Project Score and Justification**

We believe that our project can achieve the **score of 9.5/10** because the project is functionally complete, we demonstrate a solid understanding of layered architecture which includes GUI, DAO, Model and show the application of fundamental OOP concepts that’s are Encapsulation and Polymorphism. The comprehensive database design and implementation of critical features like bill generation and access control prove its real-world viability.

The downside of our project is due to the hardcoded admin identification logic and potential for improved UI as mentioned above.

1. **Future Improvements:**

* Implement new DAO methods and dedicated GUI screen for example: allow managers to view report of sales over time or track inventory levels.
* Maybe introducing the automatic inventory tracking such as when order is placed, the stock will automatically decrease or else if the stock level of the product fell below the minimum threshold, there will be an alarm.

**DUTY ROSTER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Task** | **In Charge** | **Start** | **End** | **State** | **Note** |
| 1 | Update Section I, II and Appendix A | Trần Trung Nghĩa | 24-October 25 | 28-October 25 | Done |  |
| 2 | Overloading Method | Trần Trung Nghĩa | 31-October 25 | 3-November 25 | Done |  |
| 3 | Update Section I, II, III | Âu Hoàng Minh Duy | 31-October 25 | 3-November 25 | Done |  |
| 4 | Update Section III & IV | Trần Trung Nghĩa  Âu Hoàng Minh Duy | 7-November 25 | 9-November 25 | Done |  |
| 5 | Update Section V | Âu Hoàng Minh Duy | 10-November 25 | 13-November 25 | Done |  |
| 6 | Update Section VI, VII | Trần Trung Nghĩa | 14-November 25 | 17-November 25 | Done |  |
| 7 | Update Section VIII & Report | Âu Hoàng Minh Duy | 21-November 25 | 28-November 25 | Done |  |

**REFERENCE**

1. Oracle, Official Documentation, https://docs.oracle.com/javase/tutorial/
2. Oracle, Official Documentation, https://docs.oracle.com/javase/tutorial/uiswing/
3. MySQL Official Documentation, https://dev.mysql.com/doc/refman/8.0/en/
4. Eckel, Bruce, Online
5. iText, https://itextpdf.com/
6. Oracle, https://docs.oracle.com/javase/tutorial/jdbc/

APPENDIX A: CLASS DESCRIPTION

Class 1 : Grade…. (Source: Src/Grade.java)