

Table of Contents

Abstract.....	4
1.0 Introduction -----	5
1.1 Existing Systems -----	5
1.2 Proposed System -----	5
1.2.1 Advantage of the proposed system -----	5
1.3 Software Engineering Model -----	5
1.4 Purpose (1 statement) -----	6
1.5 Project Objective/Goals and Scope -----	6
1.6 Technologies & Tools -----	6 1.7
Users (Who can use this system?) -----	6
1.8 Motivation -----	6
1.9 Timeline -----	7
2.0 Analysis of the System-----	7
2.1 Activity List -----	7
2.2 ContextDiagram -----	8 2.3 Data Flow
Diagram -----	9 3.0 Database Design
-----	9 3.1 Entity
Relationship Diagram -----	10
3.2 Table Schema Diagram -----	10
4.0 Functionality and Implementation -----	11
4.1 Features -----	11
5.0 Earned Value Analysis -----	27
6.0 Testing -----	27 7.0
Conclusion -----	28 7.1
Limitations -----	28 7.2 Future
Enhancements/Recommendations -----	28 7.3 Team
Members -----	28
7.3.1 Learning Experience and Outcome -----	29
8.0 Reference -----	31

Table of Figures

Figure 1: Spiral Software Development Model -----	5
Figure 2: Home Care Activity List -----	7
Figure 3: Context Diagram -----	8
Figure 4: Data Flow Diagram -----	9
Figure 5: Entity Relationship Diagram -----	10
Figure 6: Table Schema -----	10
Figure 7: Login Screen -----	11
Figure 8: Login Screen Code -----	11
Figure 9: Main Menu Screen -----	12
Figure 10: Meal Prep Screen -----	13
Figure 11: Meal Prep Screen Code 1 -----	14
Figure 12: Meal Prep Screen Code 2 -----	15
Figure 13: Medical Portal Screen -----	16
Figure 14: Medicine Delivery Screen -----	17
Figure 15: Payment Screen -----	18
Figure 16: Service History Screen -----	19
Figure 17: Medical History Screen -----	19
Figure 18: Medical History Screen Code -----	20
Figure 19: Update Medical History Screen -----	21
Figure 20: Update Medical History Screen Code 1 -----	22
Figure 21: Update Medical History Screen Code 2 -----	23
Figure 22: Service Provider Login Screen -----	24
Figure 23: Admin Main Menu -----	25
Figure 24: Admin Order Review Screen -----	25
Figure 25: Admin Order Review Screen Code -----	2

Abstract

This project was designed and developed to give health patient users an application where they will have meals and medicine delivered to their house based on their needs and specifications. They can also access and update their medical records through our virtual Medical Portal which has their information available to view and update. The application is intended for users of all ages but mainly for people who have a tough time getting around from place to place. Older people and people with a medical condition can most benefit from utilizing this application. The app was implemented in Netbeans Java with a MySQL database storing the information from the GUI java interface.

Keywords- Healthcare, Java, MySQL, Diet, Medicine

1. Introduction

1.1. Existing Systems

- ITriage- allows patients to find on their health conditions, gives them step by step guidance, and allows them to review previous claims and store health information
- CareAware Connect- complete workflows and manage clinical communications.
- Ambulatory EHR- providers can access complete web charts which gives them instant access to patient records across healthcare organizations
- MyChart Mobile- patients can access health data from previous in office doctors visits.

1.2. Proposed System

Home Care Activities

A Healthcare Portal for Home Care Activities(HCA) who provide services such as a virtual medical portal where they can keep track of their medical history,order meals and have them delivered to home, and provide easier access to necessary medication. These services will be marketed to the elderly and homebound individuals who find it difficult to obtain these significant services. This application will be designed for the use of both the service providers and the consumers. For this application we will be using Java as the coding language, and utilize the Jframes class to create an interactive interface.

1.2.1 Advantage of Proposed System

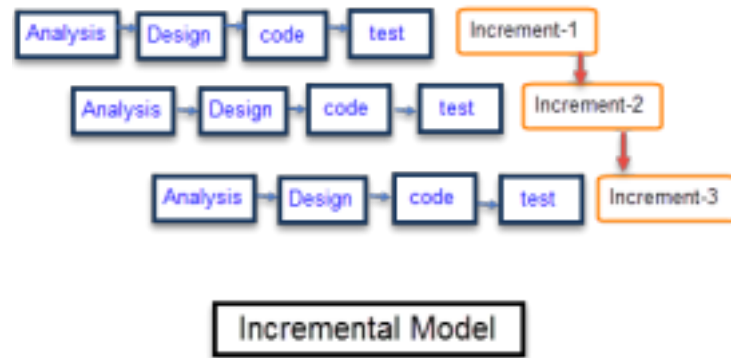
Advantages of our application for patient/users include the following:

- Easy access to Medical records
- Ability to order meal plans in just a click of a button

- Ability to make a payment directly on the application
- Delivery of medicine right to your front door

1.3. Software Engineering Model

Iterative Incremental Model
The product is designed, implemented and tested incrementally until the product is finished. The iterative incremental model divides the product into small chunks.



5

Advantages of Iterative and Incremental Model-

- Since the product is being divided into small chunks, you can easily manage the product.
- Defects are found at an early stage.
- Risk is analyzed and identified in iterations.
- Easy to make any change in the requirement and there would be no cost because you can incorporate the new requirement in the next iteration.

1.4. Purpose

To create a user friendly Healthcare Portal that gives patients the access to order meals and medicine as well as to access and update medical records.

1.5. Project Objectives/Goals and Scope

The objective of this project was to create a Healthcare portal that would use an interface friendly to the user. One of our main goals was to make the application simple and easy to use so that our intended audience of elderly and disabled would not have a tough time accessing it. The application will be available on mobile for all Android and iPhone users.

1.6. Technologies & Tools

- Netbeans Java, create Interface
- MySQL Database, store Data
- Zoom Video Conferencing, communicate amongst Team Members

- Google Drive, store project materials

1.7. Users

The users of this application can be anybody. Since we do not have any affiliation with any hospitals or doctors, anybody would be able to use our application. The target audience includes elderly and disabled but does not imply that all other users can't use it. Younger individuals or middle aged individuals can also use the application to store their medical records, order medicine, or meal plan for themselves.

1.8 Motivation

Our motivation for creating this app is to apply information technology to health and healthcare. We wanted to be able to use technology to help people with medical needs and using this app we are able to achieve that goal.

1.9 Timeline

6

The timeline for our project is as follows:

1. Proposal- 4/6
2. Deliverable(analysis)- 4/8
3. Diagram(design)- 4/8
4. Code start- 4/10
5. Code finish- 4/25
6. Testing- 4/25 to 4/26
7. Refinements/Corrections- 4/29
8. Testing(including live testing)- 4/30
9. Extra days in case of delays in project- 5/1 to 5/3
10. Project due- 5/4 or 5/6

2. Analysis of System

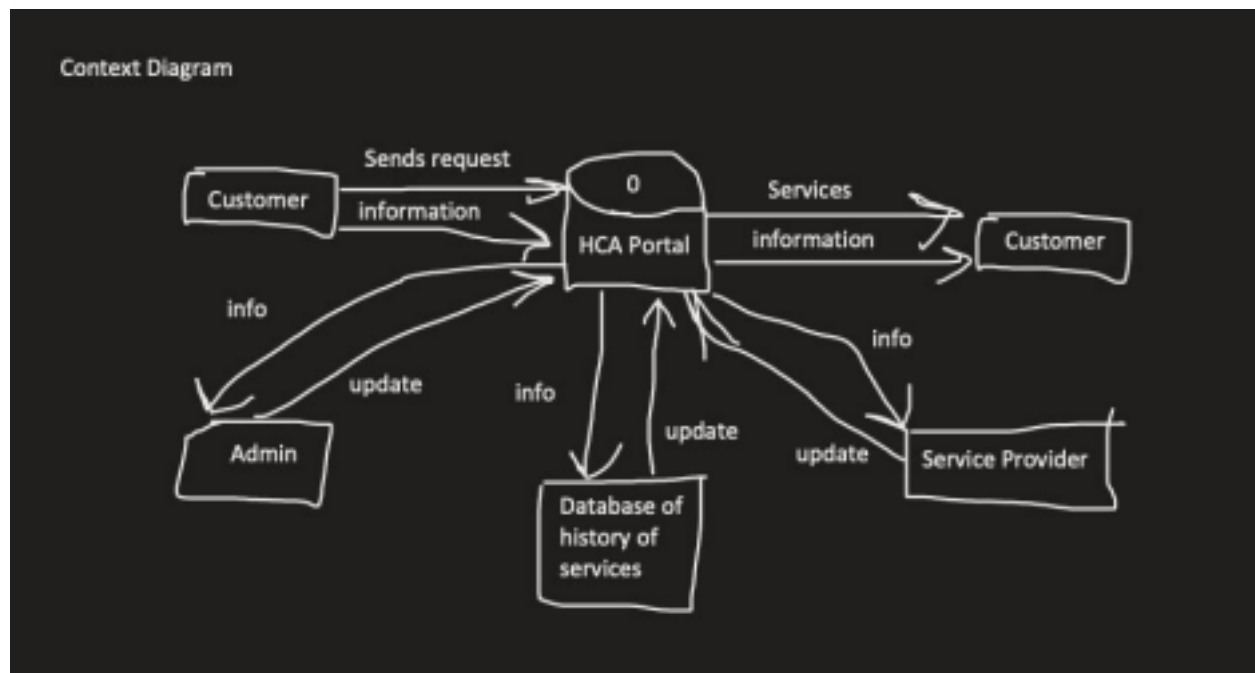
2.1 Activity List

Home Care Activity List

1. Customer needs a service
2. Customer logs into the HCA portal
3. Customer requests a service from the portal
 - a. Services: Meal prep, Medicine, Food delivery
4. Portal sends the requested services to the customer
5. All services provided are recorded and saved in a database
6. Customers can access a history of the services provided through the portal if they want

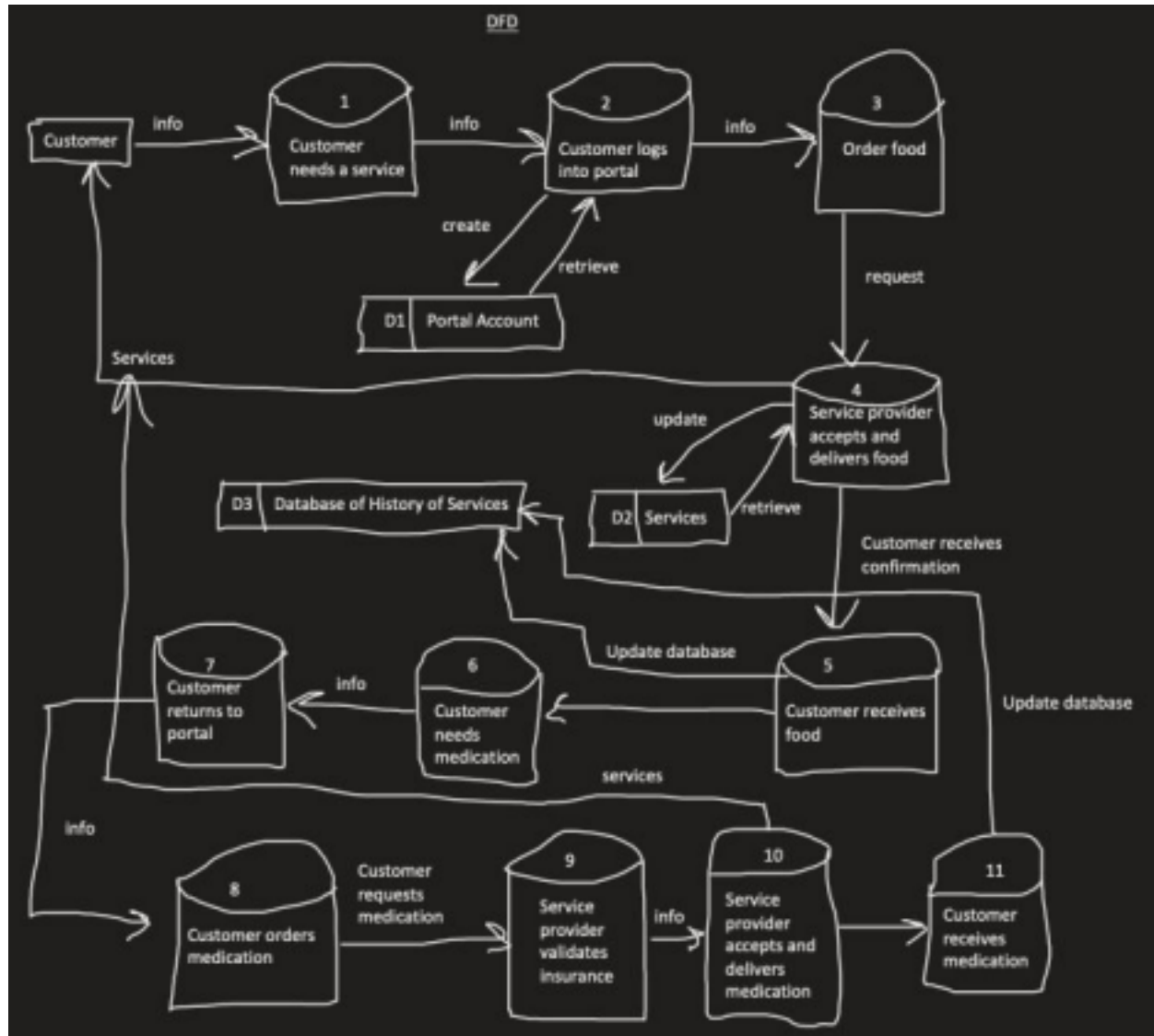
7

2.2. Context Diagram



8

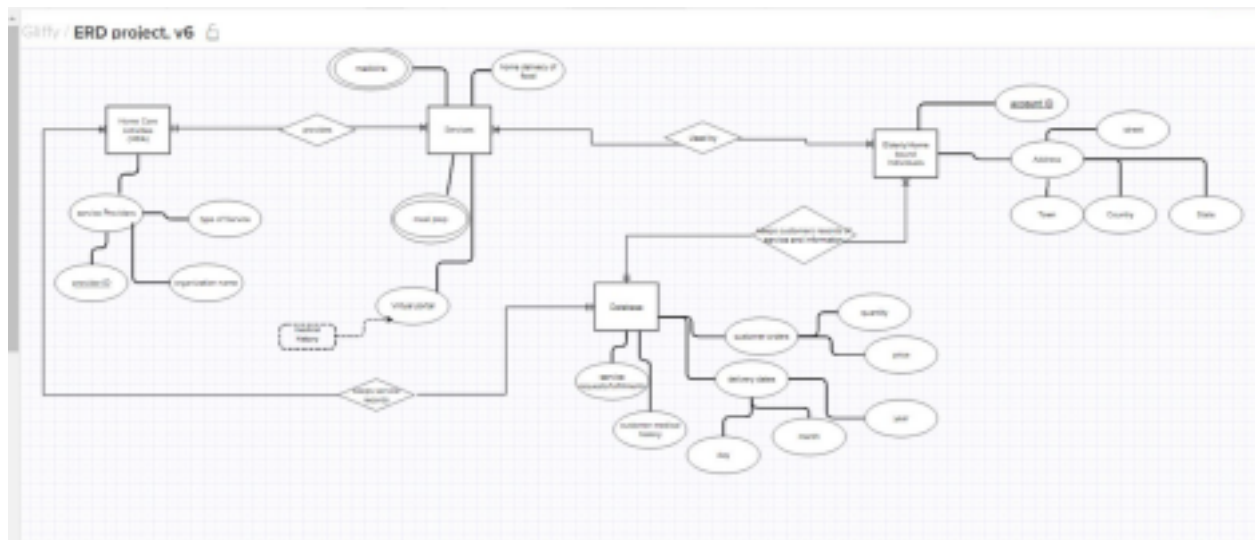
2.3 Data Flow Diagram



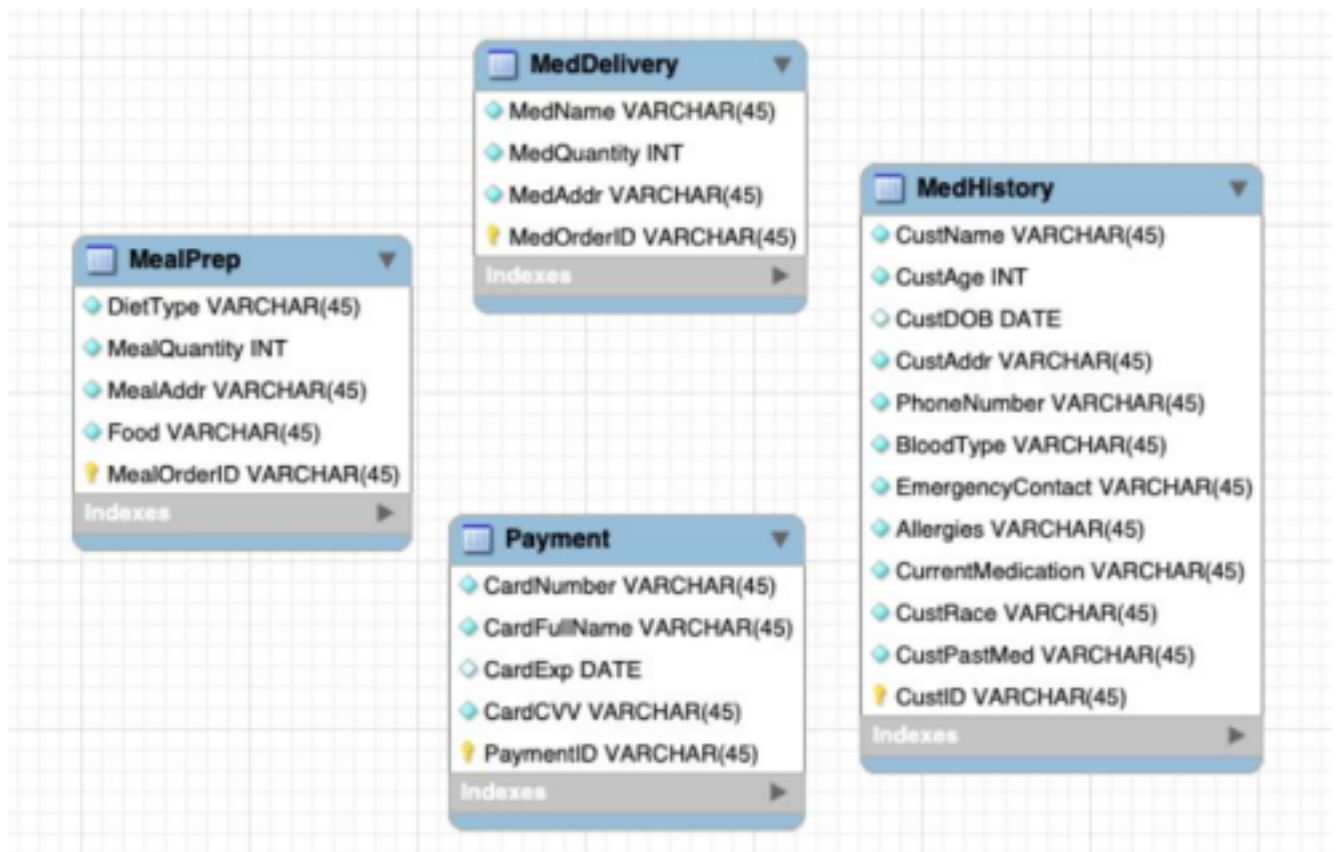
3. Database Design

The project was designed using gliffy. We selected gliffy because it was able to offer us the necessary tools to create the flowchart that we needed. It helped us in this project because it allows us to freely manipulate the objects and edit the information. This ERD was critical in helping us understand this project and the relations between all of the parts.

3.1. Entity Relationship Diagram



3.2 Table Schema



10

4. Functionality and Implementation

4.1. Features

This section features screenshots of all the JFrames in the project as well as a snippet of code, followed by a brief explanation.

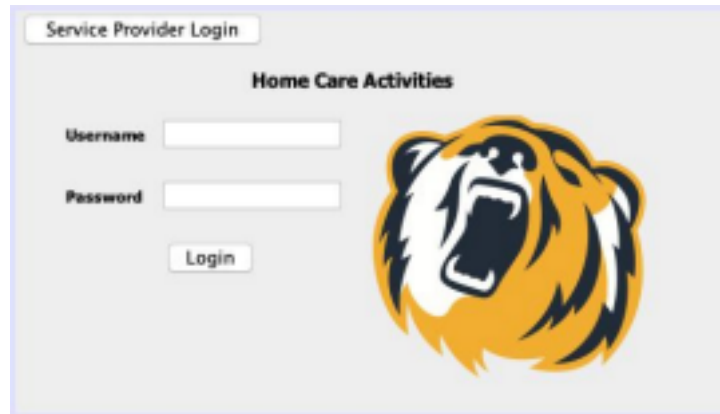


Figure 7: Login Screen

The figure above shows the login screen labeled “Home Care Activities” with the NYIT bear logo to the right of the screen. It has a username entry, password entry, login button and an option to navigate to the service provider login or administrator login.

```
private void jbtnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    String username = txtUserName.getText();
    String password = txtPassword.getText();

    if (password.equals("123") && (username.equalsIgnoreCase("John")))
    {
        txtUserName.setText(null);
        PasswordLabel.setText(null);

        dispose();
        MainMenu Info = new MainMenu();
        Info.setVisible(true);
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Invalid Username or Password");
    }
}

private void ServiceProviderLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    dispose();
    ServiceProviderLogin Info = new ServiceProviderLogin();
    Info.setVisible(true);
}
```

Figure 8: Login Screen Code

11

The above code shows that the username and password must match the specific characters in order to log in successfully. The username isn't case sensitive, but the password is, so it has to match exactly.

The button labeled “ServiceProviderLogin” closes the current window and switches to the admin login when you click it.

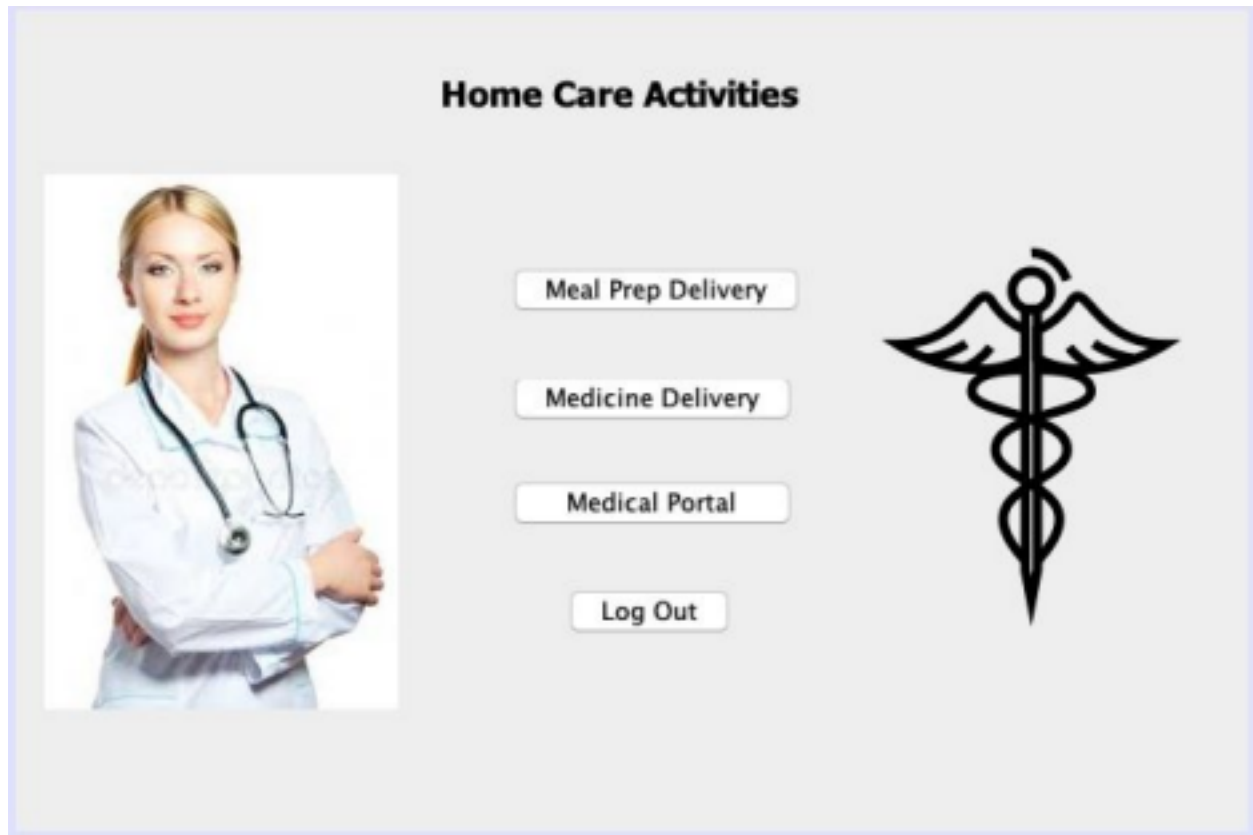
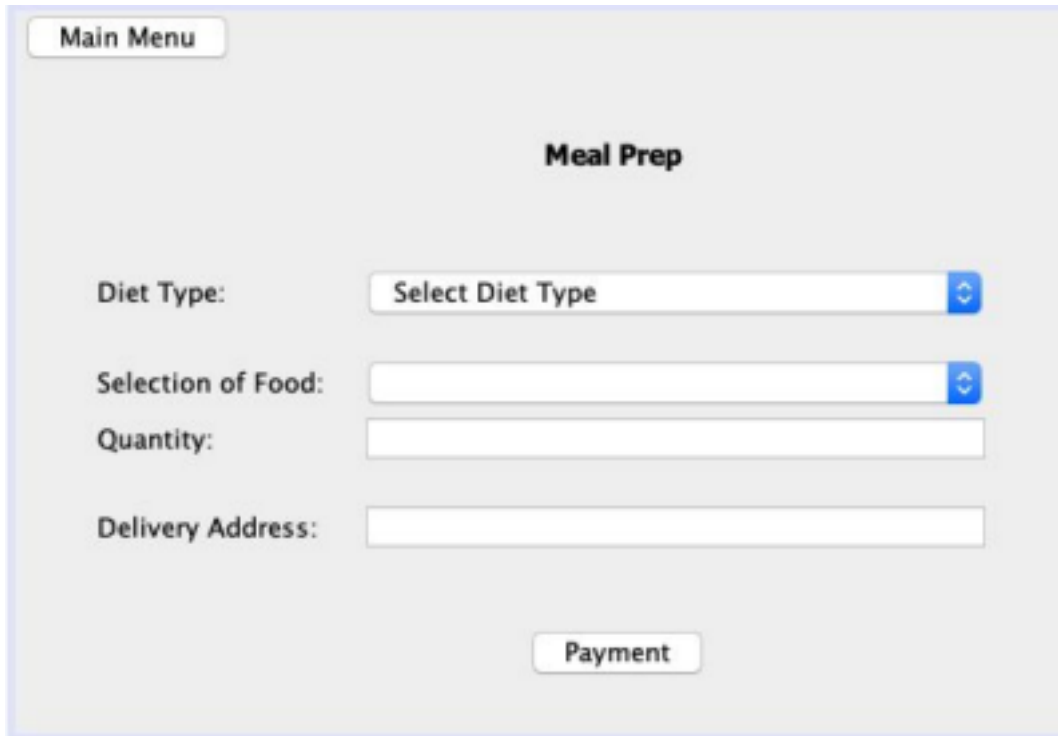


Figure 9: Main Menu Screen

The above figure shows the main menu after logging from the regular login screen. There are three buttons each leading to a different function of the program and one button to log back out. The code for this screen is fairly simple. Each button uses the “dispose();” function to close the current window and then open a new screen respective to the button you click.

The image shows a web application interface for a meal prep service. At the top left, there is a button labeled "Main Menu". The main heading is "Meal Prep". Below this, there are four input fields: "Diet Type:" with a dropdown menu showing "Select Diet Type", "Selection of Food:" with a dropdown menu, "Quantity:" with a text input field, and "Delivery Address:" with a text input field. At the bottom center, there is a button labeled "Payment".

Main Menu

Meal Prep

Diet Type:

Selection of Food:

Quantity:

Delivery Address:

Payment

Figure 10: Meal Prep Screen

This screen allows you to enter information for a meal prep order for the following values: diet type, selection of food, quantity, delivery address. When you select a certain diet type in the first dropdown menu it will then change the next dropdown menu accordingly. Allowing you to choose a few options from each diet type. After selecting what you would like to order including delivery address it will give you a transaction ID, and go to payment info.

```

private void PaymentActionPerformed(java.awt.event.ActionEvent evt) {
try
{
    // Step 1: "Load" the JDBC driver
    Class.forName("com.mysql.cj.jdbc.Driver");

    // Step 2: Establish the connection to the database
    String url = "jdbc:mysql://localhost:3306/HCA?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=
    Connection conn = DriverManager.getConnection(url,"root","password1");

    String query= "Insert into MealPrep (DietType, MealQuantity, MealAddr, Food)values(?,?,?,?)";
    PreparedStatement pst = conn.prepareStatement(query);
    pst.setString(1, DietTypeSelection.getSelectedItemAt().toString());
    pst.setString(2, MealQuantity.getText());
    pst.setString(3, MealAddr.getText());
    pst.setString(4, FoodSelection.getSelectedItemAt().toString());
    pst.executeUpdate();

    String sql = "Select Max(MealOrderId) From MealPrep";
    PreparedStatement pdt = conn.prepareStatement(sql);
    ResultSet rs = pdt.executeQuery();

    Payment Info = new Payment();
    Info.setVisible(true);

    dispose();

    JOptionPane.showMessageDialog(null, "Order Placed Successfully! Your Order Id is: " + DbUtils.resultSetToNestedList(rs)
}
catch (Exception e)
{
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
    System.err.println("D'oh! Got an exception!");
    System.err.println(e.getMessage());
}
}

```

Figure 11: Meal Prep Screen Code 1

```

private void DietTypeSelectedItemStateChanged(java.awt.event.ItemEvent evt) {
    // TODO add your handling code here:

    ArrayList<String> array=new ArrayList<>();
    Iterator<String> iter;

    if(DietTypeSelection.getSelectedItem().equals("Vegan"))
    {
        FoodSelection.removeAllItems();
        array.add("Lentil Soup");
        array.add("Crispy Buffalo Cauliflower Bites");
        array.add("Black Bean and Sweet Potato Quesadillas");
        iter=array.iterator();
        while(iter.hasNext())
        {
            FoodSelection.addItem(iter.next());
        }
    }
    else if(DietTypeSelection.getSelectedItem().equals("Kosher"))
    {
        FoodSelection.removeAllItems();
        array.add("Matzoh Ball Soup");
        array.add("Cholent");
        array.add("Shakshuka");
        iter=array.iterator();
        while(iter.hasNext())
        {
            FoodSelection.addItem(iter.next());
        }
    }
    else if(DietTypeSelection.getSelectedItem().equals("Vegetarian"))
    {

```

Figure 12: Meal Prep Screen Code 2

The above codes for the Meal Prep JFrame show the sql query used to enter the data and a sample of the code for the dropdown menu. The sql query allows you to insert the data into the Meal Prep table in the database by taking the data from each fill-in form and dropdown menu. The dropdown menu is dynamic so based on what you select from the first dropdown menu in the diet type, it changes what you choose for the second dropdown menu for the actual foods. That way you have the second dropdown menu acting like a sub-menu as it gets more specific with the choices available.

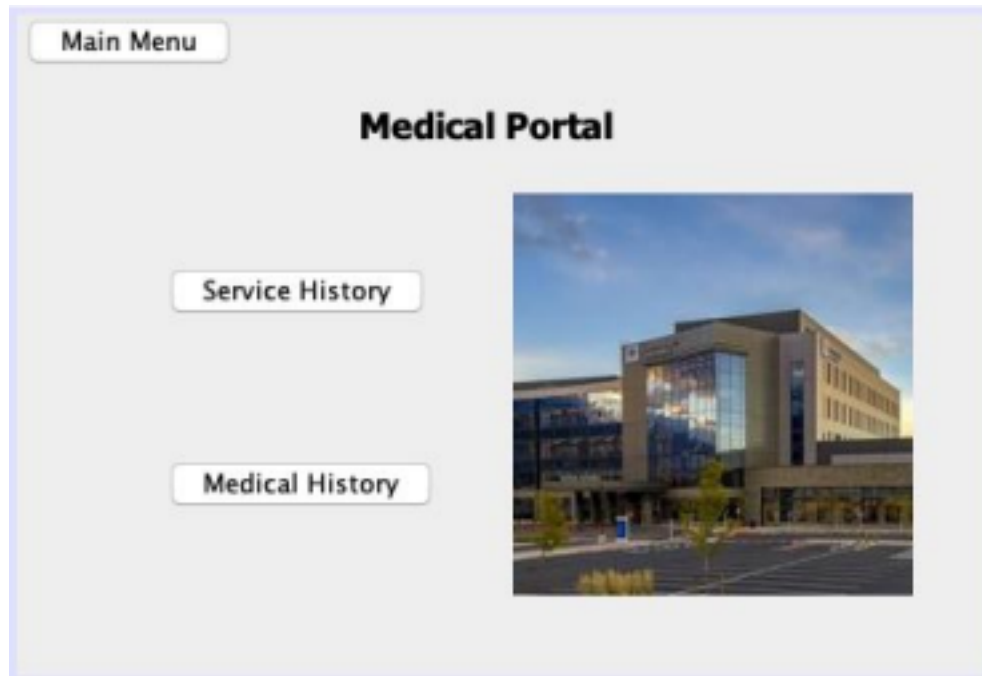


Figure 13: Medical Portal Screen

On this menu, it displays three options coded as JButtons. The first button on the top left is responsible for redirecting you back to the main menu screen. The next button is the service history button which allows the user to view past orders and payment information. The last button is the Medical history button which will show the user their medical history. Additionally, on this menu we added an image to improve the visual presentation of the menu.

The image shows a Java Swing window titled "Medicine Delivery". In the top-left corner, there is a button labeled "Main Menu". The window has a light gray background. The title "Medicine Delivery" is centered at the top. Below the title, there are three input fields. The first is labeled "Enter Medicine Name" and is a text box. To its right is a dropdown menu with the text "Select Medicine" and a blue arrow icon. The second input field is labeled "Enter Delivery Address:" and is a text box. The third input field is labeled "Enter Medicine Quantity:" and is a text box. At the bottom center of the window, there is a button labeled "Payment".

Figure 14: Medicine Delivery Screen

This menu allows the user to order medication by choosing from a drop down menu several available options for ordering. Furthermore, there are two additional text fields that allow the user to enter the quantity as well as the address to where the order will be delivered to. Lastly we have two JButtons Main Menu and Payment, where one takes you back to the main menu and the other one redirects you to the payment page.



Figure 15: Payment Screen

This screenshot displays our payment page of the application. The user must enter their card number, name, expiration date, and CVV into the adjacent text fields. Once filled out, the user may click the JButton Submit, and the payment information will be sent to the database and service history page. Additionally, if the information is not properly filled out, an error will result and prompt the user to make sure that they entered the information correctly. This payment page is used for both medicine and food delivery.



Figure 16: Service History Screen

The service history menu shows all past food orders, medicine orders, and payments which are connected to our database and are displayed using JTables. In addition, there are two JButtons used to go to the main menu of the application and the main menu of the medical portal. The user can access this page under the medical portal menu button for Service History.

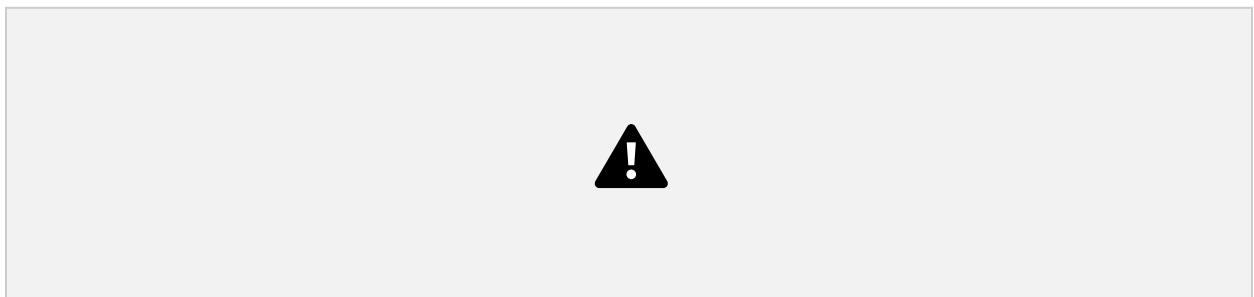


Figure 17: Medical History Screen

The user can access their medical history through a medical portal and click the JButton for Medical History. Here the user can find their medical history containing variables such as Name,

medical conditions, past medical conditions, and their unique customer ID. All this information is being displayed through a JTable which is connected to our database for storage. Additionally, there are two available JButtons that perform actions such as returning to the medical portal and redirecting the user back to the main menu of the application.

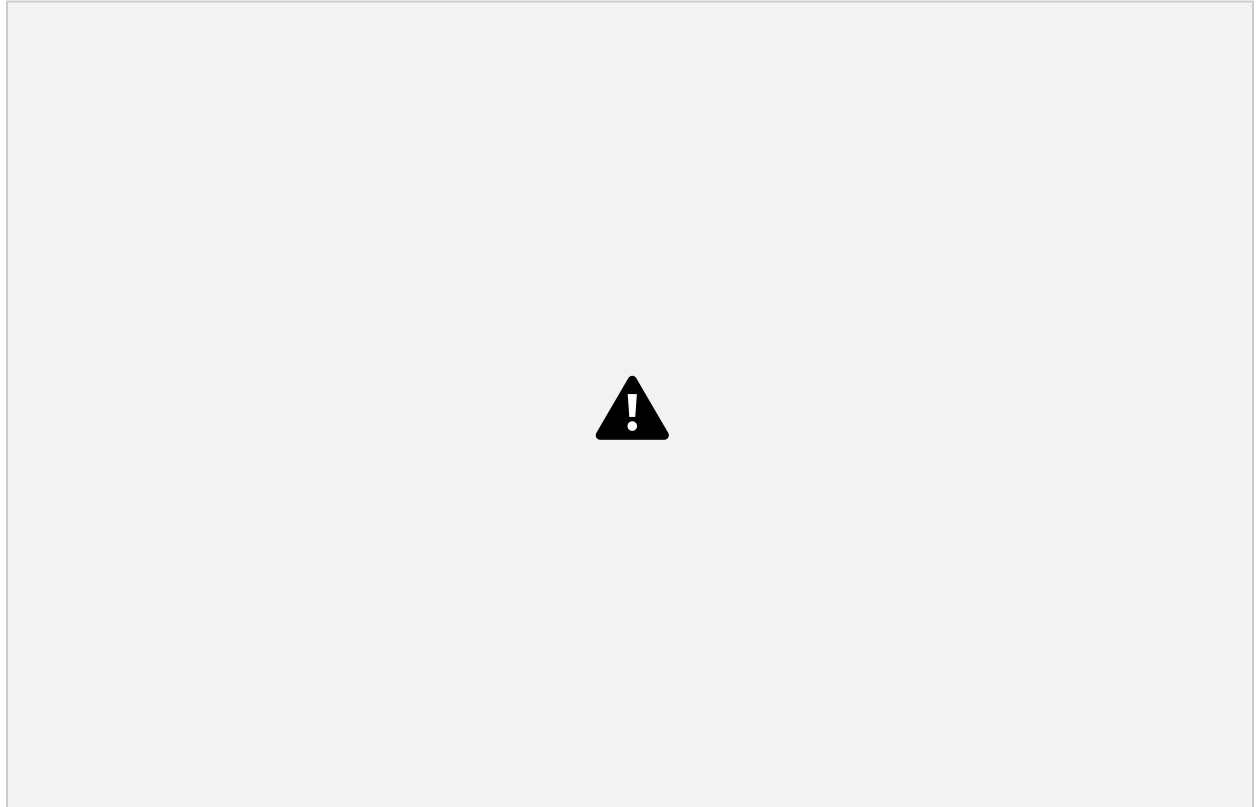


Figure 18: Medical History Screen Code

The above screenshot displays the code responsible for the Medical History page of our application. It uses the sql query “SELECT * FROM MedHistory” to display the correct table and utilizes the rs2xml.jar file to grab the information from the database. Table T3 in the above code applies the method “.setModel(DbUtils.resultSetToTableModel(rs));” in order to properly display the data.



Figure 19: Update Medical History Screen

This Screenshot displays the users medical history from the point of view of the Admin/service provider. Here the service provider can view and update the client's medical history and upon clicking the update JButton it will simultaneously update the clients info and display it to the JTable above. Lastly, there is a JButton called Back that returns the service provider to the previous screen.

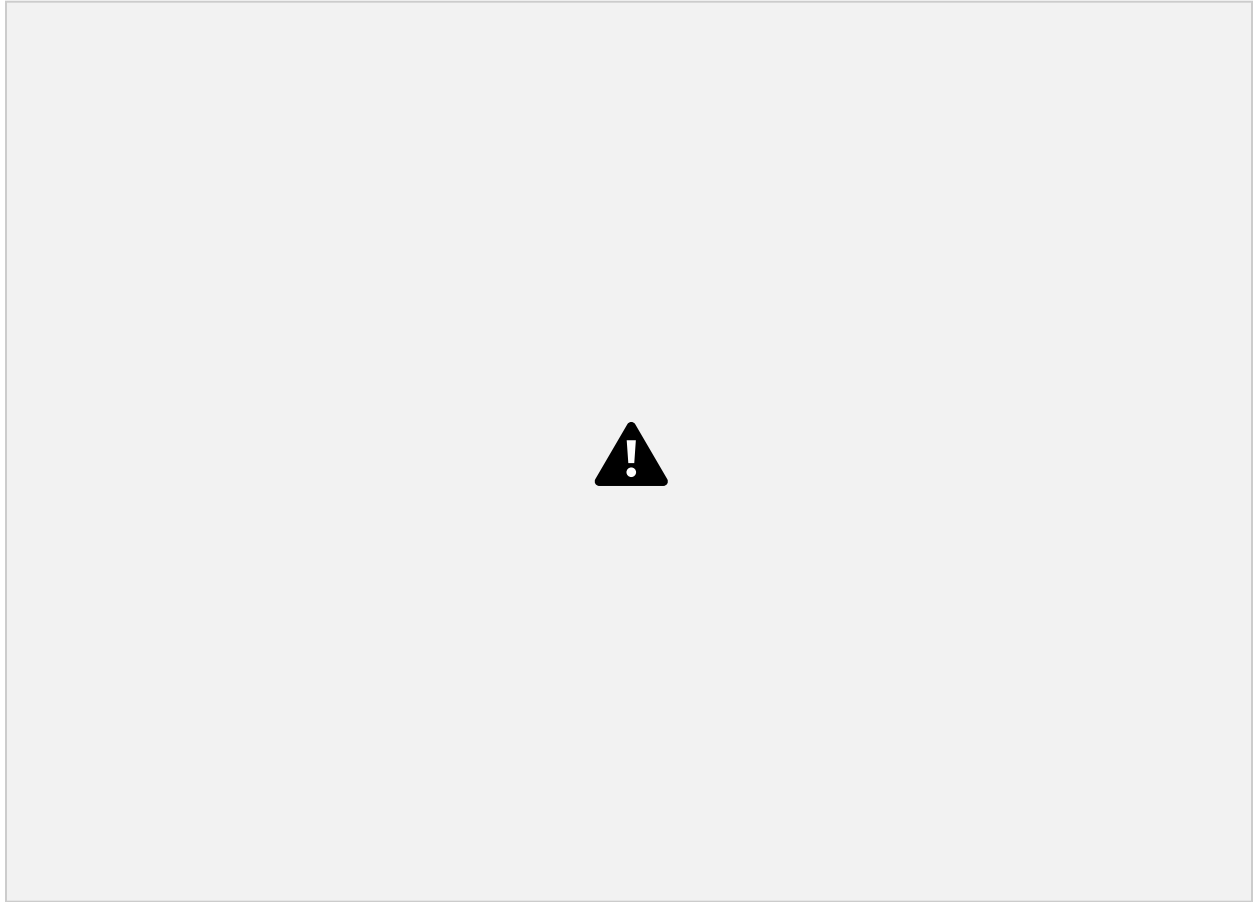


Figure 20: Update Medical History Screen Code 1

The code above displays the same connector code for the database, however the code above it configures the action of a mouse click to allow us to click on the rows of the Jtable to select the client, then we have the code to display the information from the Jtable to editable text fields in order to be updated.

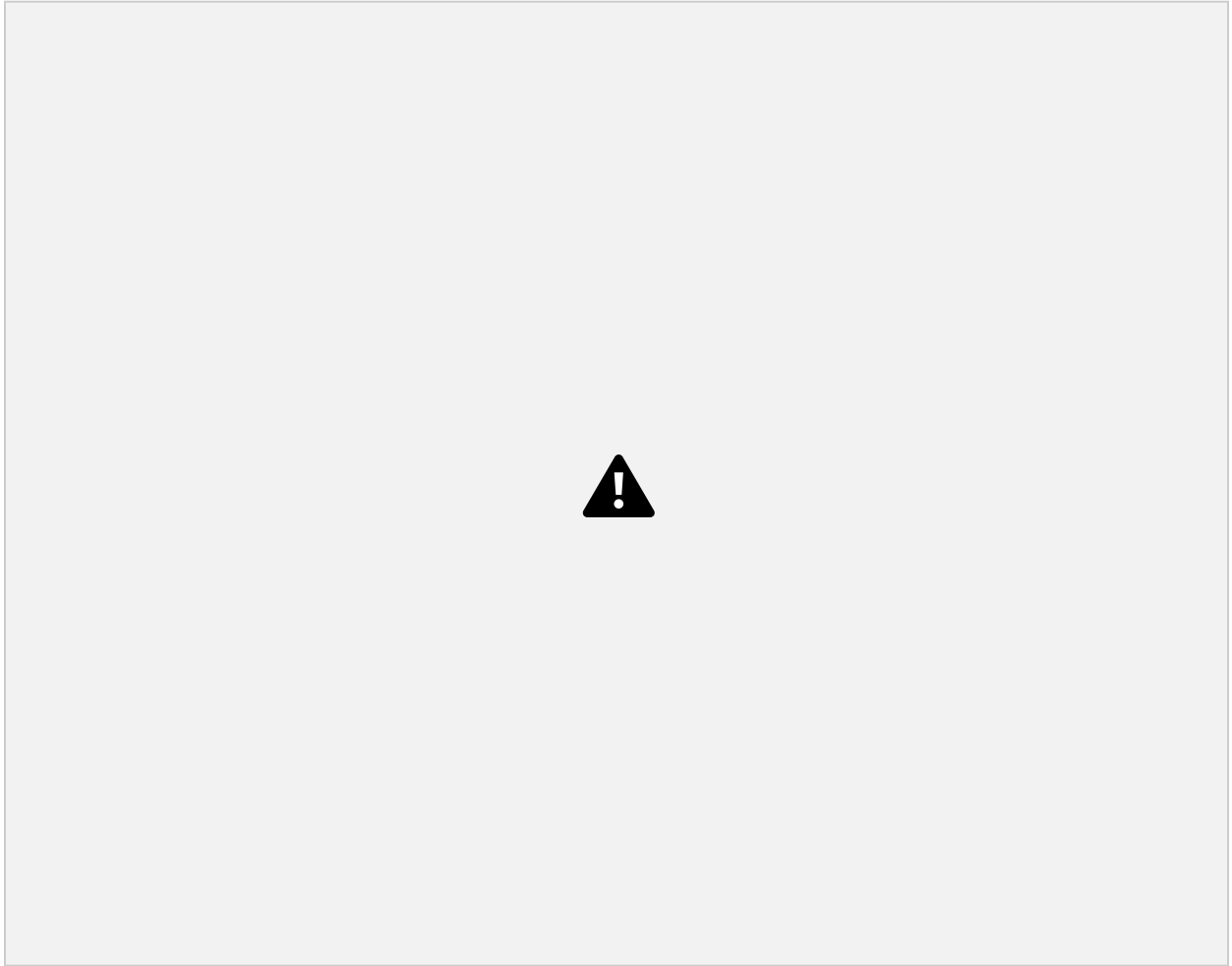


Figure 21: Update Medical History Screen Code 2

The code above is responsible for updating the medical history of the client which is performed by the service provider. As you can see, the first part of the code connects the JTable to the MySQL database, and the second part of the code obtains the new information from the text fields entered for each relating variable and displays “Updated Successfully!” upon clicking the JButton for update.

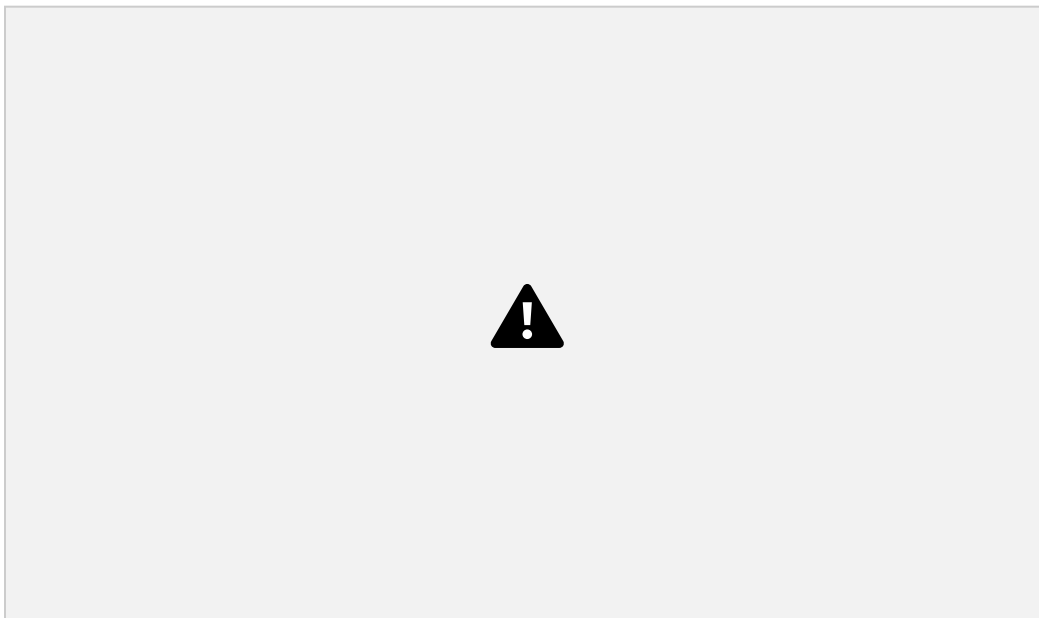


Figure 22: Service Provider Login Screen

This screen displays the Admin login for Health Care Activities. It features a username entry, a password entry along with a login button to proceed to Admin tools, and a button to go back to client login.

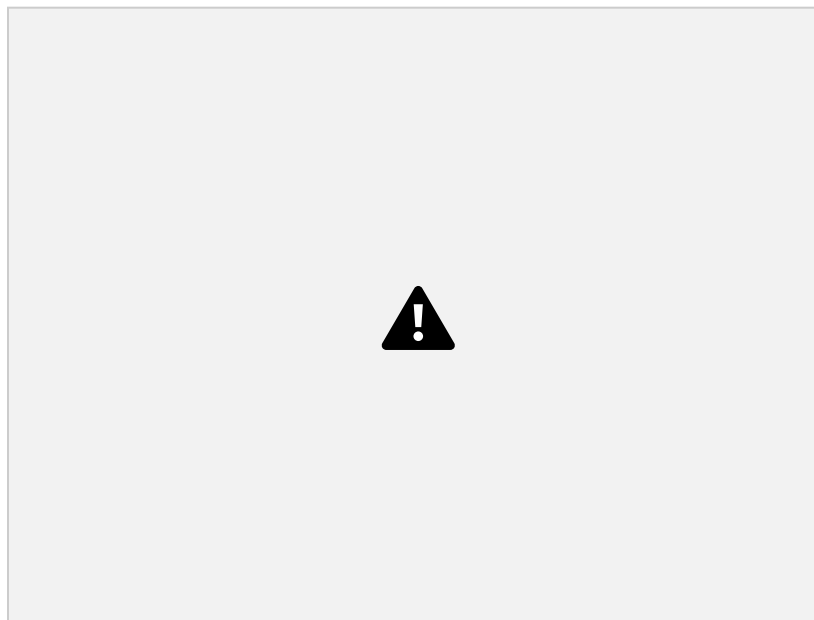


Figure 23: Admin Main Menu

This admin menu displays three options featured as Jbuttons. Firstly we have the option to Review Orders which allows the admin to see Meal Prep and Medicine Order History with the ability to delete an order. Next we have Edit Medical History where we can pull up the medical history of the given patient. Then we have a logout feature bringing us back to the login window.



Figure 24: Admin Order Review Screen

25

This is the Order history menu that admins can view displaying both Meal Prep and Medicine Order History. The Order History for both tables is displayed using Jtables that are connected to our database. Meal Prep Order History features DietType, MealQuantity, MealAddr, Food, and MealOrderID for a given order. Medicine Order History features the variables MedName, MedQuantity, MedAddr, and MedOrderID. The admin has the option to delete selected orders from both history options in case of a transaction failure.



Figure 25: Admin Order Review Screen Code

The code above displays the ability to delete a selected order from Meal Prep Order History. This occurs within a JButton which is connected to our database. After you click on the JButton the code will execute pulling up the database and deleting the given order from it. The code then displays a message saying you deleted the order successfully and then it will redisplay the table again with the updated information.

5. Earned Value Analysis

Work Task	Estimated Effort (days)	Actual Effort (days)	Estimated Completion Date	Actual Completion Date
Proposal	1	1	4/6	4/5
Deliverable (analysis)	1	1	4/8	4/7
Diagram (design)	1	1	4/8	4/7
Coding	15	10	4/25	5/2
Testing	5	2	4/30	5/3

- BCWP: 23 days

- BAC: 23 days
- EV: $BCWP/BAC = 23/23 = 100\%$
- BCWS: 23 days
- SPI: $BCWP/BCWS = 23/23 = 100\%$
- SV: $BCWP-BCWS = 23-23 = 0$ days ahead
- ACWP: 15 days
- CPI: $BCWP/ACWP = 23/15 = 153.3\%$
- CV: $BCWP-ACWP = 23-15 = 8$ days ahead

6. Testing

Black Box Testing

- Tests that the correct input, gives the correct output
- Examines the functionality and structure of the system

Functional Testing

- Focused on testing the software
- Tests system functionality
 - Typically tests each function of the application to ensure that each one is working properly

System Testing

- Tests the software on different Operating systems to ensure application runs smoothly •
- Ex: Mac and Windows

7.2 Future Enhancements

To make our Health Care portal even better, in the future we would like to add more border colors and different fonts. This would allow a more friendly user interface and would make it even more eye popping to our consumers. Another future enhancement would be to have individual service provider logins. For example, pharmacists would have an admin login and can make updates in the medicine section of our portal. Food service would have an admin login and can make updates in the diet section of our portal. We would also like to integrate this to a mobile application so that consumers can do this with a few click of buttons on their handheld devices. Lastly in the future we would add a search function in the order and medical history. This would allow users to just search whatever they need to instead of scrolling through information. This makes the app simple and more friendly for our users