

BIT203_Assignment2_E1900344

by I NYOMAN SURYA PRADIPTA -

Submission date: 09-Jan-2022 01:51PM (UTC+0800)

Submission ID: 1738885012

File name: 16112_I_NYOMAN_SURYA_PRADIPTA_-_BIT203_Assignment2_E1900344_248658_991050686.docx
(47.31M)

Word count: 19396

Character count: 158259

Assignment Cover Sheet

Student Information (For group assignment, please state names of all members)		Grade/Marks
Name	ID	
I Nyoman Surya Pradipta	E1900344	

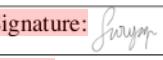
Module/Subject Information		Office Acknowledgement
Module/Subject Code	BIT203	
Module/Subject Name	Advanced OO Programming	
Lecturer/Tutor/Facilitator	Kok Chye Hock	
Due Date	1 st January 2022	
Assignment Title/Topic	Assignment 2	
Intake (where applicable)		
Word Count	19655	
		Date/Time

Declaration

- I/We have read and understood the Programme Handbook that explains on plagiarism, and I/we testify that, unless otherwise acknowledged, the work submitted herein is entirely my/our own.
- I/We declare that no part of this assignment has been written for me/us by any other person(s) except where such collaboration has been authorized by the lecturer concerned.
- I/We authorize the University to test any work submitted by me/us, using text comparison software, for instances of plagiarism. I/We understand this will involve the University or its contractors copying my/our work and storing it on a database to be used in future to test work submitted by others.

Note: 1) The attachment of this statement on any electronically submitted assignments will be deemed to have the same authority as a signed statement.

2) The Group Leader signs the declaration on behalf of all members.

Signature: 	Date: 9 January 2022
E-mail: E1900344@helplive.edu.my	

Feedback/Comments*
Main Strengths
Main Weaknesses
Suggestions for improvement

	Student acknowledge feedback/comments
Grader's signature	Student's signature: <i>Brijyan</i>
Date:	Date: 9 January 2022

Note:

- 1) A soft and hard copy of the assignment shall be submitted.
- 2) The signed copy of the assignment cover sheet shall be retained by the marker.
- 3) If the Turnitin report is required, students have to submit it with the assignment. However, departments may allow students up to **THREE (3)** working days after submission of the assignment to submit the Turnitin report. The assignment shall only be marked upon the submission of the Turnitin report.

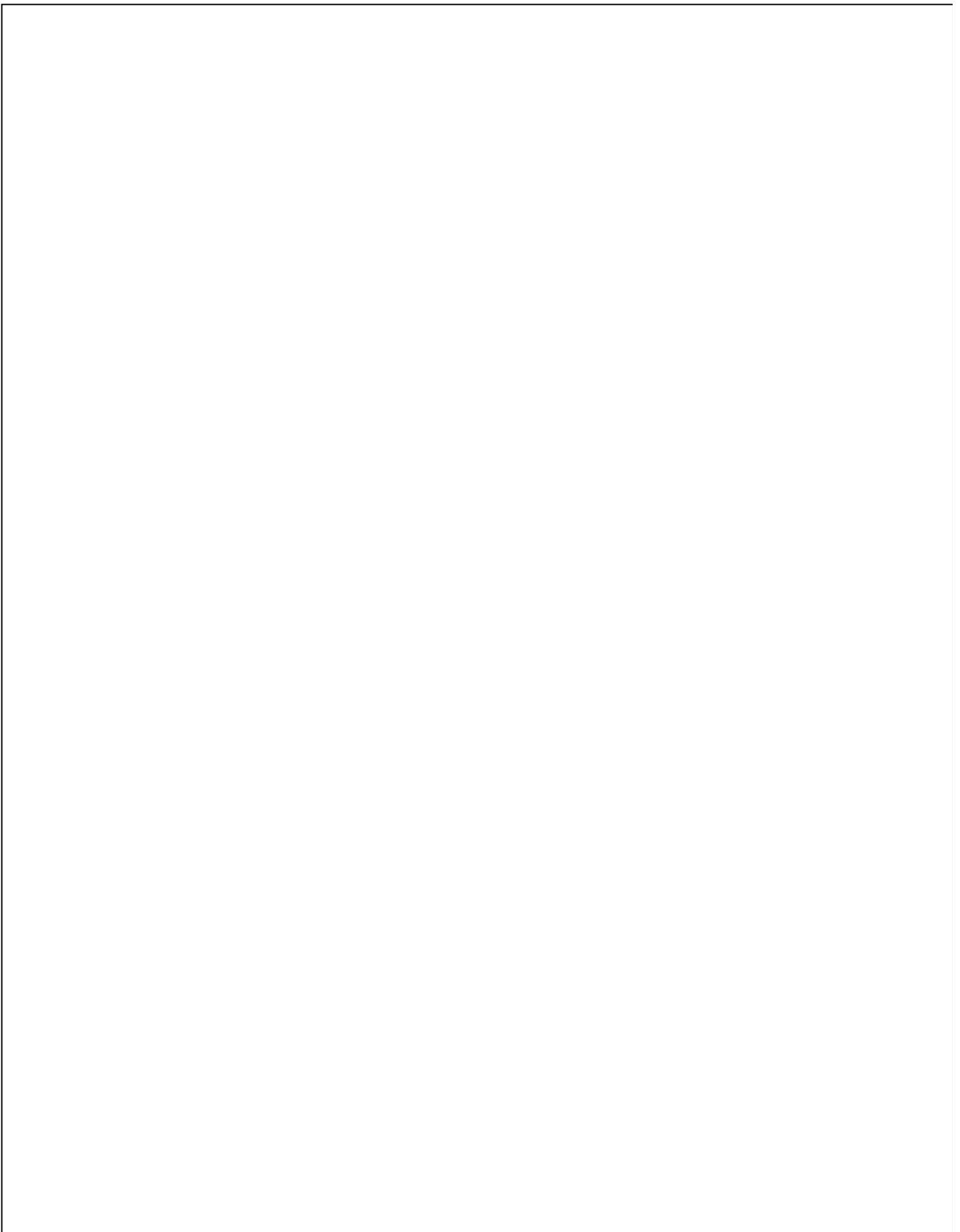
*Use additional sheets if required.

Table of Contents

Class Diagram	1
Source Code	2
PCVS	2
User	9
Administrator	12
Patient	14
Vaccine	16
HealthcareCentre	19
Batch	22
Vaccination	25
PCVSGUI	28
LoginDialog	45
AdminMenuGUI	49
VaccineBatchGUI	59
VaccinationListGUI	64
AppointmentInformationGUI	69
AdministratorInformationGUI	76
PatientInformationGUI	81
VaccinationInformationGUI	86

PatientMenuGUI.....	90
ListAvailableVaccinesGUI.....	100
ListHealthcareCentresGUI	104
AppointmentDialog	109
AdministratorTableModel	111
BatchTableModel	114
PatientTableModel.....	117
VaccinationListTableModel	120
VaccinationTableModel	123
VaccineBatchTableModel	126
Interface Design.....	129
SignUpGUI.....	Error! Bookmark not defined.
LoginDialog.....	131
AdminMenuGUI.....	132
VaccineBatchGUI.....	133
VaccinationListGUI.....	134
AppointmentInformationGUI	135
AdministratorInformationGUI	136
PatientInformationGUI	137
VaccinationInformationGUI.....	138
PatientMenuGUI.....	139

ListAvailableVaccinesGUI	140
AppointmentDialog	142
Sample Output	143
Use Case 1 Sign Up	143
1 Use Case 2 Record New Vaccine Batch	158
1 Use Case 3 Request Vaccination Appointment	187
1 Use Case 4 View Vaccine Batch Information	233
Use Case 5 Confirm Vaccination Appointment	249
1 Use Case 6 Record Vaccination Administered	264
Use Case 7 View Vaccination Appointment Status	272
Use Case 8 Detail of All User	281
Use Case 9 Display Detail of All Vaccination Appointments	285
Save File	286
Load File	291



Class Diagram

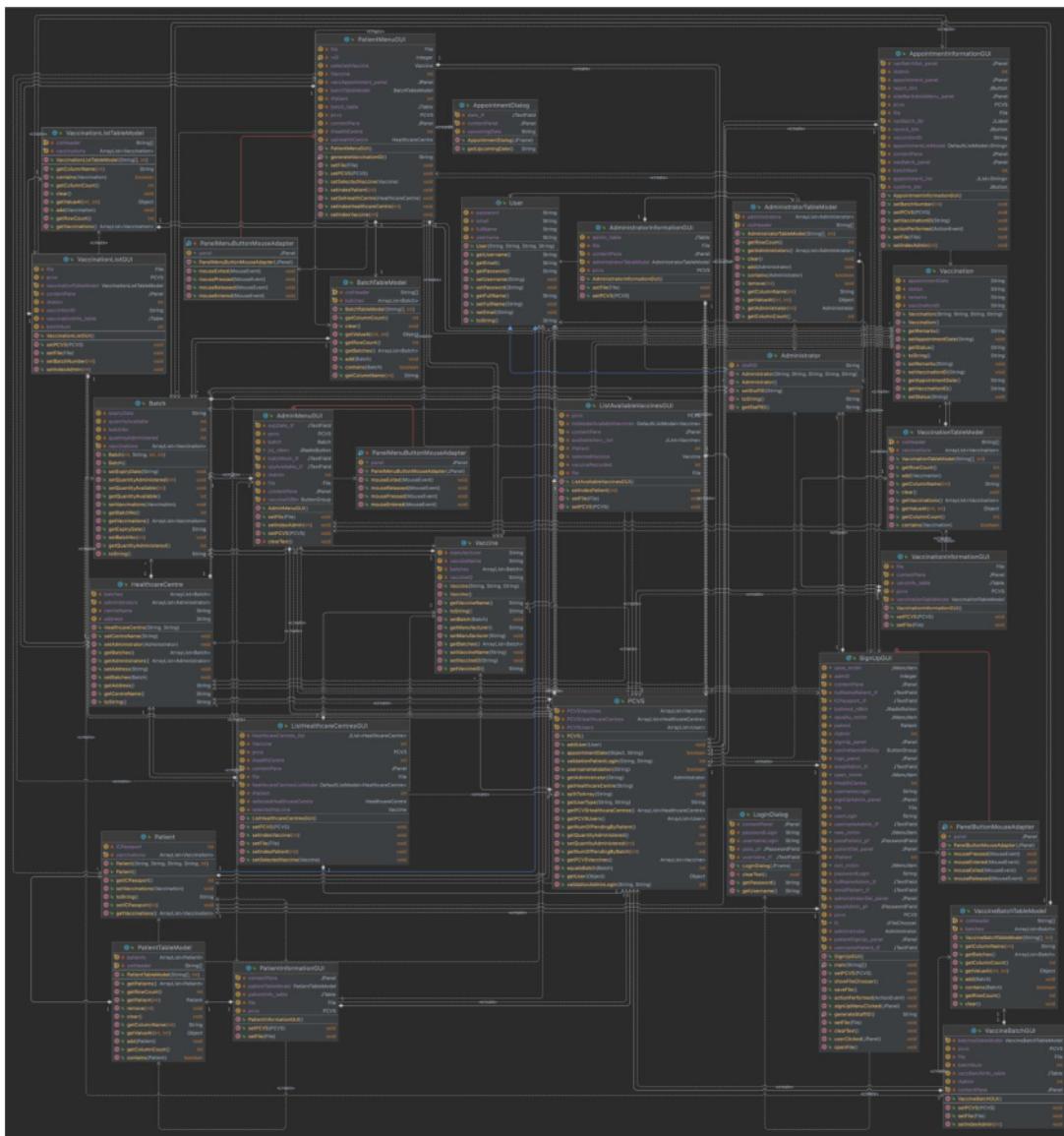


Figure 1: PCVS Class Diagram

Source Code

PCVS

```
1 package pcv;

import java.util.Arrays;
import java.util.Calendar;
import java.io.Serializable;
import java.util.Iterator;
import java.util.ArrayList;

import static java.util.stream.IntStream.range;

/**
 * The PCVS controller class defines a database
 * to control the Private Covid-19 Vaccination Scheme
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class PCVS implements Serializable {
    // ArrayList to hold collection of HealthcareCentre, User, and Vaccine
    private final ArrayList<HealthcareCentre> PCVSHC;
    private final ArrayList<User> PCVSV;
    private final ArrayList<Vaccine> PCVSVaccines;

    public PCVS() {
        // Instantiate a new ArrayList object
        PCVSV = new ArrayList<>();
        PCVSVaccines = new ArrayList<>();
        PCVSHC = new ArrayList<>();

        // Add two valid hard-coded Healthcare Centre values
        PCVSHC.add(new HealthcareCentre(
            "Balimed Hospital",
            "Jl. Mahendradatta No.57 X"));
        PCVSHC.add(new HealthcareCentre(
            "Prima Medika Hospital",
            "Jl. Raya Sesetan No.10"));

        // Add two valid hard-coded Vaccine values
        PCVSVaccines.add(new Vaccine("JNJ",
            "Janssen Pharmaceutical Companies",
            "Johnson & Johnson"));
        PCVSVaccines.add(new Vaccine("ASZ",
            "AstraZeneca, University of Oxford",
            "AstraZeneca"));
    }
}
```

```

    /**
     * Returns array of expiry date of Batch.
     *
     * @param str String to split to array.
     * @return an array value of split String
     */
    public static int[] splitToArray(String str) {
        String[] splitArray = str.split(" ");
        return Arrays.stream(splitArray).mapToInt(Integer::parseInt)
            .toArray();
    }

    /**
     * Returns this User collection.
     *
     * @return collection of User.
     */
    public ArrayList<User> getPCVSUsers() {
        return PCVSUsers;
    }

    /**
     * Returns this Healthcare Centre collection.
     *
     * @return collection of Healthcare Centre.
     */
    public ArrayList<HealthcareCentre> getPCVSHCHealthcareCentres() {
        return PCVSHCHealthcareCentres;
    }

    /**
     * Returns this Vaccine collection.
     *
     * @return collection of vaccine.
     */
    public ArrayList<Vaccine> getPCVSVaccines() {
        return PCVSVaccines;
    }

    /**
     * Registers new User to this collection.
     *
     * @param user registered User object.
     */
    public void addUser(User user) {
        PCVSUsers.add(user);
    }

    /**
     * Gets the type of User by comparing the username and password
     * in the User collection.
     * User can be patient or administrator.
     *
     * @param username the value to compare.
     * @param password the value to compare.
     * @return type of User.
     */

```

```

public String getUserType(String username, String password) {
    for (Iterator<User> i =
        PCVSUsers.iterator(); i.hasNext(); ) {
        User pcvsUser = i.next();
        if (pcvsUser.getUsername().equals(username)
            && pcvsUser.getPassword().equals(password)) {
            if (pcvsUser instanceof Patient) {
                return "patient";
            } else if (pcvsUser instanceof Administrator) {
                return "admin";
            }
        }
    }
    return null;
}

/**
 * Gets User object by comparing the object reference.
 *
 * @param obj the object to compared.
 * @return Patient or Administrator object.
 * null if object is not found.
 */
public Object getUser(Object obj) {
    for (User pcvsUser : PCVSUsers) {
        if (pcvsUser == obj) {
            return pcvsUser;
        }
    }
    return null;
}

/**
 * Returns the index of Healthcare Centre object by comparing
 * the staff ID of Administrator.
 *
 * @param staffID the String to compare.
 * @return index of Healthcare Centre.
 */
public int getHealthcareCentre(String staffID) {
    for (int k = 0; k < PCVSHealthcareCentres.size(); k++) {
        healthcareCentre hc = PCVSHealthcareCentres.get(k);
        for (int l = 0; l < hc.getAdministrators().size(); l++) {
            if (hc.getAdministrators().get(l).getStaffID()
                .equals(staffID))
                return k;
        }
    }
    return -1;
}

/**
 * Returns the Administrator object by comparing staff id.
 *
 * @param staffID the String to compare.
 * @return Administrator object, null if not found.
 */

```

```

public Administrator getAdministrator(String staffID) {
    for (HealthcareCentre hc : PCVSHealthcareCentres) {
        for (int idx = 0; idx < hc.getAdministrators().size(); idx++) {
            if (hc.getAdministrators().get(idx).getStaffID()
                .equals(staffID))
                return hc.getAdministrators().get(idx);
        }
    }
    return null;
}

/**
 * Returns boolean value by comparing usernames in Users collection.
 *
 * @param username the String to compare.
 * @return boolean value true for equal, false for not equal.
 */
public boolean usernameValidation(String username) {
    return PCVSUsers.stream()
        .anyMatch(usr -> usr.getUsername()
            .equalsIgnoreCase(username));
}

/**
 * Returns index of Healthcare Centre object by comparing
 * the username and password of Administrator to this collection
 *
 * @param username first String to compare.
 * @param password second String to compare.
 * @return an int index of HealthcareCentre object.
 */
public int validationAdminLogin(String username, String password) {
    for (int i = 0; i < getPCVSHealthcareCentres().size(); i++) {

        // Save healthcare collection.
        HealthcareCentre hc = getPCVSHealthcareCentres().get(i);
        for (int j = 0; j < hc.getAdministrators().size(); j++) {

            // Compare username
            if (hc.getAdministrators().get(j).getUsername()
                .equals(username))

                // Compare password
                if (hc.getAdministrators().get(j).getPassword()
                    .equals(password))

                    // Get index of Healthcare Centre object
                    return i;
        }
    }
    // Cannot find HealthcareCentre object
    return -1;
}

/**
 * Returns index Patient by comparing the username and password
 * to allow Patient login to PCVS application.

```

```

/*
 * @param username first String to compare.
 * @param password second String to compare.
 * @return an int index of Patient object.
 */
public int validationPatientLogin(String username, String password) {
    // Traverse the User collection
    for (int i = 0; i < getPCVSUsers().size(); i++) {

        // Downcast User collection to Patient object
        if (getPCVSUsers().get(i) instanceof Patient patient) {

            // Compare username
            if (patient.getUsername().equals(username))

                // Compare password
                if (patient.getPassword().equals(password))

                    // Get index of Patient object
                    return i;
            }
        }
        // Cannot find Patient object
        return -1;
    }

/**
 * Returns index of Healthcare Centre based on Batch object
 * to get Healthcare Centre that offering vaccines.
 *
 * @param batch object to compare.
 * @return an int index of Healthcare Centre.
 */
public int equalsBatch(Batch batch) {
    return range(0, getPCVSHCHealthcareCentres().size())
        .filter(i -> range(0, getPCVSHCHealthcareCentres().get(i)
            .getBatches().size())
            .anyMatch(j -> getPCVSHCHealthcareCentres().get(i)
                .getBatches().get(j).equals(batch)))
        .findFirst().orElse(-1);
}

/**
 * Returns number of pending of Batch.
 * Sum the pending of all Batch.
 *
 * @param iBatchNo the index batch number to compare.
 * @return an int value of number of pending.
 */
public int getNumOfPendingByBatch(int iBatchNo) {
    // Traverse the Vaccine collection
    return getPCVSVaccines().stream()
        .mapToInt(tempVC -> range(0, tempVC.getBatches().size()))
        // Same batch number
        .filter(j -> tempVC.getBatches().get(j)
            .getBatchNo() == iBatchNo)
        .mapToObj(j -> tempVC.getBatches().get(j))
}

```

```

        .mapToInt(tempBatch -> (int) range(0, tempBatch
            .getVaccinations().size())
            .filter(k -> tempBatch
                .getVaccinations().get(k)
                .getStatus().equals("pending"))
            .count()).sum());
    }

    /**
     * Returns the number of pending based on Patient collection.
     * Calculate the pending vaccination status.
     *
     * @return an int value of number of pending appointments.
     */
    public int getNumOfPendingByPatient() {
        return range(0, getPCVSUsers().size())
            .filter(i -> getPCVSUsers().get(i) instanceof Patient)
            // Downcast
            .mapToObj(i -> (Patient) getPCVSUsers().get(i))
            // Traverse Vaccination collection in Patient
            .mapToInt(patient -> (int) range(0, patient
                .getVaccinations().size())
                .filter(j -> patient.getVaccinations().get(j)
                    .getStatus().equals("pending"))
            // Increment
            .count()).sum();
    }

    /**
     *41
     * Returns the number of quantity administered Batch
     * in this Vaccine collection.
     *
     * @return an int value of number quantity administered.
     */
    public int getQuantityAdministered() {
        // Traverse Vaccine collection
        return getPCVSVaccines()
            .stream()
            // Get Batch in the Vaccine collection
            .mapToInt(tempVC -> tempVC.getBatches().stream()
                // Count quantity administered in Batch
                .mapToInt(Batch::getQuantityAdministered)
                .sum()).sum();
    }

    /**
     * Registers new quantity administered to display in Batch.
     *
     * @param inQty the value to set quantity administered.
     */
    public void setQuantityAdministered(int inQty) {
        // Traverse the Vaccine collection and Batch collection
        getPCVSVaccines().forEach(tempVaccine -> tempVaccine.getBatches()

            // Set quantity administered with inQty value
            .forEach(tempBatch -> tempBatch
                .setQuantityAdministered(inQty)));
    }
}

```

```
}

/**
 * Returns boolean value to set the local time and
 * compare the appointment date with expiry date from Batch.
 *
 * @param appointmentDate the String to compare.
 * @return a boolean value true if appointment is come first
 * before expires and vice versa.
 */
public boolean appointmentDate(Object expiryDate,
                               String appointmentDate) {
    // Split all Batch expiry date using method splitToArray
    int[] splitExpiryDate = splitToArray((String) expiryDate);

    // Create object expires to get local time from Calendar
    Calendar expires = Calendar.getInstance();

    // Set expires using expiry date from Batches
    expires.set(splitExpiryDate[2], splitExpiryDate[0],
                splitExpiryDate[1]);

    // Split appointment date from user input
    int[] splitAppointmentDate = splitToArray(appointmentDate);

    // Create object appointment to get local time from Calendar
    Calendar appointment = Calendar.getInstance();

    // Set appointment date using array
    // return value from splitToArray method
    appointment.set(splitAppointmentDate[2], splitAppointmentDate[0],
                  splitAppointmentDate[1]);

    // Appointment is come first before expires
    return !expires.after(appointment);
}
}
```

User

```
package pcvs;

import java.io.Serializable;

/**
 * User is abstract class
 * defines a simple object type that represents a User
 * and superclass of Patient and Administrator class.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public abstract class User implements Serializable {

    private String username;
    private String password;
    private String email;
    private String fullName;

    /**
     * Constructor specifying attribute of Patient or
     * Administrator objects to create.
     *
     * @param usrname the value of User is username.
     * @param pwd      the value of User is password.
     * @param eml      the value of User is email.
     * @param fName    the value of User is full name.
     */
    public User(String usrname, String pwd, String eml,
               String fName) {
        username = usrname;
        password = pwd;
        email = eml;
        fullName = fName;
    }

    /**
     * Returns this User is username account.
     *
     * @return a String value of User is username.
     */
    public String getUsername() {
        return username;
    }

    /**
     * Registers the new value of username to display in User account.
     *
     * @param usr the String to display.
     */
    public void setUsername(String usr) {
        username = usr;
    }
}
```

```
}

/**
 * Returns this User is password account.
 *
 * @return a password of User account
 */
public String getPassword() {
    return password;
}

/**
 * Registers the new value of password to display in User account.
 *
 * @param pwd the String to display.
 */
public void setPassword(String pwd) {
    password = pwd;
}

/**
 * Returns this User is email account.
 *
 * @return an email of User account
 */
public String getEmail() {
    return email;
}

/**
 * Registers the new email to display in User.
 *
 * ①
 * @param email the String to display.
 */
public void setEmail(String email) {
    this.email = email;
}

/**
 * Registers the new value of full name to display in User account.
 *
 * @return a full name of User account
 */
public String getFullName() {
    return fullName;
}

/**
 * Registers the new full name to display in User.
 *
 * ①
 * @param fullName the String to display.
 */
public void setFullName(String fullName) {
    this.fullName = fullName;
}

/**
```

```
* Returns detail information of the Patient or Administrator object.  
*  
* @return String detail Patient or Administrator information.  
*/  
@Override  
public String toString() {  
    return "User information:" +  
        "\nUsername: " + username +  
        "\nPassword: " + password +  
        "\nEmail: " + email +  
        "\nFull name: " + fullName;  
}  
}
```

Administrator

```
package pcvs;

import java.io.Serializable;

/**
 * Administrator concrete subclass of User
 * defines a simple object type that represents a Healthcare Administrator.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
1 public class Administrator extends User
    implements Serializable {

    private String staffID;

    /**
     * Constructor default value.
     */
    public Administrator() {
        this("unknown", "unknown", "unknown",
            "unknown", "unknown");
    }

    /**
     * Constructor specifying attribute of Administrator objects to create.
     *
     * @param username the value of Administrator is username.
     * @param pwd      the value of Administrator is password.
     * @param eml      the value of Administrator is email.
     * @param fName    the value of Administrator is full name.
     * @param stfID    the value of Administrator is staff id.
     */
    public Administrator(String username, String pwd, String eml,
                        String fName, String stfID) {
        super(username, pwd, eml, fName);
        staffID = stfID;
    }

    /**
     * Returns this Administrator account is staff id.
     *
     * @return a staff id of Administrator account.
     */
    public String getStaffID() {
        return staffID;
    }

    /**
     * Registers the new value of staff id to display
    
```

```
* in Administrator account.  
*  
* @param stfID the String to display.  
*/  
public void setStaffID(String stfID) {  
    this.staffID = stfID;  
}  
  
/**  
 * Returns detail information of the Administrator Object.  
 *  
 * @return String detail Administrator information.  
 */  
@Override  
public String toString() {  
    // call to string of User.  
    return super.toString() +  
        "\nStaff ID: " + staffID;  
}  
}
```

Patient

```
package pcvs;
20
import java.io.Serializable;
import java.util.ArrayList;

/**
 * Patient concrete subclass of User
 * defines a simple object type that represents a Patient.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */

public class Patient extends User implements Serializable {
    // ArrayList to hold collection of Vaccination
    private final ArrayList<Vaccination> vaccinations;
    private int ICPassport;

    /**
     * Constructor default value.
     */
    public Patient() {
        this("unknown", "unknown",
            "unknown", "unknown", 0);
    }

    /**
     * Constructor specifying attribute of Patient objects to create.
     *
     * @param username the value of Patient is username.
     * @param pwd      the value of Patient is password.
     * @param eml      the value of Patient is email.
     * @param fName    the value of Patient is full name.
     * @param icp      the value of Patient is ic passport.
     */
    public Patient(String username, String pwd, String eml,
                   String fName, int icp) {
        super(username, pwd, eml, fName);
        ICPassport = icp;
        // Instantiate a new ArrayList object of vaccinations
        this.vaccinations = new ArrayList<>();
    }

    /**
     * Returns this ic passport of Patient.
     *
     * @return a String value of ICPassport.
     */
    public int getICPassport() {
        return ICPassport;
    }
}
```

```
/**
 * Registers new IC Passport to display in Patient.
 *
 * @param ICPassport the String to display.
 */
public void setICPassport(int ICPassport) {
    this.ICPassport = ICPassport;
}

/**
 * Returns this Vaccination collection.
 *
 * @return a vaccination collection.
 */
public ArrayList<Vaccination> getVaccinations() {
    return vaccinations;
}

/**
 * Registers the Vaccination object to this collection.
 *
 * @param vc the value to registered.
 */
public void setVaccinations(Vaccination vc) {
    this.vaccinations.add(vc);
}

/**
 * Returns detail information of the Patient Object.
 *
 * @return String detail Patient information.
 */
@Override
public String toString() {
    // call to string of User.
    return super.toString() +
        "\nIC Passport: " + ICPassport;
}
}
```

Vaccine

```
package pcvs;
20
import java.io.Serializable;
import java.util.ArrayList;

/**
 * Vaccine class defines a simple object type that represents a Vaccine.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class Vaccine implements Serializable {
    private String vaccineID;
    private String manufacturer;
    private String vaccineName;
    // ArrayList to hold collection of Batch
    private final ArrayList<Batch> batches;

    /**
     * Constructor to create object with default attribute value.
     */
    public Vaccine() {
        this("unknown", "unknown", "unknown");
    }

    /**
     * Constructor specifying attribute of Vaccine objects to create.
     *
     * @param vac_id the value of vaccine is id.
     * @param mnfturer the value of vaccine is manufacturer.
     * @param vac_name the value of vaccine is name.
     */
    public Vaccine(String vac_id, String mnfturer,
                  String vac_name) {
        // Instantiate a new ArrayList object
        // of administrators and batches
        this.batches = new ArrayList<>();
        vaccineID = vac_id;
        manufacturer = mnfturer;
        vaccineName = vac_name;
    }

    /**
     * Returns this Batch collection.
     *
     * @return a Batch collection.
     */
    public ArrayList<Batch> getBatches() {
        return batches;
    }
}
```

```

    /**
     * Registers Batch object to this collectoin.
     *
     * @param batch registered batch object.
     */
    public void setBatch(Batch batch) {
        this.batches.add(batch);
    }

    /**
     * Returns this vaccine id of Vaccine object.
     *
     * @return a String value of vaccineID.
     */
    public String getVaccineID() {
        return vaccineID;
    }

    /**
     * Registers new value of vaccine id
     * to display in vaccine.
     *
     * @param vacID the string to display.
     */
    public void setVaccineID(String vacID) {
        this.vaccineID = vacID;
    }

    /**
     * Returns this Vaccine is manufacturer.
     *
     * @return this manufacturer of Vaccine value.
     */
    public String getManufacturer() {
        return manufacturer;
    }

    /**
     * Registers new manufacturer to display in Vaccine.
     *
     * @param mnfturer the string to display.
     */
    public void setManufacturer(String mnfturer) {
        manufacturer = mnfturer;
    }

    /**
     * Returns this vaccine name of Vaccine object.
     *
     * @return a String value of vaccineName.
     */
    public String getVaccineName() {
        return vaccineName;
    }

    /**

```

```
* Registers new vaccine name to display in Vaccine.  
*  
* @param vac_name the string to display.  
*/  
public void setVaccineName(String vac_name) {  
    vaccineName = vac_name;  
}  
  
/**  
 * Returns detail information of the Vaccine object.  
 *  
 * @return String detail Vaccine information.  
 */  
@Override  
public String toString() {  
    return vaccineName + " vaccine, developed by " + manufacturer;  
}  
}
```

HealthcareCentre

```
package pcvs;
20
import java.io.Serializable;
import java.util.ArrayList;

/**
 * HealthcareCentre class defines a simple object type
 * that represents a HealthcareCentre.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class HealthcareCentre implements Serializable {
    // ArrayList to hold collection of Administrator and Batch
    private final ArrayList<Administrator> administrators;
    private final ArrayList<Batch> batches;
    private String centreName;
    private String address;

    /**
     * Constructor specifying attribute of Healthcare Centre
     * objects to create.
     *
     * @param cent_name   the Healthcare Centre is name.
     * @param cent_address the Healthcare Centre is address.
     */
    public HealthcareCentre(String cent_name, String cent_address) {
        centreName = cent_name;
        address = cent_address;
        // Instantiate a new ArrayList object of administrators and batches
        this.administrators = new ArrayList<>();
        this.batches = new ArrayList<>();
    }

    /**
     * Returns this Administrator collection.
     *
     * @return this collection of Administrator.
     */
    public ArrayList<Administrator> getAdministrators() {
        return administrators;
    }

    /**
     * Registers Administrator object to HealthcareCentre collection.
     *
     * @param admin registered administrator object.
     */
    public void setAdministrator(Administrator admin) {
        this.administrators.add(admin);
    }
}
```

```
/**  
 * Returns this Batch collection.  
 *  
 * @return this Batch collection.  
 */  
public ArrayList<Batch> getBatches() {  
    return batches;  
}  
  
/**  
 * Registers Batch object to Healthcare Collection.  
 *  
 * @param bacth registered Batch object.  
 */  
public void setBatches(Batch bacth) {  
    this.batches.add(bacth);  
}  
  
/**  
 * Gets this centre 1 name of Healthcare Centre.  
 *  
 * @return the Healthcare Centre is name.  
 */  
public String getCentreName() {  
    return centreName;  
}  
  
/**  
 * Registers the new centre name to display in Healthcare Centre.  
 *  
 * @param cent_name the String to display.  
 */  
public void setCentreName(String cent_name) {  
    centreName = cent_name;  
}  
  
/**  
 * Gets the Healthcare Centre is address.  
 *  
 * @return Healthcare Centre is 1 address.  
 */  
public String getAddress() {  
    return address;  
}  
  
/**  
 * Registers the Healthcare Centre is address to display.  
 *  
 * @param cent_address the String to display.  
 */  
public void setAddress(String cent_address) {  
    this.address = cent_address;  
}  
  
/**  
 * Returns Healthcare Centre is detail information.  
 */
```

```
* @return the String detail Healthcare Centre information.  
*/  
@Override  
public String toString() {  
    return centreName + " is located at " + address;  
}  
}
```

Batch

```
package pcvs;
44
import java.util.ArrayList;
import java.io.Serializable;

/**
 * Batch class defines a simple object type that represents a Batch.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
1
public class Batch implements Serializable {
    private int batchNo;
    private int quantityAvailable;
    private int quantityAdministered;
    private String expiryDate;
    // ArrayList to hold collection of Vaccination
    private final ArrayList<Vaccination> vaccinations;

    /**
     * Constructor to create object with default attribute value.
     */
    public Batch() {
        this(0, "", 0, 0);
    }

    /**
     * Constructor specifying attribute of Batch objects to create.
     *
     * @param bNo          the batch number of Batch.
     * @param exp          the expiry date of Batch.
     * @param qty_available the quantity available of Batch.
     * @param qty_administered the quantity administered of Batch.
     */
    public Batch(int bNo, String exp, int qty_available,
                int qty_administered) {
        // Instantiate a new ArrayList object of vaccinations
        this.vaccinations = new ArrayList<>();
        batchNo = bNo;
        expiryDate = exp;
        quantityAvailable = qty_available;
        quantityAdministered = qty_administered;
    }

    /**
     * Gets the vaccination collection in Batch collection.
     *
     * @return an collection of vaccination.
     */
```

```

        */
    public ArrayList<Vaccination> getVaccinations() {
        return vaccinations;
    }

    /**
     * Registers the vaccination to Batch collection.
     *
     * @param vaccination the registered vaccination object.
     */
    public void setVaccinations(Vaccination vaccination) {
        this.vaccinations.add(vaccination);
    }

    /**
     * Returns this batch number of Batch.
     *
     * @return batch number of Batch.
     */
    public int getBatchNo() {
        return batchNo;
    }

    /**
     * Registers the new batch number to display in Batch.  

     * 1
     * @param bNo the String to display.
     */
    public void setBatchNo(int bNo) {
        batchNo = bNo;
    }

    /**
     * Returns this expiry date of Batch.
     *
     * @return a String value of expiry date.
     */
    public String getExpiryDate() {
        return expiryDate;
    }

    /**
     * Registers the expiry date to display in Batch.
     *
     * @param exp the String to display.
     */
    public void setExpiryDate(String exp) {
        this.expiryDate = exp;
    }

    /**
     * Returns this quantity available of Batch.
     *
     * @return an int value of quantity available.
     */
    public int getQuantityAvailable() {
        return quantityAvailable;
    }
}

```

```
}

/**
 * Registers the quantity available to display in Batch.
 *
 * @param qty_available the int to display.
 */
public void setQuantityAvailable(int qty_available) {
    quantityAvailable = qty_available;
}

/**
 * Returns this quantity administered of Batch.
 *
 * @return an int value of quantity administered.
 */
public int getQuantityAdministered() {
    return quantityAdministered;
}

/**
 * Registers the quantity administered to display in Batch.
 *
 * @param qty_administered the int to display.
 */
public void setQuantityAdministered(int qty_administered) {
    this.quantityAdministered = qty_administered;
}

/**
 * Returns detail information of Batch.
 *
 * @return String detail Batch information.
 */
@Override
public String toString() {
    return "Batch information:" +
        "\nBatch number: " + batchNo +
        "\nExpiry date: " + expiryDate +
        "\nQuantity available: " + quantityAvailable + "\n";
}
}
```

Vaccination

```
package pcvs;

import java.io.Serializable;

/**
 * Vaccination class defines a simple object type
 * that represents a Vaccination.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
1 public class Vaccination implements Serializable {
    private String vaccinationID;
    private String status;
    private String remarks;
    private String appointmentDate;

    /**
     * Constructor to create object with default attribute value.
     */
    public Vaccination() {
        this("unknown", "unknown",
            "unknown", "unknown");
    }

    /**
     * Constructor specifying attribute of Vaccination objects to create.
     *
     * @param vacID the vaccination 1 of Vaccination.
     * @param dt the appointment date of Vaccination.
     * @param stts the status of Vaccination.
     * @param rmks the remarks of Vaccination.
     */
    public Vaccination(String vacID, String dt,
                       String stts, String rmks) {
        vaccinationID = vacID;
        appointmentDate = dt;
        status = stts;
        remarks = rmks;
    }

    /**
     * Returns this vaccination id of Vaccination object.
     *
     * @return a String value of vaccinationID.
     */
    public String getVaccinationID() {
        return vaccinationID;
    }
}
```

```
/**  
 * Registers the new vaccination id to display in Vaccination.  
 *  
 * @param vacId the String to display.  
 */  
public void setVaccinationID(String vacId) {  
    this.vaccinationID = vacId;  
}  
  
/**  
 * Returns this appointment date of Vaccination object.  
 *  
 * @return this Vaccination is date.  
 */  
public String getAppointmentDate() {  
    return appointmentDate;  
}  
  
/**  
 * Registers the new appointment date to display in Vaccination.  
 *  
 * @param dt the String to display.  
 */  
public void setAppointmentDate(String dt) {  
    appointmentDate = dt;  
}  
  
/**  
 * Returns this remarks of Vaccination object.  
 *  
 * @return this Vaccination is remarks.  
 */  
public String getRemarks() {  
    return remarks;  
}  
  
/**  
 * Registers the new remarks to display in Vaccination.  
 *  
 * @param rmks the String to display.  
 */  
public void setRemarks(String rmks) {  
    remarks = rmks;  
}  
  
/**  
 * Returns this status of Vaccination object.  
 *  
 * @return this Vaccination is status.  
 */  
public String getStatus() {  
    return status;  
}  
  
/**  
 * Registers the new status to display in Vaccination.
```

```
* @param stts the value to display.
 */
public void setStatus(String stts) {
    status = stts;
}

/**
 * Returns detail information of the Vaccination Object.
 *
 * @return String detail Vaccination information.
 */
@Override
public String toString() {
    return "Vaccination ID: " + vaccinationID +
        "\nAppointment date: " + appointmentDate +
        "\nStatus: " + status + "\n";
}
}
```

PCVSGUI

```
package frame;

import dialog.LoginDialog;
import pcvs.Administrator;
import pcvs.PCVS;
import pcvs.Patient;
10
import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.io.*;

import static java.awt.Color.*;
import static java.awt.Font.*;
import static javax.swing.JFileChooser.*;
import static javax.swing.JOptionPane.*;
import static javax.swing.SwingConstants.*;

/**
 * PCVSGUI class that will display the main page to the user and
 * as a repository for pcvs data objects to file.
 * PCVSGUI allows users to register for accounts
 * as a patient or administrator.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
39/
public class PCVSGUI extends JFrame
    implements ActionListener {

    private static Integer admID = 0;
    final JFileChooser f1Chooser = new JFileChooser();
    private final JPanel signUpAdmin_pnl;
32    private final JPanel patientSignUp_pnl;
    private final JTextField usernameAdmin_tf;
    private final JTextField emailAdmin_tf;
    private final JTextField fullNameAdmin_tf;
36    private final JPasswordField passAdmin_pf;
    JMenuItem open_mntm;
    JMenuItem new_mntm;
    JMenuItem save_mntm;
    JMenuItem saveAs_mntm;
25    JMenuItem exit_mntm;
    private final JTextField usernamePatient_tf;
    private final JTextField emailPatient_tf;
    private final JTextField fullNamePatient_tf;
```

```

private final JPasswordField passPatient_pf;
private final JTextField ICPassport_tf;
private final JPanel administratorSel_pnl;
private final JPanel patientSel_pnl;
JRadioButton balimed_rdbtn;
// Global attribute used on methods or listeners.
private PCVS pcvs;
private String usernameLogin;
private File fileDir = null;
private String passwordLogin;
private String userLogin;
private Patient patient;
private int iAdmin;
private int iPatient;
private int iHealthCentre;
private Administrator administrator;

/**
 * Class constructor to
 * create GUI for sign up user.
 */
public PCVSGUI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(250, 150, 1000, 600);

    JMenuBar pcvsMenuBar = new JMenuBar();
    pcvsMenuBar.setFont(new Font("Montserrat", PLAIN, 14));
    pcvsMenuBar.setBorderPainted(false);
    pcvsMenuBar.setBackground(WHITE);
    setJMenuBar(pcvsMenuBar);

    JMenuItem file_mn = new JMenu("File");
    file_mn.setFont(new Font(".AppleSystemUIFont", PLAIN, 14));
    pcvsMenuBar.add(file_mn);

    new_mntm = new JMenuItem("New");
    file_mn.add(new_mntm);

    open_mntm = new JMenuItem("Open");
    file_mn.add(open_mntm);

    save_mntm = new JMenuItem("Save");
    file_mn.add(save_mntm);

    saveAs_mntm = new JMenuItem("Save As");
    file_mn.add(saveAs_mntm);

    exit_mntm = new JMenuItem("Exit");
    file_mn.add(exit_mntm);

    // Add a listener to call when the condition is true.
    new_mntm.addActionListener(this);
    saveAs_mntm.addActionListener(this);
    open_mntm.addActionListener(this);
    save_mntm.addActionListener(this);
    exit_mntm.addActionListener(this);
}

```

```

// Global component used on methods or listeners.
JPanel content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel logo_pnl = new JPanel();
logo_pnl.setBounds(0, 0, 400, 550);
content_pnl.add(logo_pnl);
logo_pnl.setLayout(null);

JLabel logo_lbl = new JLabel("");
logo_lbl.setIcon(new ImageIcon(
    PCVSGUI.class.getResource("/res/signUp.png")));
logo_lbl.setBounds(0, 0, 400, 550);
logo_pnl.add(logo_lbl);

JPanel signUp_pnl = new JPanel();
signUp_pnl.setBackground(WHITE);
signUp_pnl.setBounds(400, 0, 600, 550);
content_pnl.add(signUp_pnl);
signUp_pnl.setLayout(null);

JPanel login_pnl = new JPanel();
login_pnl.setOpaque(false);
login_pnl.setBounds(0, 0, 600, 125);
signUp_pnl.add(login_pnl);
login_pnl.setBackground(WHITE);
login_pnl.setLayout(null);

JLabel signUptitle_lbl = new JLabel("Sign up to PCVS ");
signUptitle_lbl.setForeground(DARK_GRAY);
signUptitle_lbl.setBounds(130, 57, 179, 40);
login_pnl.add(signUptitle_lbl);
signUptitle_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 18));

JLabel description_lbl = new JLabel("Already have account?");
description_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
description_lbl.setBounds(130, 103, 179, 16);
login_pnl.add(description_lbl);
description_lbl.setForeground(GRAY);

JButton login_btn = new JButton("Log in");
login_btn.addActionListener(e -> {
    LoginDialog login = new LoginDialog(PCVSGUI.this);
    login.pack();
    login.setBounds(0, 0, 1000, 600);
    login.setLocationRelativeTo(PCVSGUI.this);
    login.setVisible(true);

    usernameLogin = login.getUsernameAcc();
    passwordLogin = login.getPasswordAcc();
}

```

```

// Gets the user type , either patient or administrator.
userLogin = pcvs.getUserType(usernameLogin, passwordLogin);

// Get admin and patient indexes
// as a way to get objects by index.
iAdmin = pcvs.validationAdminLogin(
    usernameLogin, passwordLogin);
iPatient = pcvs.validationPatientLogin(
    usernameLogin, passwordLogin);

if (usernameLogin != null && passwordLogin != null) {
    if (userLogin != null) {
        if (userLogin.equals("admin")) {
            showMessageDialog(PCVSGUI.this,
                "Welcome to " +
                pcvs.getPCVSHealthcareCentres()
                    .get(iAdmin)
                    .getCentreName(),
                null, PLAIN_MESSAGE);

            // Launch the administrator menu GUI.
            AdminMenuGUI adminMenu = new AdminMenuGUI();
            adminMenu.pack();
            adminMenu.setBounds(0, 0, 1000, 600);
            adminMenu.setLocationRelativeTo(PCVSGUI.this);

            // Copy value of attribute to another JFrame.
            adminMenu.setPCVS(pcvs);
            adminMenu.setFile(fileDir);
            adminMenu.setIndexAdmin(iAdmin);

            adminMenu.setVisible(true);
            setVisible(false);
        } else if (userLogin.equals("patient")) {
            showMessageDialog(PCVSGUI.this,
                "Welcome, " + pcvs.getPCVSUsers()
                    .get(iPatient)
                    .getFullName(),
                null, PLAIN_MESSAGE);

            // Launch patient menu GUI.
            PatientMenuGUI patientMenu = new PatientMenuGUI();
            patientMenu.pack();
            patientMenu.setBounds(0, 0, 1000, 600);
            patientMenu.setLocationRelativeTo(PCVSGUI.this);

            // Copy value of attribute to another JFrame.
            patientMenu.setPCVS(pcvs);
            patientMenu.setIndexPatient(iPatient);
            patientMenu.setFile(fileDir);

            patientMenu.setVisible(true);
            setVisible(false);
        }
    } else
        showMessageDialog(PCVSGUI.this,
            "Can't find your account", null,

```

```

        WARNING_MESSAGE);
    } else
        showMessageDialog(PCVSGUI.this,
            "Sign in canceled", null,
            WARNING_MESSAGE);
    });
login_btn.setVerticalAlignment(BOTTOM);
login_btn.setForeground(new Color(79, 189, 194));
login_btn.setBackground(WHITE);
login_btn.setBorder(null);
login_btn.setOpaque(true);
login_btn.setBounds(285, 90, 39, 29);
login_pnl.add(login_btn);

signUpAdmin_pnl = new JPanel();
signUpAdmin_pnl.setLayout(null);
signUpAdmin_pnl.setBackground(WHITE);
signUpAdmin_pnl.setBounds(0, 124, 600, 426);
signUp_pnl.add(signUpAdmin_pnl);

JLabel usernameAdmin_lbl = new JLabel("Username");
usernameAdmin_lbl.setForeground(DARK_GRAY);
usernameAdmin_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
usernameAdmin_lbl.setBounds(130, 83, 94, 18);
signUpAdmin_pnl.add(usernameAdmin_lbl);

JPanel usernameAdmin_pnl = new JPanel();
usernameAdmin_pnl.setLayout(null);
usernameAdmin_pnl.setOpaque(false);
usernameAdmin_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
usernameAdmin_pnl.setBounds(130, 113, 125, 40);
signUpAdmin_pnl.add(usernameAdmin_pnl);

usernameAdmin_tf = new JTextField();
usernameAdmin_tf.setFont(
    new Font("AppleSystemUIFont", PLAIN, 12));
usernameAdmin_tf.setForeground(DARK_GRAY);
usernameAdmin_tf.setColumns(10);
usernameAdmin_tf.setBorder(null);
usernameAdmin_tf.setBackground(WHITE);
usernameAdmin_tf.setBounds(6, 6, 113, 28);
usernameAdmin_pnl.add(usernameAdmin_tf);

JLabel emailAdmin_lbl = new JLabel("Email");
emailAdmin_lbl.setForeground(DARK_GRAY);
emailAdmin_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
emailAdmin_lbl.setBounds(130, 165, 94, 18);
signUpAdmin_pnl.add(emailAdmin_lbl);

JPanel emailAdmin_pnl = new JPanel();
emailAdmin_pnl.setLayout(null);
emailAdmin_pnl.setOpaque(false);
emailAdmin_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));

```

```

emailAdmin_pnl.setBounds(130, 195, 262, 40);
signUpAdmin_pnl.add(emailAdmin_pnl);

emailAdmin_tf = new JTextField();
emailAdmin_tf.setFont(
    new Font(".AppleSystemUIFont", PLAIN, 12));
emailAdmin_tf.setForeground(DARK_GRAY);
emailAdmin_tf.setColumns(10);
emailAdmin_tf.setBorder(null);
emailAdmin_tf.setBackground(WHITE);
emailAdmin_tf.setBounds(6, 6, 250, 28);
emailAdmin_pnl.add(emailAdmin_tf);

JLabel fullNameAdmin_lbl = new JLabel("Full Name");
fullNameAdmin_lbl.setForeground(DARK_GRAY);
fullNameAdmin_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
fullNameAdmin_lbl.setToolTipText("");
fullNameAdmin_lbl.setBounds(130, 247, 94, 18);
signUpAdmin_pnl.add(fullNameAdmin_lbl);

JPanel fullNameAdmin_pnl = new JPanel();
fullNameAdmin_pnl.setLayout(null);
fullNameAdmin_pnl.setOpaque(false);
fullNameAdmin_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
fullNameAdmin_pnl.setBounds(130, 277, 262, 40);
signUpAdmin_pnl.add(fullNameAdmin_pnl);

fullNameAdmin_tf = new JTextField();
fullNameAdmin_tf.setForeground(DARK_GRAY);
fullNameAdmin_tf.setFont(
    new Font(".AppleSystemUIFont", PLAIN, 12));
fullNameAdmin_tf.setColumns(10);
fullNameAdmin_tf.setBorder(null);
fullNameAdmin_tf.setBackground(WHITE);
fullNameAdmin_tf.setBounds(6, 6, 250, 28);
fullNameAdmin_pnl.add(fullNameAdmin_tf);

JLabel passwordAdmin_lbl = new JLabel("Password");
passwordAdmin_lbl.setForeground(DARK_GRAY);
passwordAdmin_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
passwordAdmin_lbl.setBounds(267, 83, 94, 18);
signUpAdmin_pnl.add(passwordAdmin_lbl);

JPanel passAdmin_pnl = new JPanel();
passAdmin_pnl.setLayout(null);
passAdmin_pnl.setOpaque(false);
passAdmin_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
passAdmin_pnl.setBounds(267, 113, 125, 40);
signUpAdmin_pnl.add(passAdmin_pnl);

passAdmin_pf = new JPasswordField();
passAdmin_pf.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 9));

```

```

passAdmin_pf.setForeground(DARK_GRAY);
passAdmin_pf.setBorder(null);
passAdmin_pf.setBounds(6, 6, 113, 28);
passAdmin_pnl.add(passAdmin_pf);

patientSignUp_pnl = new JPanel();
patientSignUp_pnl.setLayout(null);
patientSignUp_pnl.setBackground(WHITE);
patientSignUp_pnl.setBounds(0, 124, 600, 426);
signUp_pnl.add(patientSignUp_pnl);

JLabel usernamePatient_lbl = new JLabel("Username");
usernamePatient_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
usernamePatient_lbl.setBounds(130, 16, 94, 18);
patientSignUp_pnl.add(usernamePatient_lbl);

JPanel usernamePatient_pnl = new JPanel();
usernamePatient_pnl.setLayout(null);
usernamePatient_pnl.setOpaque(false);
usernamePatient_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
usernamePatient_pnl.setBounds(130, 46, 125, 40);
patientSignUp_pnl.add(usernamePatient_pnl);

usernamePatient_tf = new JTextField();
usernamePatient_tf.setForeground(DARK_GRAY);
usernamePatient_tf.setFont(
    new Font("Helvetica", PLAIN, 13));
usernamePatient_tf.setColumns(10);
usernamePatient_tf.setBorder(null);
usernamePatient_tf.setBackground(WHITE);
usernamePatient_tf.setBounds(6, 6, 113, 28);
usernamePatient_pnl.add(usernamePatient_tf);

JLabel emailPatient_lbl = new JLabel("Email");
emailPatient_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
emailPatient_lbl.setBounds(130, 98, 94, 18);
patientSignUp_pnl.add(emailPatient_lbl);

JPanel emailPatient_pnl = new JPanel();
emailPatient_pnl.setLayout(null);
emailPatient_pnl.setOpaque(false);
emailPatient_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
emailPatient_pnl.setBounds(130, 128, 262, 40);
patientSignUp_pnl.add(emailPatient_pnl);

emailPatient_tf = new JTextField();
emailPatient_tf.setForeground(DARK_GRAY);
emailPatient_tf.setFont(new Font("Helvetica", PLAIN, 13));
emailPatient_tf.setColumns(10);
emailPatient_tf.setBorder(null);
emailPatient_tf.setBackground(WHITE);
emailPatient_tf.setBounds(6, 6, 250, 28);
emailPatient_pnl.add(emailPatient_tf);

```

```

JLabel fullNamePatient_lbl = new JLabel("Full Name");
fullNamePatient_lbl.setToolTipText("");
fullNamePatient_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
fullNamePatient_lbl.setBounds(130, 180, 94, 18);
patientSignUp_pnl.add(fullNamePatient_lbl);

JPanel fullNamePatient_pnl = new JPanel();
fullNamePatient_pnl.setLayout(null);
fullNamePatient_pnl.setOpaque(false);
fullNamePatient_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
fullNamePatient_pnl.setBounds(130, 210, 262, 40);
patientSignUp_pnl.add(fullNamePatient_pnl);

fullNamePatient_tf = new JTextField();
fullNamePatient_tf.setForeground(DARK_GRAY);
fullNamePatient_tf.setFont(
    new Font("Helvetica", PLAIN, 13));
fullNamePatient_tf.setColumns(10);
fullNamePatient_tf.setBorder(null);
fullNamePatient_tf.setBackground(WHITE);
fullNamePatient_tf.setBounds(6, 6, 250, 28);
fullNamePatient_pnl.add(fullNamePatient_tf);

JLabel passPatient_lbl = new JLabel("Password");
passPatient_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
passPatient_lbl.setBounds(267, 16, 94, 18);
patientSignUp_pnl.add(passPatient_lbl);

JPanel passPatient_pnl = new JPanel();
passPatient_pnl.setLayout(null);
passPatient_pnl.setOpaque(false);
passPatient_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
passPatient_pnl.setBounds(267, 46, 125, 40);
patientSignUp_pnl.add(passPatient_pnl);

passPatient_pf = new JPasswordField();
passPatient_pf.setForeground(DARK_GRAY);
passPatient_pf.setFont(new Font("Helvetica", PLAIN, 9));
passPatient_pf.setBorder(null);
passPatient_pf.setBounds(6, 6, 113, 28);
passPatient_pnl.add(passPatient_pf);

JLabel ICPassportPatient_lbl = new JLabel("IC or Passport");
ICPassportPatient_lbl.setToolTipText("");
ICPassportPatient_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
ICPassportPatient_lbl.setBounds(130, 262, 98, 18);
patientSignUp_pnl.add(ICPassportPatient_lbl);

JPanel ICPassport_pnl = new JPanel();
ICPassport_pnl.setLayout(null);
ICPassport_pnl.setOpaque(false);

```

```

ICPassport_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
ICPassport_pnl.setBounds(130, 292, 262, 40);
patientSignUp_pnl.add(ICPassport_pnl);

ICPassport_tf = new JTextField();
ICPassport_tf.setForeground(DARK_GRAY);
ICPassport_tf.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
ICPassport_tf.setColumns(10);
ICPassport_tf.setBorder(null);
ICPassport_tf.setBackground(WHITE);
ICPassport_tf.setBounds(6, 6, 250, 28);
ICPassport_pnl.add(ICPassport_tf);

JButton btnCreateAnAccountPT = new JButton("Create an Account");
btnCreateAnAccountPT.addActionListener(e -> {
    String username = usernamePatient_tf.getText().trim();
    String pwd = passPatient_pf.getText().trim();
    String eml = emailPatient_tf.getText().trim();
    String fName = fullNamePatient_tf.getText().trim();
    String icPassStr = ICPassport_tf.getText().trim();

    if (username.isEmpty() || pwd.isEmpty() || eml.isEmpty()
        || fName.isEmpty() || icPassStr.isEmpty()) {
        showMessageDialog(PCVSGUI.this,
            "Please fill in all fields",
            null, WARNING_MESSAGE);
        usernamePatient_tf.requestFocus();
    } else {
        try {
            int ICPassport = Integer.parseInt(icPassStr);
            patient = new Patient(username, pwd, eml,
                fName, ICPassport);

            // Validate username to avoid duplicate username.
            if (pcvs.usernameValidation(patient.getUsername())) {
                showMessageDialog(PCVSGUI.this,
                    "That username is taken! " +
                    "Try another username",
                    null, WARNING_MESSAGE);
            } else {
                pcvx.addUser(patient);
                showMessageDialog(PCVSGUI.this,
                    "Sign up success! A Patient account " +
                    "is created\n\n" +
                    + patient.toString(),
                    null, PLAIN_MESSAGE);
                clearText();
            }
        } catch (NumberFormatException nfe) {
            showMessageDialog(null,
                "Invalid Integers!",
                null, WARNING_MESSAGE);
            ICPassport_tf.setText("");
            ICPassport_tf.requestFocus();
        }
    }
}

```

```

        }
    });
btnCreateAnAccountPT.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
btnCreateAnAccountPT.setBounds(130, 344, 137, 40);

patientSignUp_pnl.add(btnCreateAnAccountPT);
getRootPane().setDefaultButton(btnCreateAnAccountPT);

JPanel administrator_pnl = new JPanel();
administrator_pnl.setBounds(320, 0, 140, 40);
signUp_pnl.add(administrator_pnl);
administrator_pnl.addMouseListener(
    new PanelButtonMouseAdapter(administrator_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            signUpMenuClicked(signUpAdmin_pnl);
            userClicked(administratorSel_pnl);
        }
    });
administrator_pnl.setLayout(null);
administrator_pnl.setBorder(null);
administrator_pnl.setBackground(new Color(79, 189, 194));

JLabel lblAdministrator = new JLabel("Administrator");
lblAdministrator.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
lblAdministrator.setHorizontalAlignment(CENTER);
lblAdministrator.setForeground(WHITE);
lblAdministrator.setBorder(null);
lblAdministrator.setBackground(new Color(79, 189, 194));
lblAdministrator.setBounds(6, 6, 128, 28);
administrator_pnl.add(lblAdministrator);

administratorSel_pnl = new JPanel();
administratorSel_pnl.setBackground(new Color(253, 210, 155));
administratorSel_pnl.setBounds(0, 0, 140, 3);
administrator_pnl.add(administratorSel_pnl);

JPanel Patient_pnl = new JPanel();
Patient_pnl.setBounds(460, 0, 140, 40);
signUp_pnl.add(Patient_pnl);
Patient_pnl.addMouseListener(
    new PanelButtonMouseAdapter(Patient_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            signUpMenuClicked(patientSignUp_pnl);
            userClicked(patientSel_pnl);
        }
    });
Patient_pnl.setLayout(null);
Patient_pnl.setBorder(null);
Patient_pnl.setBackground(new Color(79, 189, 194));

JLabel lblPatient = new JLabel("Patient");
lblPatient.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));

```

```

lblPatient.setHorizontalAlignment(CENTER);
lblPatient.setForeground(WHITE);
lblPatient.setBorder(null);
lblPatient.setBackground(new Color(79, 189, 194));
lblPatient.setBounds(6, 6, 128, 28);
Patient_pnl.add(lblPatient);

patientSel_pnl = new JPanel();
patientSel_pnl.setBackground(new Color(253, 210, 155));
patientSel_pnl.setBounds(0, 0, 140, 3);
Patient_pnl.add(patientSel_pnl);

JLabel healthcareCentres_lbl = new JLabel("");
healthcareCentres_lbl.setForeground(GRAY);
healthcareCentres_lbl.setHorizontalTextPosition(LEFT);
healthcareCentres_lbl.setFont(
    new Font(".AppleSystemUIFont", PLAIN, 12));
healthcareCentres_lbl.setBounds(130, 53, 395, 18);
signUpAdmin_pnl.add(healthcareCentres_lbl);

balimed_rdbtn = new JRadioButton("Balimed");
balimed_rdbtn.setForeground(DARK_GRAY);
balimed_rdbtn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 14));
balimed_rdbtn.addItemListener(e -> {
    // Show different healthcare centre
    // when JRadioButton is clicked.
    if (balimed_rdbtn.isSelected()) {
        healthcareCentres_lbl.setText(
            pcvs.getPCVSHealthcareCentres().get(0).toString());
    } else
        healthcareCentres_lbl.setText("");
});
balimed_rdbtn.setSelected(true);
ButtonGroup centreNameBtnGrp = new ButtonGroup();
centreNameBtnGrp.add(balimed_rdbtn);
balimed_rdbtn.setBounds(130, 16, 94, 23);
signUpAdmin_pnl.add(balimed_rdbtn);

JRadioButton primaMedika_rdbtn = new JRadioButton("Prima Medika");
primaMedika_rdbtn.setForeground(DARK_GRAY);
primaMedika_rdbtn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 14));
primaMedika_rdbtn.addItemListener(e -> {
    // Show different healthcare centre
    // when JRadioButton is clicked.
    if (primaMedika_rdbtn.isSelected()) {
        healthcareCentres_lbl.setText(
            pcvs.getPCVSHealthcareCentres().get(1).toString());
    } else
        healthcareCentres_lbl.setText("");
});
centreNameBtnGrp.add(primaMedika_rdbtn);
primaMedika_rdbtn.setBounds(265, 16, 127, 23);
signUpAdmin_pnl.add(primaMedika_rdbtn);

JButton createAccount_btn = new JButton("Create an Account");

```

```

createAccount_btn.addActionListener(e -> {
    String usrname = usernameAdmin_tf.getText().trim();
    String pwd = passAdmin_pf.getText().trim();
    String eml = emailAdmin_tf.getText().trim();
    String fName = fullNameAdmin_tf.getText().trim();
    iHealthCentre = balimed_rdbtn.isSelected() ? 0 : 1;
    if (usrname.isEmpty() || pwd.isEmpty()
        || eml.isEmpty() || fName.isEmpty()) {
        showMessageDialog(PCVSGUI.this,
            "Please fill in all fields", null,
            WARNING_MESSAGE);
        usernameAdmin_tf.requestFocus();
    } else {

        // Validate username to avoid duplicate username.
        if (pcvs.usernameValidation(usrname)) {
            showMessageDialog(PCVSGUI.this,
                "That username is taken! " +
                "Try another username",
                null, WARNING_MESSAGE);
            usernameAdmin_tf.requestFocus();
        } else {
            administrator = new Administrator(usrname, pwd,
                eml, fName, generateStaffID());

            // Save administrator object to pcvs
            pcvs.getPCVSHealthcareCentres().get(iHealthCentre)
                .setAdministrator(administrator);
            pcvs.addUser(administrator);

            showMessageDialog(PCVSGUI.this,
                "Sign up success! A Healthcare " +
                "Administrator account is created\n\n"
                + administrator.toString(),
                null, PLAIN_MESSAGE);
            clearText();
        }
    }
});
createAccount_btn.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
createAccount_btn.setBounds(130, 329, 137, 40);
signUpAdmin_pnl.add(createAccount_btn);

userClicked(administratorSel_pnl);
signUpMenuClicked(signUpAdmin_pnl);
}

<**
 * Launch the main page of pcvs application. 31
 */
* @param args not used.
*/
public static void main(String[] args) {
    EventQueue.invokeLater(() -> {
        try {
            PCVSGUI frm = new PCVSGUI(); 
        }
    });
}

```

```

        frm.setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
});

}

/**
 * Calculates this length of the Administrator ID.
 * Returns two positive integers + id, zero , or one integer + id.
 *
 * @return a String with the increment of id.
 */
public static String generateStaffID() {
    // if length id 1 digit add with 00 + id
    String id = ((admID.toString().length() == 1) ? ("00" + admID)
        // if length id 2 digit add with 0 + id
        : ((admID.toString().length() == 2) ? ("0" + admID)
        : admID.toString());
    admID++; // increment
    return "ADM" + id;
}

/**
 * Assign specific actions to menu item components
 * for managing files in pcvs applications.
 *
 * @param evt determine the object where the event occurs.
22
@Override
public void actionPerformed(ActionEvent evt) {
    if (evt.getSource() == open_mntm || evt.getSource() == new_mntm
        || evt.getSource() == exit_mntm) {
        int choice = showConfirmDialog(this,
            "Do you want to save the changes made " +
            "to the file?",
            "Warning", YES_NO_CANCEL_OPTION,
            WARNING_MESSAGE);

        if (choice == YES_OPTION) {
            // Existing directory file.
            if (fileDir != null)
                saveFile();
            else
                showFileChooser();
        } else if (choice == NO_OPTION) {
            int retVal = f1Chooser.showOpenDialog(this);
            if (retVal == APPROVE_OPTION) {
                fileDir = f1Chooser.getSelectedFile();

                // Read this file.
                openFile();

                showMessageDialog(PCVSGUI.this,
                    "File opened: " + fileDir.getName()
                    + " success!",
                    null, PLAIN_MESSAGE);
            }
        }
    }
}

```

```

        }
    }
} else if (evt.getSource() == saveAs_mntm ||
    (evt.getSource() == save_mntm && fileDir == null)) {
    showFileChooser();
} else if (evt.getSource() == save_mntm && fileDir != null) {
    saveFile();
    showMessageDialog(PCVSGUI.this,
        "Saving: " + fileDir.getName(), null,
        PLAIN_MESSAGE);
}
}

< /**
 * Save the Serializable object into a file
 * based on the file directory selected by the user.
 */
public void saveFile() {
    FileOutputStream fileoutStrm;
    try {
        fileoutStrm = new FileOutputStream(fileDir);
        ObjectOutputStream objoutStr =
            new ObjectOutputStream(fileoutStrm);
        objoutStr.writeObject(pcvs);
        objoutStr.flush();
        objoutStr.close();
    } catch (IOException exc) {
        showMessageDialog(PCVSGUI.this,
            "Save file failed", null,
            WARNING_MESSAGE);
    }
}

< /**
 * Reading data in user-selected directory files.
 */
public void openFile() {
    FileInputStream fileInStrm;
    try {
        fileInStrm = new FileInputStream(fileDir);
        ObjectInputStream objInStrm =
            new ObjectInputStream(fileInStrm);
        try {
            pcvs = (PCVS) objInStrm.readObject();
        } catch (IOException | ClassNotFoundException exc) {
            exc.printStackTrace();
        }

        objInStrm.close();
    } catch (Exception e) {
        showMessageDialog(PCVSGUI.this,
            "Open file failed", null,
            WARNING_MESSAGE);
    }
}

/*

```

```

        * Display the file directory to the user.
    */
    public void showFileChooser() {
        int retVal = f1Chooser.showSaveDialog(this);
        if (retVal == APPROVE_OPTION) {
            fileDir = f1Chooser.getSelectedFile();
            saveFile();
            showMessageDialog(PCVSGUI.this,
                "Saving: " + fileDir.getName(), null,
                PLAIN_MESSAGE);
        } else
            showMessageDialog(PCVSGUI.this,
                "Save cancelled", null,
                WARNING_MESSAGE);
    }

    /**
     * Remove the field when the user
     * has successfully entered the information.
     */
    private void clearText() {
        usernameAdmin_tf.setText("");
        passAdmin_pf.setText("");
        emailAdmin_tf.setText("");
        fullNameAdmin_tf.setText("");
        balimed_rdbtn.setSelected(true);
        usernamePatient_tf.setText("");
        passPatient_pf.setText("");
        emailPatient_tf.setText("");
        fullNamePatient_tf.setText("");
        ICPassport_tf.setText("");
    }

    /**
     * Shows the sign-up menu between administrator and patient,
     * hiding it when not selected by the user.
     *
     * @param panel user-selected menu.
     */
    public void signUpMenuClicked(JPanel panel) {
        signUpAdmin_pnl.setVisible(false);
        patientSignUp_pnl.setVisible(false);
        panel.setVisible(true);
    }

    /**
     * Show highlights on user-selected menu.
     *
     * @param panel user-selected menu.
     */
    public void userClicked(JPanel panel) {
        administratorSel_pnl.setOpaque(false);
        patientSel_pnl.setOpaque(false);
        panel.setOpaque(true);
    }

}

```

```

* Registers this pcvs object to another pcvs attribute
* in another JFrame.
* This method will save the current pcvs object
* to avoid create new data of pcvs.
*
* @param pcvsDate the current pcvs object to save.
*/
public void setPCVS(PCVS pcvsDate) {
    pcvs = pcvsDate;
}

/**
* Registers this file directory to another
* file attribute in another JFrame.
* This method will save the current file directory
* to avoid current file lost.
*
* @param directory the directory file to save.
*/
public void setFileDir(File directory) {
    fileDir = directory;
}

/**
* PanelButtonMouseAdapter inner class to display a different color
* when the mouse entered the panel.
*/
private static class PanelButtonMouseAdapter extends MouseAdapter {
    JPanel panel;

    /**
     * PanelButtonMouseAdapter constructor for invocation in listeners.
     *
     * @param panel determine which panel receives the action.
     */
    public PanelButtonMouseAdapter(JPanel panel) {
        this.panel = panel;
    }

    /**
     * Change this color when a mouse button
     * has been entered on a component. 5
     *
     * @param e the event to process.
     */
    @Override
    public void mouseEntered(MouseEvent e) {
        panel.setBackground(new Color(209, 244, 240));
    }

    /**
     * Change this color when a mouse button
     * has been exited on a component. 5
     *
     * @param e the event to process.
     */
    @Override

```

```
public void mouseExited(MouseEvent e) {
    panel.setBackground(new Color(79, 189, 194));
}

/**
 * Change this color when a mouse button
 * has been pressed on a component.
 *
 * @param e the event to process.
14
@Override
public void mousePressed(MouseEvent e) {
    panel.setBackground(new Color(79, 189, 194));
}

/**
 * Change this color when mouse button
 * has been released.
 *
 * @param e the event to process.
6/
@Override
public void mouseReleased(MouseEvent e) {
    panel.setBackground(new Color(209, 244, 240));
}
}
```

LoginDialog

```
package dialog;

import frame.SignUpGUI;
3
import javax.swing.border.EmptyBorder;
import javax.swing.*;
import javax.swing.border.LineBorder;
import java.awt.event.WindowAdapter;
import java.awt.*;
import java.awt.event.WindowEvent;

import static java.awt.BorderLayout.*;
import static java.awt.Color.*;
import static java.awt.Font.*;

public class LoginDialog extends JDialog {

    private final JTextField usernameUser_tf;
    private final JPasswordField passUser_pf;

    private String usernameLogin;
    private String passwordLogin;

    /**
     * LoginDialog class allow pcvs user login to the application
     * based on valid account.
     *
     * @author I Nyoman Surya Pradipta
     * Student ID: E1900344
     * Date: 01-06-2021
     * Java version: java 17 2021-09-14 LTS
     * IDE : IntelliJ IDEA & Eclipse
     */
    public LoginDialog(JFrame parent) {
        super(parent, true);
        setTitle("Log in");
        setBounds(0, 0, 1000, 600);
        getContentPane().setLayout(new BorderLayout());
        JPanel content_pnl = new JPanel();
        content_pnl.setLayout(new EmptyBorder(5, 5, 5, 5));
        getContentPane().add(content_pnl, CENTER);
        content_pnl.setLayout(null);

        {
            JPanel logo_pnl = new JPanel();
            logo_pnl.setBounds(0, 0, 400, 533);
            content_pnl.add(logo_pnl);
            logo_pnl.setLayout(null);

            JLabel logo_lbl = new JLabel("");
            logo_lbl.setIcon(new ImageIcon(SignUpGUI.class
                .getResource("/res/signUp.png")));
            logo_lbl.setBounds(0, 0, 400, 533);
            logo_pnl.add(logo_lbl);
        }
    }
}
```

```

}
{
    JPanel signIn_pnl = new JPanel();
    signIn_pnl.setBackground(WHITE);
    signIn_pnl.setBounds(400, 0, 600, 533);
    content_pnl.add(signIn_pnl);
    signIn_pnl.setLayout(null);
    {
        JLabel title_lbl = new JLabel("Sign in to PCVS ");
        title_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
            PLAIN, 18));
        title_lbl.setBounds(155, 145, 179, 40);
        signIn_pnl.add(title_lbl);
        title_lbl.setForeground(DARK_GRAY);
    }
    {
        JLabel desc_lbl = new JLabel(
            "Use your Administrator or Patient account");
        desc_lbl.setFont(new Font(".AppleSystemUIFont",
            PLAIN, 13));
        desc_lbl.setForeground(GRAY);
        desc_lbl.setBounds(153, 197, 276, 16);
        signIn_pnl.add(desc_lbl);
    }
    {
        16
        JLabel username_lbl = new JLabel("Username");
        username_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
            PLAIN, 14));
        username_lbl.setBounds(155, 225, 94, 18);
        signIn_pnl.add(username_lbl);
    }
    {
        JPanel username_pnl = new JPanel();
        username_pnl.setLayout(null);
        username_pnl.setOpaque(false);
        username_pnl.setBorder(new LineBorder(
            new Color(238, 238, 238), 2, true));
        username_pnl.setBounds(155, 255, 262, 40);
        signIn_pnl.add(username_pnl);
        {
            usernameUser_tf = new JTextField();
            usernameUser_tf.setForeground(DARK_GRAY);
            usernameUser_tf.setFont(new Font(".AppleSystemUIFont",
                PLAIN, 13));
            usernameUser_tf.setColumns(10);
            usernameUser_tf.setBorder(null);
            usernameUser_tf.setBackground(WHITE);
            usernameUser_tf.setBounds(6, 6, 250, 28);
            username_pnl.add(usernameUser_tf);
        }
    }
    {
        JLabel pass_lbl = new JLabel("Password");
        pass_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
            PLAIN, 14));
        pass_lbl.setBounds(155, 307, 94, 18);
        signIn_pnl.add(pass_lbl);
    }
}

```

```

}
{
    JPanel pass_pnl = new JPanel();
    pass_pnl.setLayout(null);
    pass_pnl.setOpaque(false);
    pass_pnl.setBorder(new LineBorder(
        new Color(238, 238, 238), 2, true));
    pass_pnl.setBounds(155, 337, 262, 40);
    signIn_pnl.add(pass_pnl);
    {
        passUser_pf = new JPasswordField();
        passUser_pf.setForeground(DARK_GRAY);
        passUser_pf.setFont(new Font(".AppleSystemUIFont",
            PLAIN, 9));
        passUser_pf.setBorder(null);
        passUser_pf.setBounds(6, 6, 250, 28);
        pass_pnl.add(passUser_pf);
    }
}
{
    JButton signIn_btn = new JButton("Sign In");
    signIn_btn.setFont(new Font("Neue Haas Grotesk Text Pro",
        PLAIN, 14));
    signIn_btn.setBounds(155, 389, 130, 40);
    signIn_pnl.add(signIn_btn); ①
    signIn_btn.addActionListener(e -> {
        String username = usernameUser_tf.getText().trim();
        String password = passUser_pf.getText().trim();

        if (username.equals("") || password.equals("")) {
            JOptionPane.showMessageDialog(LoginDialog.this,
                "Please fill up all fields", null,
                JOptionPane.WARNING_MESSAGE);
            usernameUser_tf.requestFocus();
        } else {
            clearText();
            usernameLogin = username;
            passwordLogin = password;
            setVisible(false);
        }
    });
    getRootPane().setDefaultButton(signIn_btn);
}
addWindowListener(new WindowAdapter() {
    // anonymous class if user chooses to close dialog
    public void windowClosing(WindowEvent we) {
        usernameLogin = null;
        passwordLogin = null;
    }
}); ②
{
    JPanel btn_pnl = new JPanel();
    btn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));
    getContentPane().add(btn_pnl, SOUTH);
    {
        JButton cancel_btn = new JButton("Cancel");

```

```
    cancel_btn.addActionListener(e -> setVisible(false));
    {
        JButton clear_btn = new JButton("Clear");
        clear_btn.addActionListener(e -> clearText());
        btn_pnl.add(clear_btn);
    }
    btn_pnl.add(cancel_btn);
}
}

/**
 * Remove the field when the user
 * has successfully entered the information.
 */
private void clearText() {
    usernameUser_tf.setText("");
    passUser_pf.setText("");
}

/**
 * Returns this username of User input
 * to validate in SignUpGUI.
 * If username and password valid, pcvs GUI application
 * allow User access the menu.
 *
 * @return User account is username.
 */
public String getUsernameAcc() {
    return usernameLogin;
}

/**
 * Returns this password of User input
 * to validate in SignUpGUI.
 * If username and password valid, pcvs GUI application
 * allow User access the menu.
 *
 * @return User account is password.
 */
public String getPassAcc() {
    return passwordLogin;
}
}
```

AdminMenuGUI

```
package frame;

import pcvs.Batch;
import pcvs.PCVS;

12 import javax.swing.border.EmptyBorder;
import javax.swing.*;
import javax.swing.border.LineBorder;

import java.awt.event.MouseAdapter;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.io.File;

import static java.awt.Color.*;
import static java.awt.Font.*;
import static javax.swing.JFrame.*;
import static javax.swing.JOptionPane.*;

/**
 * AdminMenuGUI class to allow Administrator
 * access menu pcvs GUI application.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
23/
public class AdminMenuGUI extends JFrame {

    private final JTextField batchNum_tf;
    private final JTextField expDate_tf;
    private final JTextField qtyAvailable_tf;
    JRadioButton jnj_rdbtn;
    private PCVS pcvs;
    private File file = null;
    private int iAdmin;
    private Batch batch;

    /**
     * Class Constructor to create
     * admin menu for administrator.
     */
    public AdminMenuGUI() {
        7 setDefaultCloseOperation(EXIT_ON_CLOSE);
        setBounds(250, 150, 1000, 600);
        JPanel conte2t_pnl = new JPanel();
        content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(content_pnl);
        content_pnl.setLayout(null);
    }
}
```

```

pcvs = new PCVS();

JPanel adminMenu_pnl = new JPanel();
adminMenu_pnl.setBounds(0, 0, 1000, 572);
content_pnl.add(adminMenu_pnl);
adminMenu_pnl.setLayout(null);

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
adminMenu_pnl.add(sideBarAdminMenu_pnl);
sideBarAdminMenu_pnl.setLayout(null);

JPanel recordBatch_pnl = new JPanel();
recordBatch_pnl.setLayout(null);
recordBatch_pnl.setBackground(new Color(79, 189, 194));
recordBatch_pnl.setBounds(0, 155, 300, 40);
sideBarAdminMenu_pnl.add(recordBatch_pnl);

JLabel recordBatch_lbl = new JLabel("Record Vaccine Batch");
recordBatch_lbl.setForeground(WHITE);
recordBatch_lbl.setFont(
    new Font(".AppleSystemUIFont", PLAIN, 16));
recordBatch_lbl.setBounds(36, 6, 258, 28);
recordBatch_pnl.add(recordBatch_lbl);

JPanel recordBatchSel_panel = new JPanel();
recordBatchSel_panel.setBackground(new Color(253, 210, 155));
recordBatchSel_panel.setBounds(0, 0, 5, 40);
recordBatch_pnl.add(recordBatchSel_panel);

JPanel viewBatch_pnl = new JPanel();
viewBatch_pnl.addMouseListener(
    new PanelMenuButtonMouseListener(viewBatch_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            VaccineBatchGUI vaccineBatch =
                new VaccineBatchGUI();
            vaccineBatch.pack();
            vaccineBatch.setBounds(0, 0, 1000, 600);
            vaccineBatch
                .setLocationRelativeTo(AdminMenuGUI.this);
            vaccineBatch.setPCVS(pcvs);
            vaccineBatch.setFile(file);
            vaccineBatch.setIndexAdmin(iAdmin);
            vaccineBatch.setVisible(true);
            setVisible(false);
        }
    });
viewBatch_pnl.setLayout(null);
viewBatch_pnl.setBackground(new Color(40, 143, 148));
viewBatch_pnl.setBounds(0, 195, 300, 40);
sideBarAdminMenu_pnl.add(viewBatch_pnl);

JLabel viewBatch_lbl = new JLabel("View Vaccine Batch");
viewBatch_lbl.setForeground(WHITE);

```

```

viewBatch_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
viewBatch_lbl.setBounds(36, 6, 258, 28);
viewBatch_pnl.add(viewBatch_lbl);

JPanel adminInfo_pnl = new JPanel();
adminInfo_pnl.addMouseListener(
    new PanelMenuButtonMouseListener(adminInfo_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            AdministratorInformationGUI adminInf =
                new AdministratorInformationGUI();
            adminInf.pack();
            adminInf.setBounds(0, 0, 1000, 600);
            adminInf.setLocationRelativeTo(AdminMenuGUI.this);
            adminInf.setPCVS(pcvs);
            adminInf.setFile(file);
            adminInf.setVisible(true);
            setVisible(false);
        }
    });
adminInfo_pnl.setLayout(null);
adminInfo_pnl.setBackground(new Color(40, 143, 148));
adminInfo_pnl.setBounds(0, 235, 300, 40);
sideBarAdminMenu_pnl.add(adminInfo_pnl);

JLabel adminInfo_lbl = new JLabel("Administrator Information");
adminInfo_lbl.setForeground(WHITE);
adminInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
adminInfo_lbl.setBounds(36, 6, 258, 28);
adminInfo_pnl.add(adminInfo_lbl);

JPanel patientInfo_pnl = new JPanel();
patientInfo_pnl.addMouseListener(
    new PanelMenuButtonMouseListener(patientInfo_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            PatientInformationGUI patientInf =
                new PatientInformationGUI();
            patientInf.pack();
            patientInf.setBounds(0, 0, 1000, 600);
            patientInf
                .setLocationRelativeTo(AdminMenuGUI.this);
            patientInf.setPCVS(pcvs);
            patientInf.setFile(file);

            patientInf.setVisible(true);
            setVisible(false);
        }
    });
patientInfo_pnl.setLayout(null);
patientInfo_pnl.setBackground(new Color(40, 143, 148));
patientInfo_pnl.setBounds(0, 275, 300, 40);
sideBarAdminMenu_pnl.add(patientInfo_pnl);

JLabel patientInfo_lbl = new JLabel("Patient Information");

```

```

patientInfo_lbl.setForeground(WHITE);
patientInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
patientInfo_lbl.setBounds(36, 6, 258, 28);
patientInfo_pnl.add(patientInfo_lbl);

JPanel vaccinationInfo_pnl = new JPanel();
vaccinationInfo_pnl.addMouseListener(
    new PanelMenuButtonMouseAdapter(vaccinationInfo_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            VaccinationInformationGUI vaccinationInfo =
                new VaccinationInformationGUI();
            vaccinationInfo.pack();
            vaccinationInfo.setBounds(0, 0, 1000, 600);
            vaccinationInfo
                .setLocationRelativeTo(AdminMenuGUI.this);
            vaccinationInfo.setPCVS(pcv);
            vaccinationInfo.setFile(file);
            vaccinationInfo.setVisible(true);
            setVisible(false);
        }
    });
vaccinationInfo_pnl.setLayout(null);
vaccinationInfo_pnl.setBackground(new Color(40, 143, 148));
vaccinationInfo_pnl.setBounds(0, 315, 300, 40);
sideBarAdminMenu_pnl.add(vaccinationInfo_pnl);

JLabel vaccinationInfo_lbl = new JLabel("Vaccination Information");
vaccinationInfo_lbl.setForeground(WHITE);
vaccinationInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
vaccinationInfo_lbl.setBounds(36, 6, 258, 28);
vaccinationInfo_pnl.add(vaccinationInfo_lbl);

JButton logOut_btn = new JButton("Log Out");
logOut_btn.addActionListener(e -> {
    int choice = showConfirmDialog(AdminMenuGUI.this,
        "Are you su10 want to log out?",
        "Warning", YES_NO_CANCEL_OPTION,
        WARNING_MESSAGE);
    if (choice == YES_OPTION) {
        PCVSGUI pcv_gui = new PCVSGUI();
        pcv_gui.pack();
        pcv_gui.setBounds(0, 0, 1000, 600);
        pcv_gui.setLocationRelativeTo(AdminMenuGUI.this);
        pcv_gui.setPCVS(pcv);
        pcv_gui.setFileDir(file);
        pcv_gui.setVisible(true);
        setVisible(false);
    }
});
logOut_btn.setForeground(DARK_GRAY);
logOut_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
logOut_btn.setBounds(6, 475, 288, 40);

```

```

sideBarAdminMenu_pnl.add(logOut_btn);

JPanel adminContent_pnl = new JPanel();
adminContent_pnl.setBounds(300, 0, 700, 572);
adminMenu_pnl.add(adminContent_pnl);
adminContent_pnl.setLayout(null);

JPanel recordBatchFrame_pnl = new JPanel();
recordBatchFrame_pnl.setBackground(WHITE);
recordBatchFrame_pnl.setBounds(0, 0, 700, 572);
adminContent_pnl.add(recordBatchFrame_pnl);
recordBatchFrame_pnl.setLayout(null);

JLabel vacIDSel_lbl = new JLabel("Select Vaccine ID");
vacIDSel_lbl.setBounds(219, 117, 123, 18);
recordBatchFrame_pnl.add(vacIDSel_lbl);
vacIDSel_lbl.setToolTipText("");
vacIDSel_lbl.setForeground(DARK_GRAY);
vacIDSel_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 14));

JLabel vaccine_lbl = new JLabel("");
vaccine_lbl.setHorizontalAlignment(SwingConstants.CENTER);
vaccine_lbl.setToolTipText("");
vaccine_lbl.setForeground(DARK_GRAY);
vaccine_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 12));
vaccine_lbl.setBounds(107, 182, 486, 18);
recordBatchFrame_pnl.add(vaccine_lbl);

jnj_rdbtn = new JRadioButton("JNJ");
jnj_rdbtn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
jnj_rdbtn.addItemListener(e -> {
    if (jnj_rdbtn.isSelected()) {
        vaccine_lbl.setText(pcvs.getPCVSVaccines().get(0)
            .toString());
    } else
        vaccine_lbl.setText("");
});
jnj_rdbtn.setSelected(true);
ButtonGroup vaccineIDBtn = new ButtonGroup();
vaccineIDBtn.add(jnj_rdbtn);
jnj_rdbtn.setForeground(DARK_GRAY);
jnj_rdbtn.setBounds(219, 147, 64, 23);
recordBatchFrame_pnl.add(jnj_rdbtn);

JRadioButton asz_rdbtn = new JRadioButton("ASZ");
asz_rdbtn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
asz_rdbtn.addItemListener(e -> {
    if (asz_rdbtn.isSelected()) {
        vaccine_lbl.setText(pcvs.getPCVSVaccines().get(1)
            .toString());
    } else
        vaccine_lbl.setText("");
});

```

```
vaccineIDBtn.add(asz_rdbtn);
asz_rdbtn.setForeground(DARK_GRAY);
asz_rdbtn.setBounds(417, 147, 64, 23);
recordBatchFrame_pnl.add(asz_rdbtn);

JLabel batchNum_lbl = new JLabel("Batch Number");
batchNum_lbl.setToolTipText("");
batchNum_lbl.setForeground(DARK_GRAY);
batchNum_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 14));
batchNum_lbl.setBounds(219, 223, 97, 18);
recordBatchFrame_pnl.add(batchNum_lbl);

JPanel batchNum_pnl = new JPanel();
batchNum_pnl.setLayout(null);
batchNum_pnl.setOpaque(false);
batchNum_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
batchNum_pnl.setBounds(219, 253, 262, 40);
recordBatchFrame_pnl.add(batchNum_pnl);

batchNum_tf = new JTextField();
batchNum_tf.setForeground(DARK_GRAY);
batchNum_tf.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
batchNum_tf.setColumns(10);
batchNum_tf.setBorder(null);
batchNum_tf.setBackground(WHITE);
batchNum_tf.setBounds(6, 6, 250, 28);
batchNum_pnl.add(batchNum_tf);

JLabel expDate_lbl = new JLabel("Expiry Date (mm dd yyyy)");
expDate_lbl.setToolTipText("");
expDate_lbl.setForeground(DARK_GRAY);
expDate_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 14));
expDate_lbl.setBounds(219, 305, 173, 18);
recordBatchFrame_pnl.add(expDate_lbl);

JPanel expDate_pnl = new JPanel();
expDate_pnl.setLayout(null);
expDate_pnl.setOpaque(false);
expDate_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
expDate_pnl.setBounds(219, 335, 262, 40);
recordBatchFrame_pnl.add(expDate_pnl);

expDate_tf = new JTextField();
expDate_tf.setForeground(DARK_GRAY);
expDate_tf.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
expDate_tf.setColumns(10);
expDate_tf.setBorder(null);
expDate_tf.setBackground(WHITE);
expDate_tf.setBounds(6, 6, 250, 28);
expDate_pnl.add(expDate_tf);
```

```

JLabel qtyAvailable_lbl = new JLabel("Quantity Available");
qtyAvailable_lbl.setToolTipText("");
qtyAvailable_lbl.setForeground(DARK_GRAY);
qtyAvailable_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 14));
qtyAvailable_lbl.setBounds(219, 387, 123, 18);
recordBatchFrame_pnl.add(qtyAvailable_lbl);

JPanel qtyAvailable_pnl = new JPanel();
qtyAvailable_pnl.setLayout(null);
qtyAvailable_pnl.setOpaque(false);
qtyAvailable_pnl.setBorder(new LineBorder(
    new Color(238, 238, 238), 2, true));
qtyAvailable_pnl.setBounds(219, 417, 262, 40);
recordBatchFrame_pnl.add(qtyAvailable_pnl);

qtyAvailable_tf = new JTextField();
qtyAvailable_tf.setForeground(DARK_GRAY);
qtyAvailable_tf.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
qtyAvailable_tf.setColumns(10);
qtyAvailable_tf.setBorder(null);
qtyAvailable_tf.setBackground(WHITE);
qtyAvailable_tf.setBounds(6, 6, 250, 28);
qtyAvailable_pnl.add(qtyAvailable_tf);

JButton record_btn = new JButton("Record");
record_btn.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 13));
record_btn.addActionListener(e -> {
    String batchNorStr = batchNum_tf.getText().trim();
    String expiryDate = expDate_tf.getText().trim();
    String qtyAvailableStr = qtyAvailable_tf.getText().trim();
    int iVaccine = jnj_rdbtn.isSelected() ? 0 : 1;

    if (batchNorStr.isEmpty() || expiryDate.isEmpty()
        || qtyAvailableStr.isEmpty()) {
        showMessageDialog(AdminMenuGUI.this,
            "Please fill up all fields", null,
            WARNING_MESSAGE);
        batchNum_tf.requestFocus();
    } else {
        int batchNo = Integer.parseInt(batchNorStr);
        int qtyAvailbale = Integer.parseInt(qtyAvailableStr);
        clearText();
        batch = new Batch(batchNo, expiryDate, qtyAvailbale, 0);

        showMessageDialog(AdminMenuGUI.this,
            "The batch is recorded for the vaccine " +
            "and healthcare centre\n\n" + batch,
            null, PLAIN_MESSAGE);

        pcvs.getPCVSVaccines().get(iVaccine).setBatch(batch);
        pcvs.getPCVSHealthcareCentres().get(iAdmin)
            .setBatches(batch);
    }
});

```

```

        record_btn.setBounds(219, 469, 130, 40);
        recordBatchFrame_pnl.add(record_btn);

        JPanel recordBatchTitle_pnl = new JPanel();
        recordBatchTitle_pnl.setBackground(PINK);
        recordBatchTitle_pnl.setBounds(0, 0, 700, 70);
        recordBatchFrame_pnl.add(recordBatchTitle_pnl);
        recordBatchTitle_pnl.setLayout(null);

        JLabel recordBatchTitle_lbl = new JLabel("Record Vaccine Batch");
        recordBatchTitle_lbl.setHorizontalTextPosition(SwingConstants.CENTER);
        recordBatchTitle_lbl.setForeground(WHITE);
        recordBatchTitle_lbl.setFont(
            new Font("Neue Haas Grotesk Text Pro", PLAIN, 20));
        recordBatchTitle_lbl.setBounds(6, 6, 688, 58);
        recordBatchTitle_pnl.add(recordBatchTitle_lbl);

    }

    /**
     * Remove the field when the user
     * has successfully entered the information.
     */
    private void clearText() {
        batchNum_tf.setText("");
        expDate_tf.setText("");
        qtyAvailable_tf.setText("");
        jnj_rdbtn.setSelected(true);
    }

    /**
     * Registers this pcvs object to another pcvs attribute
     * in another JFrame.
     * This method will save the current pcvs object
     * to avoid create new data of pcvs.
     *
     * @param pcvsObj the current pcvs object to save.
     */
    public void setPCVS(PCVS pcvsObj) {
        pcvs = pcvsObj;
    }

    /**
     * Registers this file directory to another
     * file attribute in another JFrame.
     * This method will save the current file directory
     * to avoid current file lost.
     *
     * @param directory the directory file to save.
     */
    public void setFile(File directory) {
        file = directory;
    }

    /**
     * Registers this index admin to another
     * index admin attribute in another JFrame

```

```

        * to avoid lost value.
        *
        * @param idx the value to save.
        */
    public void setIndexAdmin(int idx) {
        iAdmin = idx;
    }

    /**
     * PanelButtonMouseAdapter inner class to display a different color
     * when the mouse entered the panel.
     */
    private static class PanelMenuButtonMouseAdapter extends MouseAdapter {
        JPanel panel;

        /**
         * PanelButtonMouseAdapter constructor for invocation in listeners.
         *
         * @param panel determine which panel receives the action.
         */
        public PanelMenuButtonMouseAdapter(JPanel panel) {
            this.panel = panel;
        }

        /**
         * Change this color when a mouse button
         * has been entered on a component. 5
         *
         * @param e the event to process.
6
@Override
public void mouseEntered(MouseEvent e) {
    panel.setBackground(new Color(79, 189, 194));
}

/**
    * Change this color when a mouse button
5
has been exited on a component.
*
* @param e the event to process.
6
@Override
public void mouseExited(MouseEvent e) {
    panel.setBackground(new Color(40, 143, 148));
}

/**
    * Change this color when a mouse button
5
has been pressed on a component.
*
* @param e the event to process.
14
@Override
public void mousePressed(MouseEvent e) {
    panel.setBackground(new Color(40, 143, 148));
}

```

```
/**  
 * Change this color when mouse button  
 * has been released.  
 *  
 * @param e the event to process.  
 */  
@Override  
public void mouseReleased(MouseEvent e) {  
    panel.setBackground(new Color(79, 189, 194));  
}  
}  
}
```

VaccineBatchGUI

```
package frame;
1
import pcvs.Batch;
import pcvs.HealthcareCentre;
import pcvs.PCVS;
import pcvs.Vaccine;
import table.VaccineBatchTableModel;

import javax.swing.border.EmptyBorder;
import java.awt.*;
import javax.swing.*;
import java.awt.event.WindowEvent;
import java.io.File;
import java.awt.event.WindowAdapter;
import java.util.stream.IntStream;
29
import static java.awt.Color.PINK;
import static java.awt.Color.WHITE;
import static java.awt.Font.PLAIN;
import static javax.swing.JOptionPane.*;
import static javax.swing.SwingConstants.CENTER;

/**
 * VaccineBatchGUI class to allow healthcare administrator to
 * list the available batch of vaccines.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class VaccineBatchGUI extends JFrame {

    // Global component used on methods or listeners.
    private final JPanel content_pnl;
    private final VaccineBatchTableModel batchesTableModel;
    private JTable vaccBatchInfo_table;
    // Global attribute used on methods or listeners.
    private PCVS pcvs;
    private File file = null;
    private int iAdmin;
    private int batchNum;

    /**
     * Create the frame.
     */
    public VaccineBatchGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display the available
             * batch of vaccine at the healthcare centres.
             * @param e the event to process.
             */
        });
    }
}
```

```

    */
@Override
public void windowOpened(WindowEvent e) {
    batchesTableModel.clear();
    HealthcareCentre hc =
        pcvs.getPCVSHCHealthcareCentres().get(iAdmin);

    // Traverse vaccines.
    for (int p = 0; p < pcvs.getPCVSVaccines().size(); p++) {
        Vaccine vaccine = pcvs.getPCVSVaccines().get(p);

        // Traverse batches in vaccines.
        for (int r = 0; r < vaccine.getBatches().size(); r++) {
            Batch batchVacc = vaccine.getBatches().get(r);

            // Traverse vaccinations in vaccines.
            for (int s = 0; s <
                batchVacc.getVaccinations().size(); s++) {

                // Traverse batches at healthcare centre.
                for (int x = 0; x < hc
                    .getBatches().size(); x++) {
                    Batch batchHC = hc.getBatches().get(x);
                    String status = batchVacc.getVaccinations()
                        .get(s).getStatus();

                    // Same batch.
                    if (batchVacc.equals(batchHC)) {
                        if (status.equals("pending") ||
                            status.equals("confirmed")) {

                            // Avoid duplicate.
                            if (!batchesTableModel
                                .contains(batchVacc)) {
                                batchesTableModel
                                    .add(batchVacc);
                            }
                        }
                    }
                }
            }
        }
    });
setDefaultCloseOperation(EXIT_ON_CLOSE);
setBounds(250, 250, 1000, 600);
content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setLayout(null);
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));

```

```

sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(sideBarAdminMenu_pnl);

JPanel vaccBatch_pnl = new JPanel();
vaccBatch_pnl.setLayout(null);
vaccBatch_pnl.setBackground(new Color(79, 189, 194));
vaccBatch_pnl.setBounds(0, 195, 300, 40);
sideBarAdminMenu_pnl.add(vaccBatch_pnl);

JLabel vaccBatch_lbl = new JLabel("View Vaccine Batch");
vaccBatch_lbl.setForeground(WHITE);
vaccBatch_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
vaccBatch_lbl.setBounds(36, 6, 258, 28);
vaccBatch_pnl.add(vaccBatch_lbl);

JPanel vaccBatchSel_pnl = new JPanel();
vaccBatchSel_pnl.setBounds(0, 0, 5, 40);
vaccBatch_pnl.add(vaccBatchSel_pnl);
vaccBatchSel_pnl.setBackground(new Color(253, 210, 155));

JPanel vaccBatchInfo_pnl = new JPanel();
vaccBatchInfo_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(vaccBatchInfo_pnl);
vaccBatchInfo_pnl.setLayout(null);

JPanel vaccBatchInfoTitle_pnl = new JPanel();
vaccBatchInfoTitle_pnl.setLayout(null);
vaccBatchInfoTitle_pnl.setBackground(PINK);
vaccBatchInfoTitle_pnl.setBounds(0, 0, 700, 70);
vaccBatchInfo_pnl.add(vaccBatchInfoTitle_pnl);

JLabel vaccBatchInfo_lbl =
    new JLabel("View Vaccine Batch Information");
vaccBatchInfo_lbl.setHorizontalAlignment(CENTER);
vaccBatchInfo_lbl.setForeground(WHITE);
vaccBatchInfo_lbl.setFont(new Font("Neue Haas Grotesk Text Pro",
    PLAIN, 20));
vaccBatchInfo_lbl.setBounds(6, 6, 688, 58);
vaccBatchInfoTitle_pnl.add(vaccBatchInfo_lbl);

JPanel vaccBatchInfoList_pnl = new JPanel();
vaccBatchInfoList_pnl.setBounds(0, 69, 700, 465);
vaccBatchInfo_pnl.add(vaccBatchInfoList_pnl);
vaccBatchInfoList_pnl.setLayout(null);

JScrollPane vaccBatchInfo_scrollPane =
    new JScrollPane(vaccBatchInfo_table);
vaccBatchInfo_scrollPane.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
vaccBatchInfo_scrollPane.setBorder(null);
vaccBatchInfo_scrollPane.setBounds(0, 0, 700, 465);
vaccBatchInfoList_pnl.add(vaccBatchInfo_scrollPane);

String[] colHeaderVaccineBatch = {"Name", "Batch Number",
    "Expiry Date", "Quantity Available"};
batchesTableModel = new VaccineBatchTableModel(

```

```

        colHeaderVaccineBatch, 0);

vaccBatchInfo_table = new JTable(batchesTableModel);
vaccBatchInfo_table.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 12));
vaccBatchInfo_table.setBorder(null);
vaccBatchInfo_scrollPane.setViewportView(vaccBatchInfo_table);

JPanel vaccBatchInfoBtn_pnl = new JPanel();
vaccBatchInfoBtn_pnl.setBackground(PINK);
vaccBatchInfoBtn_pnl.setBounds(0, 533, 700, 39);
vaccBatchInfo_pnl.add(vaccBatchInfoBtn_pnl);
vaccBatchInfoBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

 JButton sel_btn = new JButton("Select");
sel_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
sel_btn.addActionListener(e -> {
    int selRow = vaccBatchInfo_table.getSelectedRow();
    if (selRow != -1) {
        Object batchNumObject =
            vaccBatchInfo_table.getValueAt(selRow, 0);
        // Gets this batch number
        batchNum = (Integer) batchNumObject;

        // Find batch based on batch number.
pcvs.getPCVSVaccines().forEach(vaccine -> IntStream
    .range(0, vaccine.getBatches().size())
    .filter(j -> vaccine.getBatches().get(j)
        .getBatchNo() == batchNum)
    .mapToObj(j -> vaccine.getBatches().get(j))
    .forEach(batch ->
        showMessageDialog(
            VaccineBatchGUI.this,
            "Batch expiry date: "
                + batch.getExpiryDate() +
            "\nNumber of available: "
                + batch.getQuantityAvailable() +
            "\nAdministered vaccinations: "
                + batch.getQuantityAdministered() +
            "\nNumber of pending: "
                + pcvx.getNumOfPendingByBatch(
                    batchNum),
            null, PLAIN_MESSAGE)));
    }

VaccinationListGUI vaccinationList =
    new VaccinationListGUI();
vaccinationList.pack();
vaccinationList.setBounds(0, 0, 1000, 600);
vaccinationList.setLocationRelativeTo(
    VaccineBatchGUI.this);

vaccinationList.setPCVS(pcvx);
vaccinationList.setFile(file);
vaccinationList.setBatchNumber(batchNum);
vaccinationList.setIndexAdmin(iAdmin);

vaccinationList.setVisible(true);

```

```

        setVisible(false);
    } else {
        showMessageDialog(
            VaccineBatchGUI.this,
            "Please select the batch", null,
            WARNING_MESSAGE);
    }
});
vaccBatchInfoBtn_pnl.add(sel_btn);
}

/**
 * Registers this pcvs object to another pcvs attribute
 * in another JFrame.
 * This method will save the current pcvs object
 * to avoid create new data of pcvs.
 *
 * @param pcvsObj the current pcvs object to save.
 */
public void setPCVS(PCVS pcvsObj) {
    pcvs = pcvsObj;
}

/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
 */
public void setFile(File directory) {
    file = directory;
}

/**
 * Registers this index admin to another
 * index admin attribute in another JFrame
 * to avoid lost value.
 *
 * @param idx the value to save.
 */
public void setIndexAdmin(int idx) {
    iAdmin = idx;
}
}

```

VaccinationListGUI

```
package frame;

import pcvs.PCVS;
import table.VaccinationListTableModel;
11
import javax.swing.border.EmptyBorder;
import java.awt.*;
import javax.swing.*;
import java.awt.event.WindowEvent;
import java.awt.event.WindowAdapter;
import java.util.Objects;
import java.io.File;
import java.util.stream.IntStream;

import static java.awt.Color.PINK;
import static java.awt.Color.WHITE;
9import static java.awt.FlowLayout.*;
import static java.awt.Font.PLAIN;
import static javax.swing.JOptionPane.WARNING_MESSAGE;
import static javax.swing.JOptionPane.showMessageDialog;
import static javax.swing.SwingConstants.CENTER;

/**
 * VaccinationListGUI class to allow pcvs application
 * to display list of vaccination by selected batch number.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
14/
public class VaccinationListGUI extends JFrame {

    private final VaccinationListTableModel vaccinationTableModel;
    private JTable vaccinationInfo_table;
27    private PCVS pcvs;
    private File file = null;
    private int iAdmin;
    private int batchNum;
    private String vaccintionID;

    /**
     * Constructor to create the frame.
     */
    public VaccinationListGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display the list of vaccination
             * by selected batch number.
             * @param e the event to process.
             */
            @Override
```

```

public void windowOpened(WindowEvent e) {
    // Traverse vaccine collection.
    pcvs.getPCSVSVotes().forEach(vaccine -> {
        int bound = vaccine.getBatches().size();
        IntStream.range(0, bound).filter(j ->
            vaccine.getBatches().get(j)
                // Equals batch number.
                .getBatchNo() == batchNum).mapToObj(j -> {
                    vaccine.getBatches().get(j)).forEach(batch -> {
                        int bound1 = batch
                            .getVaccinations().size();
                        // Add pending and confirmed vaccination
                        // to table
                        IntStream.range(0, bound1).filter(k ->
                            Objects.equals(batch.getVaccinations()
                                .get(k).getStatus(), "pending") ||
                            Objects.equals(batch.getVaccinations()
                                .get(k).getStatus(), "confirmed"))
                                .filter(k -> !vaccinationTableModel.contains(
                                    batch.getVaccinations().get(k)))
                                .forEach(k -> vaccinationTableModel.add(
                                    batch.getVaccinations().get(k)));
                    });
                });
            });
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setBounds(250, 150, 120, 600);
        JPanel content_pnl = new JPanel();
        content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(content_pnl);
        content_pnl.setLayout(null);

        pcvs = new PCVS();

        JPanel sideBarAdminMenu_pnl = new JPanel();
        sideBarAdminMenu_pnl.setLayout(null);
        sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
        sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
        content_pnl.add(sideBarAdminMenu_pnl);

        JPanel vacBatch_pnl = new JPanel();
        vacBatch_pnl.setLayout(null);
        vacBatch_pnl.setBackground(new Color(79, 189, 194));
        vacBatch_pnl.setBounds(0, 195, 300, 40);
        sideBarAdminMenu_pnl.add(vacBatch_pnl);

        JLabel vacBatch_lbl = new JLabel("View Vaccine Batch");
        vacBatch_lbl.setForeground(WHITE);
        vacBatch_lbl.setFont(new Font(".AppleSystemUIFont",
            PLAIN, 16));
        vacBatch_lbl.setBounds(36, 6, 258, 28);
        vacBatch_pnl.add(vacBatch_lbl);

        JPanel vacBatchSel_pnl = new JPanel();
        vacBatchSel_pnl.setBackground(new Color(253, 210, 155));
        vacBatchSel_pnl.setBounds(0, 0, 5, 40);

```

```

vacBatch_pnl.add(vacBatchSel_pnl);

JPanel vaccinationInfo_pnl = new JPanel();
vaccinationInfo_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(vaccinationInfo_pnl);
vaccinationInfo_pnl.setLayout(null);

JPanel vaccinationInfoTitle_pnl = new JPanel();
vaccinationInfoTitle_pnl.setLayout(null);
vaccinationInfoTitle_pnl.setBackground(PINK);
vaccinationInfoTitle_pnl.setBounds(0, 0, 700, 70);
vaccinationInfo_pnl.add(vaccinationInfoTitle_pnl);

JLabel vaccinationInfo_lbl =
    new JLabel("List Of Vaccination Information");
vaccinationInfo_lbl.setHorizontalTextPosition(CENTER);
vaccinationInfo_lbl.setForeground(WHITE);
vaccinationInfo_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro",
        PLAIN, 20));
vaccinationInfo_lbl.setBounds(6, 6, 688, 58);
vaccinationInfoTitle_pnl.add(vaccinationInfo_lbl);

JPanel vaccinationInfoList_pnl = new JPanel();
vaccinationInfoList_pnl.setLayout(null);
vaccinationInfoList_pnl.setBounds(0, 69, 700, 465);
vaccinationInfo_pnl.add(vaccinationInfoList_pnl);

JScrollPane vaccinationInfo_scrollPane =
    new JScrollPane(vaccinationInfo_table);
vaccinationInfo_scrollPane.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
vaccinationInfo_scrollPane.setBorder(null);
vaccinationInfo_scrollPane.setBounds(0, 0, 700, 465);
vaccinationInfoList_pnl.add(vaccinationInfo_scrollPane);

String[] colHeaderVaccination = {"Vaccination ID",
    "Appointment Date", "Status"};
vaccinationTableModel =
    new VaccinationListTableModel(colHeaderVaccination, 0);

vaccinationInfo_table = new JTable(vaccinationTableModel);
vaccinationInfo_table.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 12));
vaccinationInfo_table.setBorder(null);
vaccinationInfo_scrollPane.setViewportView(vaccinationInfo_table);

JPanel vaccinationInfoBtn_panel = new JPanel();
vaccinationInfoBtn_panel.setBackground(PINK);
vaccinationInfoBtn_panel.setBounds(0, 533, 700, 39);
vaccinationInfo_pnl.add(vaccinationInfoBtn_panel);
vaccinationInfoBtn_panel.setLayout(
    new FlowLayout(RIGHT));

JButton back_btn = new JButton("Back");
back_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
back_btn.addActionListener(e -> {

```

```

VaccineBatchGUI vaccineBatch = new VaccineBatchGUI();
vaccineBatch.pack();
vaccineBatch.setBounds(0, 0, 1000, 600);
vaccineBatch.setLocationRelativeTo(VaccinationListGUI.this);

vaccineBatch.setPCVS(pcv);
vaccineBatch.setFile(file);
vaccineBatch.setIndexAdmin(iAdmin);

vaccineBatch.setVisible(true);
setVisible(false);
});
vaccinationInfoBtn_panel.add(back_btn);

JButton sel_btn = new JButton("Select");
sel_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
sel_btn.addActionListener(e -> {
    int selRow = vaccinationInfo_table.getSelectedRow();
    if (selRow != -1) {
        Object vaccinationIDObj =
            vaccinationInfo_table.getValueAt(selRow, 0);
        vaccinationID = (String) vaccinationIDObj;

        AppointmentInformationGUI appointment =
            new AppointmentInformationGUI();
        appointment.pack();
        appointment.setBounds(0, 0, 1000, 600);
        appointment.setLocationRelativeTo(VaccinationListGUI.this);

        appointment.setPCVS(pcv);
        appointment.setFile(file);
        appointment.setVaccinationID(vaccinationID);
        appointment.setIndexAdmin(iAdmin);
        appointment.setBatchNumber(batchNum);

        appointment.setVisible(true);
        setVisible(false);
    } else {
        showMessageDialog(VaccinationListGUI.this,
            "Please select the vaccination",
            null, WARNING_MESSAGE);
    }
});
vaccinationInfoBtn_panel.add(sel_btn);
}

/**
 * Registers this pcv object to another pcv attribute
 * in another JFrame.
 * This method will save the current pcv object
 * to avoid create new data of pcv.
 *
 * @param pcvObj the current pcv object to save.
 */
public void setPCVS(PCVS pcvObj) {
    pcv = pcvObj;
}

```

```
/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
 */
public void setFile(File directory) {
    file = directory;
}

/**
 * Registers this batch number to display appointment
 * information in another JFrame.
 *
 * @param num batch number to save.
 */
public void setBatchNumber(int num) {
    batchNum = num;
}

/**
 * Registers this index admin to another
 * index admin attribute in another JFrame
 * to avoid lost value.
 *
 * @param idx the value to save.
 */
public void setIndexAdmin(int idx) {
    iAdmin = idx;
}

}
```

AppointmentInformationGUI

```
package frame;

import pcvs.*;

import javax.swing.border.EmptyBorder;
import java.awt.event.ActionEvent;
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.*;
import java.awt.event.WindowEvent;
import java.awt.event.WindowAdapter;
import java.util.stream.IntStream;
import java.io.File;

import static java.awt.Color.*;
import static java.awt.Font.*;
import static javax.swing.JOptionPane.*;
import static javax.swing.ListSelectionModel.*;
import static javax.swing.SwingConstants.*;

/**
 * AppointmentInformationGUI class allow pcvs GUI application to
 * display detail appointment information:vaccine, patient,
 * and vaccination in table format.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
17/
public class AppointmentInformationGUI extends JFrame
    implements ActionListener {

    private final DefaultListModel<String> appointmentListModel;
    private final JButton confirm_btn;
    private final JButton reject_btn;
    private final JButton record_btn;

    private PCVS pcvs;
    private String vaccinationID;
    private int batchNum;
    private int iAdmin;
    private File file = null;

    /**
     * Constructor to create the frame.
     */
    public AppointmentInformationGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display information about

```

```

    * vaccine, patient, and vaccination in table format.
    * @param e the event to process.
    */
@Override
public void windowOpened(WindowEvent e) {
    appointmentListModel.clear();
    pcvs.getPCVSVaccines()
        // Traverse Batch in Vaccine collection.
        .forEach(vaccine -> vaccine.getBatches()

            // Traverse Vaccination in Batch collection.
            .forEach(batch -> batch.getVaccinations().stream()
                .filter(vacc -> vacc.getVaccinationID()

                    // Same vaccination ID.
                    .equals(vaccinationID)).forEach(vacc -> {
                        int bound1 = pcvs.getPCVSUsers().size();
                        // Traverse Vaccination in Patient collection.
                        IntStream.range(0, bound1).forEach(x -> {
                            if (pcvs.getPCVSUsers().get(x) instanceof
                                Patient pt) {
                                int bound = ((Patient) pcvs.getPCVSUsers()
                                    .get(x)).getVaccinations().size();

                                // Add information to table.
                                IntStream.range(0, bound).filter(y -> vacc
                                    .equals(pt.getVaccinations().get(y)))
                                    .filter(y -> !appointmentListModel
                                        .contains(pt) &&
                                        !appointmentListModel.contains(batch) &&
                                        !appointmentListModel.contains(vaccine))
                                    .forEach(y -> {
                                        appointmentListModel
                                            .addElement("Full Name: " +
                                                pt.getFullName());
                                        appointmentListModel
                                            .addElement("IC or Passport: " +
                                                pt.getICPassport());
                                        appointmentListModel
                                            .addElement("Batch Number: " +
                                                + batch.getBatchNo());
                                        appointmentListModel
                                            .addElement("Expiry Date: " +
                                                batch.getExpiryDate());
                                        appointmentListModel.addElement("Vaccine: " +
                                            + vaccine);
                                    });
                            }
                        });
                    }));
}
};

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(250, 150, 1000, 600);
JPanel content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);

```

```

content_pnl.setLayout(null);

pcvs = new PCVS();
appointmentListModel = new DefaultListModel<>();

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setLayout(null);
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(sideBarAdminMenu_pnl);

JPanel vacBatch_pnl = new JPanel();
vacBatch_pnl.setLayout(null);
vacBatch_pnl.setBackground(new Color(79, 189, 194));
vacBatch_pnl.setBounds(0, 195, 300, 40);
sideBarAdminMenu_pnl.add(vacBatch_pnl);

JLabel vacBatch_lbl = new JLabel("View Vaccine Batch");
vacBatch_lbl.setForeground(WHITE);
vacBatch_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
vacBatch_lbl.setBounds(36, 6, 258, 28);
vacBatch_pnl.add(vacBatch_lbl);

JPanel vacBatchSel_pnl = new JPanel();
vacBatchSel_pnl.setBackground(new Color(253, 210, 155));
vacBatchSel_pnl.setBounds(0, 0, 5, 40);
vacBatch_pnl.add(vacBatchSel_pnl);

JPanel appointment_pnl = new JPanel();
appointment_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(appointment_pnl);
appointment_pnl.setLayout(null);

JPanel appointmentTitle_pnl = new JPanel();
appointmentTitle_pnl.setLayout(null);
appointmentTitle_pnl.setBackground(PINK);
appointmentTitle_pnl.setBounds(0, 0, 700, 70);
appointment_pnl.add(appointmentTitle_pnl);

JLabel appointment_lbl =
    new JLabel("Vaccination Appointment Information");
appointment_lbl.setHorizontalAlignment(CENTER);
appointment_lbl.setForeground(WHITE);
appointment_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro",
        PLAIN, 20));
appointment_lbl.setBounds(6, 6, 688, 58);
appointmentTitle_pnl.add(appointment_lbl);

JPanel appointmentBtn_pnl = new JPanel();
appointmentBtn_pnl.setBackground(PINK);
appointmentBtn_pnl.setBounds(0, 533, 700, 39);
appointment_pnl.add(appointmentBtn_pnl);
appointmentBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

 JButton back_btn = new JButton("Back");

```

```

back_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
back_btn.addActionListener(e -> {
    VaccinationListGUI vaccList = new VaccinationListGUI();
    vaccList.pack();
    vaccList.setBounds(0, 0, 1000, 600);
    vaccList
        .setLocationRelativeTo(AppointmentInformationGUI.this);
    vaccList.setPCVS(pcvs);
    vaccList.setFile(file);
    vaccList.setBatchNumber(batchNum);
    vaccList.setIndexAdmin(iAdmin);
    vaccList.setBatchNumber(batchNum);
    vaccList.setVisible(true);
    setVisible(false);
});

JButton logOut_btn = new JButton("Log Out");
logOut_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
logOut_btn.addActionListener(e -> {
    PCVSGUI pcvs_gui = new PCVSGUI();
    pcvs_gui.pack();
    pcvs_gui.setBounds(0, 0, 1000, 600);
    pcvs_gui.setLocationRelativeTo(AppointmentInformationGUI.this);
    pcvs_gui.setPCVS(pcvs);
    pcvs_gui.setFileDir(file);
    pcvs_gui.setVisible(true);
    setVisible(false);
});
appointmentBtn_pnl.add(logOut_btn);
appointmentBtn_pnl.add(back_btn);

reject_btn = new JButton("Reject");
reject_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
appointmentBtn_pnl.add(reject_btn);

confirm_btn = new JButton("Confirm");
confirm_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
appointmentBtn_pnl.add(confirm_btn);

record_btn = new JButton("Record");
record_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
appointmentBtn_pnl.add(record_btn);

reject_btn.addActionListener(this);
confirm_btn.addActionListener(this);
record_btn.addActionListener(this);

JPanel appointmentList_pnl = new JPanel();
appointmentList_pnl.setLayout(null);
appointmentList_pnl.setBounds(0, 70, 700, 463);
appointment_pnl.add(appointmentList_pnl);

```

```

        JList<String> appointment_lst =
            new JList<>(appointmentListModel);
        appointment_lst.setSelectionMode(
            SINGLE_SELECTION);
        appointment_lst.setBounds(0, 0, 700, 463);
        appointmentList_pnl.add(appointment_lst);
    }

    /**
     * Registers this pcvs object to another pcvs attribute
     * in another JFrame.
     * This method will save the current pcvs object
     * to avoid create new data of pcvs.
     *
     * @param pcvsObj the current pcvs object to save.
     */
    public void setPCVS(PCVS pcvsObj) {
        pcvs = pcvsObj;
    }

    /**
     * Registers this file directory to another
     * file attribute in another JFrame.
     * This method will save the current file directory
     * to avoid current file lost.
     *
     * @param directory the directory file to save.
     */
    public void setFile(File directory) {
        file = directory;
    }

    /**
     * Registers this vaccination id to display
     * appointment information in this JFrame.
     *
     * @param id the value to record.
     */
    public void setVaccinationID(String id) {
        vaccintionID = id;
    }

    /**
     * Registers this batch number to display appointment
     * information in this JFrame.
     *
     * @param num batch number to record.
     */
    public void setBatchNumber(int num) {
        batchNum = num;
    }

    /**
     * Registers this index admin to another
     * index admin attribute in another JFrame
     * to avoid lost value.
     */

```

```

        * @param idx the value to save.
        */
    public void setIndexAdmin(int idx) {
        iAdmin = idx;
    }

    /**
     * Assign specific actions allow administrator to
     * confirm, reject, or record the vaccination apointment.
     *
     * @param e determine the object where the event occurs.
     */
    @Override
    public void actionPerformed(ActionEvent e) {
        IntStream.range(0, pcvs.getPCVSUsers().size())
            .filter(x -> pcvs.getPCVSUsers().get(x) instanceof Patient)
            .mapToObj(x -> (Patient) pcvs.getPCVSUsers().get(x))
            .forEach(patient -> {
                int bound = patient.getVaccinations().size();
                IntStream.range(0, bound).forEach(y -> {
                    Vaccination vaccination = patient
                        .getVaccinations().get(y);

                    // same vaccination id
                    if (patient.getVaccinations().get(y).getVaccinationID()
                        .equals(vaccinationID)) {
                        if (e.getSource() == confirm_btn) {

                            // confirm the appointment
                            patient.getVaccinations().get(y)
                                .setStatus("confirmed");

                            showMessageDialog(
                                AppointmentInformationGUI.this,
                                "Appointment \"confirmed\"\n" +
                                    "Sent email notification to patient",
                                null, PLAIN_MESSAGE);
                        } else if (e.getSource() == reject_btn) {
                            // reject the appointment
                            patient.getVaccinations().get(y)
                                .setStatus("rejected");
                            String remarks =
                                showInputDialog("Enter remarks: ");
                            vaccination.setRemarks(remarks);
                            showMessageDialog(
                                AppointmentInformationGUI.this,
                                "Appointment \"rejected\"\n" +
                                    "Sent email notification to patient",
                                null, PLAIN_MESSAGE);
                        }
                    }
                });
            });

        if (e.getSource() == record_btn) {
            int adCount = 0;
            for (int x = 0; x < pcvs.getPCVSUsers().size(); x++) {

```

```

// Downcast User collection to Patient collection
if (pcvs.getPCVSUsers().get(x) instanceof Patient pt) {

    // Traverse Vaccination collection
    // in Patient collection
    for (int y = 0;
        y < pt.getVaccinations().size(); y++) {
        Vaccination vaccination =
            pt.getVaccinations().get(y);
        // Same Vaccination ID
        if (pt.getVaccinations().get(y)
            .getVaccinationID()
            .equals(vaccinationID)) {
            pt.getVaccinations().get(y)
                .setStatus("administered");
            String remarks =
                showInputDialog("Enter remarks: ");

            adCount++;
            pcvs.setQuantityAdministered(adCount);
            vaccination.setRemarks(remarks);

            showMessageDialog(
                AppointmentInformationGUI.this,
                "The vaccination status " +
                "is set to \"administered\" and " +
                "an email sent to the patient",
                null, PLAIN_MESSAGE);
        }
    }
}
AdminMenuGUI adminMenu = new AdminMenuGUI();
adminMenu.pack();
adminMenu.setBounds(0, 0, 1000, 600);
adminMenu.setLocationRelativeTo(AppointmentInformationGUI.this);
adminMenu.setPCVS(pcvs);
adminMenu.setFile(file);
adminMenu.setIndexAdmin(iAdmin);

adminMenu.setVisible(true);
setVisible(false);
}
}

```

AdministratorInformationGUI

```
package frame;
1 import pcvs.Administrator;
import pcvs.PCVS;
import pcvs.User;
import table.AdministratorTableModel;
11 import javax.swing.*;
import java.awt.*;
import javax.swing.border.EmptyBorder;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
import java.io.File;
38 import static javax.swing.JOptionPane.showConfirmDialog;
import static javax.swing.SwingConstants.*;

/**
 * AdministratorInformationGUI class to pcvs GUI application
 * to display detail Administrator information.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA
14/
public class AdministratorInformationGUI extends JFrame {

    private final AdministratorTableModel administratorTableModel;
    private JTable admin_table;
    private PCVS pcvs;
    private File file = null;

    /**
     * Create the AdminMenuFrame.
     */
    public AdministratorInformationGUI() {
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowOpened(WindowEvent e) {
                administratorTableModel.clear();
                IntStream.range(0, pcvs.getPCVSUsers().size())

                    // get Administrator object in User collection
                    .filter(i -> pcvs.getPCVSUsers().get(i)
                        instanceof Administrator)
                    .mapToObj(i -> (Administrator)
                        pcvs.getPCVSUsers().get(i))
            }
        });
    }
}
```

```

        // avoid duplicate
        .filter(administrator -> !administratorTableModel
            .contains(administrator))
        .forEach(administratorTableModel::add);
    }
}

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(250, 150, 1200, 600);
JPanel content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setLayout(null);
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(sideBarAdminMenu_pnl);

JPanel adminInfo_pnl = new JPanel();
adminInfo_pnl.setLayout(null);
adminInfo_pnl.setBackground(new Color(79, 189, 194));
adminInfo_pnl.setBounds(0, 235, 300, 40);
sideBarAdminMenu_pnl.add(adminInfo_pnl);

JLabel adminInfo_lbl = new JLabel("Administrator Information");
adminInfo_lbl.setForeground(Color.WHITE);
adminInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 16));
adminInfo_lbl.setBounds(36, 6, 258, 28);
adminInfo_pnl.add(adminInfo_lbl);

JPanel adminInfoSel_pnl = new JPanel();
adminInfoSel_pnl.setBackground(new Color(253, 210, 155));
adminInfoSel_pnl.setBounds(0, 0, 5, 40);
adminInfo_pnl.add(adminInfoSel_pnl);

JPanel adminTable_pnl = new JPanel();
adminTable_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(adminTable_pnl);
adminTable_pnl.setLayout(null);

JScrollPane admin_scrollPane = new JScrollPane(admin_table);
admin_scrollPane.setBorder(null);
admin_scrollPane.setBounds(0, 70, 700, 466);
adminTable_pnl.add(admin_scrollPane);

String[] adminColHeader = {"Username", "Password", "Email",
    "Full Name", "Staff ID"};
administratorTableModel =
    new AdministratorTableModel(adminColHeader, 0);

admin_table = new JTable(administratorTableModel);
admin_table.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 12));

```

```

admin_table.setBorder(null);
admin_scrollPane.setViewportView(admin_table);

JPanel adminInfoTitle_pnl = new JPanel();
adminInfoTitle_pnl.setLayout(null);
adminInfoTitle_pnl.setBackground(Color.PINK);
adminInfoTitle_pnl.setBounds(0, 0, 700, 70);
adminTable_pnl.add(adminInfoTitle_pnl);

JLabel adminInfoTitle_lbl =
    new JLabel("Administrator Information");
adminInfoTitle_lbl.setHorizontalAlignment(CENTER);
adminInfoTitle_lbl.setForeground(Color.WHITE);
adminInfoTitle_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", Font.PLAIN, 20));
adminInfoTitle_lbl.setBounds(6, 6, 688, 58);
adminInfoTitle_pnl.add(adminInfoTitle_lbl);

JPanel adminBtn_pnl = new JPanel();
adminBtn_pnl.setBackground(Color.PINK);
adminBtn_pnl.setBounds(0, 533, 700, 39);
adminTable_pnl.add(adminBtn_pnl);
adminBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

JButton originalOrder_btn = new JButton("Original Order");
originalOrder_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 13));
originalOrder_btn.addActionListener(e -> {
    administratorTableModel.clear();
    IntStream.range(0, pcvs.getPCVSUsers().size())

        // get Administrator object in User collection
        .filter(i -> pcvs.getPCVSUsers().get(i)
            instanceof Administrator)
        .mapToObj(i -> (Administrator)
            pcvs.getPCVSUsers().get(i))

        // avoid duplicate
        .filter(administrator -> !administratorTableModel
            .contains(administrator))
        .forEach(administratorTableModel::add);
});
adminBtn_pnl.add(originalOrder_btn);

JButton sorted_btn = new JButton("Sorted by Full Name");
sorted_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 13));
sorted_btn.addActionListener(e -> {
    administratorTableModel.clear();
    ArrayList<Administrator> administrators =
        IntStream.range(0, pcvs.getPCVSUsers().size())
            .filter(i -> pcvs.getPCVSUsers().get(i)
                instanceof Administrator)
            .mapToObj(i -> (Administrator)
                pcvs.getPCVSUsers().get(i))

        // compare by full name

```

```

        .sorted(Comparator.comparing(User::getFullName))
        .collect(Collectors.toCollection(ArrayList::new));
    administrators.forEach(administratorTableModel::add);
});
adminBtn_pnl.add(sorted_btn);

JButton delete_btn = new JButton("Delete Selection");
delete_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 13));
delete_btn.addActionListener(e -> {
    int selRow = admin_table.getSelectedRow();
    if (selRow != -1) {
        Administrator a =
            administratorTableModel.getAdministrator(selRow);
        int n = showConfirmDialog(null,
            "Are you sure you want to delete " +
            a.getUsername() + " with staff ID " +
            a.getStaffID());
        if (n == JOptionPane.YES_OPTION) {
            administratorTableModel.remove(selRow);
            if (pcvs.getUser(a) != null) {
                pcvs.getUser(a).remove(pcv);
                int iAdmin = pcvs
                    .getHealthcareCentre(a.getStaffID());
                pcvs.getPCVSHealthcareCentres().get(iAdmin)
                    .getAdministrators()
                    .remove(pcv);
            }
        }
    } else
        JOptionPane.showMessageDialog(null,
            "Please select a row!", null,
            JOptionPane.WARNING_MESSAGE);
});
adminBtn_pnl.add(delete_btn);
19
JButton back_btn = new JButton("Back");
back_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 13));
back_btn.addActionListener(e -> {
    AdminMenuGUI adminMenu = new AdminMenuGUI();
    adminMenu.pack();
    adminMenu.setBounds(0, 0, 1000, 600);
    adminMenu.setLocationRelativeTo(
        AdministratorInformationGUI.this);
    adminMenu.setPCVS(pcv);
    adminMenu.setFile(file);
    adminMenu.setVisible(true);
    setVisible(false);
});
adminBtn_pnl.add(back_btn);
}

/**
 * Registers this pcv object to another pcv attribute
 * in another JFrame.

```

```
* This method will save the current pcvs object
* to avoid create new data of pcvs.
*
* @param pcvsObj the current pcvs object to save.
*/
public void setPCVS(PCVS pcvsObj) {
    this.pcvs = pcvsObj;
}

/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
*/
public void setFile(File directory) {
    file = directory;
}
}
```

PatientInformationGUI

```
package frame;
1
import pcvs.PCVS;
import pcvs.Patient;
import pcvs.User;
import table.PatientTableModel;

4import javax.swing.border.EmptyBorder;
import java.awt.event.WindowAdapter;
import javax.swing.*;
import java.awt.event.WindowEvent;
import java.awt.*;
import java.io.File;
import java.util.Comparator;
import java.util.ArrayList;
import java.util.stream.IntStream;
import java.util.stream.Collectors;

import static java.awt.Color.PINK;
import static java.awt.Color.WHITE;
import static 6ava.awt.Font.PLAIN;
import static javax.swing.JOptionPane.*;
import static javax.swing.SwingConstants.CENTER;

/**
 * PatientInformationGUI class allow pcvs GUI application to
 * display detail Patient information.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
2*/
public class PatientInformationGUI extends JFrame {

    private final JPanel content_pnl;
    private final PatientTableModel patientTableModel;
    private JTable patientInfo_table;
    private PCVS pcvs;
    private File file = null;

    public PatientInformationGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display detail Patient information
             * in table model.
             * @param e the event to process.
             */
            @Override
            public void windowOpened(WindowEvent e) {
                patientTableModel.clear();
            }
        });
    }
}
```

```

        IntStream.range(0, pcvs.getPCVSUsers().size())

            // get Patient object in User collection
            .filter(i -> pcvs.getPCVSUsers().get(i)
                instanceof Patient)

            // avoid duplicate
            .filter(i -> !patientTableModel
                .contains((Patient) pcvs.getPCVSUsers().get(i)))
            .forEach(i -> patientTableModel
                .add((Patient) pcvs.getPCVSUsers().get(i)));
    }

});  

    setDefaultCloseOperation(EXIT_ON_CLOSE);
setBounds(250, 150, 1000, 600);
content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setLayout(null);
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(sideBarAdminMenu_pnl);

JPanel patientInfo_pnl = new JPanel();
patientInfo_pnl.setLayout(null);
patientInfo_pnl.setBackground(new Color(79, 189, 194));
patientInfo_pnl.setBounds(0, 275, 300, 40);
sideBarAdminMenu_pnl.add(patientInfo_pnl);

JLabel patientInfo_lbl = new JLabel("Patient Information");
patientInfo_lbl.setForeground(WHITE);
patientInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
patientInfo_lbl.setBounds(36, 6, 258, 28);
patientInfo_pnl.add(patientInfo_lbl);

JPanel patientInfoSel_pnl = new JPanel();
patientInfoSel_pnl.setBackground(new Color(253, 210, 155));
patientInfoSel_pnl.setBounds(0, 0, 5, 40);
patientInfo_pnl.add(patientInfoSel_pnl);

JPanel patientInfoTable_pnl = new JPanel();
patientInfoTable_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(patientInfoTable_pnl);
patientInfoTable_pnl.setLayout(null);

JScrollPane patientInfo_scrollPane =
    new JScrollPane(patientInfo_table);
patientInfo_scrollPane.setBorder(null);
patientInfo_scrollPane.setBounds(0, 70, 700, 464);
patientInfoTable_pnl.add(patientInfo_scrollPane);

```

```

String[] colHeader = {"Username", "Password", "Email",
                     "Full Name", "IC or Passport"};
patientTableModel = new PatientTableModel(colHeader, 0);

patientInfo_table = new JTable(patientTableModel);
patientInfo_table.setFont(new Font(".AppleSystemUIFont",
                                    PLAIN, 12));
patientInfo_table.setBorder(null);
patientInfo_scrollPane.setViewportView(patientInfo_table);

JPanel patientInfoTitle_pnl = new JPanel();
patientInfoTitle_pnl.setLayout(null);
patientInfoTitle_pnl.setBackground(PINK);
patientInfoTitle_pnl.setBounds(0, 0, 700, 70);
patientInfoTable_pnl.add(patientInfoTitle_pnl);

JLabel patientInfoTitle_lbl = new JLabel("Patient Information");
patientInfoTitle_lbl.setHorizontalAlignment(CENTER);
patientInfoTitle_lbl.setForeground(WHITE);
patientInfoTitle_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 20));
patientInfoTitle_lbl.setBounds(6, 6, 688, 58);
patientInfoTitle_pnl.add(patientInfoTitle_lbl);

JPanel patientBtn_pnl = new JPanel();
patientBtn_pnl.setBackground(PINK);
patientBtn_pnl.setBounds(0, 533, 700, 39);
patientInfoTable_pnl.add(patientBtn_pnl);
patientBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

 JButton originalOrder_btn = new JButton("Original Order");
originalOrder_btn.setFont(new Font(".AppleSystemUIFont",
                                    PLAIN, 12));
originalOrder_btn.addActionListener(e -> {
    patientTableModel.clear();
    IntStream.range(0, pcvs.getPCVSUsers().size())
        .filter(i -> pcvs.getPCVSUsers().get(i)
            instanceof Patient)
        .filter(i -> !patientTableModel.contains((Patient)
            pcvs.getPCVSUsers().get(i)))
        .forEach(i -> patientTableModel
            .add((Patient) pcvs.getPCVSUsers().get(i)));
});
patientBtn_pnl.add(originalOrder_btn);

 JButton delete_btn = new JButton("Delete Selection");
delete_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 12));
delete_btn.addActionListener(e -> {
    int selRow = patientInfo_table.getSelectedRow();
    if (selRow != -1) {
        Patient p = patientTableModel.getPatient(selRow);
        int choice = showConfirmDialog(null,
            "Are you sure you want to remove " +
            p.getUsername(), "Warning",
            OK_CANCEL_OPTION,
            WARNING_MESSAGE);
        if (choice == YES_OPTION) {

```

```

        patientTableModel.remove(selRow);
        if (pcvs.getUser(p) != null) {
            pcvs.getPCVSUsers().remove(pcvsc.getUser(p));
        }
    }
} else
    showMessageDialog(null,
        "Please select a row!", null,
        WARNING_MESSAGE);
});

JButton sorted_btn = new JButton("Sorted by Full Name");
sorted_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 12));
sorted_btn.addActionListener(e -> {
    patientTableModel.clear();
    ArrayList<Patient> patients =
        IntStream.range(0, pcvs.getPCVSUsers().size())
            .filter(i -> pcvs.getPCVSUsers().get(i)
                instanceof Patient)
            .mapToObj(i -> (Patient) pcvs.getPCVSUsers()
                .get(i))

        // compare by full name
        .sorted(Comparator.comparing(User::getFullName))
        .collect(Collectors.toCollection(ArrayList::new));
    patients.forEach(patientTableModel::add);
});
patientBtn_pnl.add(sorted_btn);
delete_btn.setActionCommand("Cancel");
patientBtn_pnl.add(delete_btn);
19
 JButton back_btn = new JButton("Back");
back_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 12));
back_btn.addActionListener(e -> {
    AdminMenuGUI adminMenu = new AdminMenuGUI();
    adminMenu.pack();
    adminMenu.setBounds(0, 0, 1000, 600);
    adminMenu
        .setLocationRelativeTo(PatientInformationGUI.this);
    adminMenu.setPCVS(pcvsc);
    adminMenu.setFile(file);
    adminMenu.setVisible(true);
    setVisible(false);
});
patientBtn_pnl.add(back_btn);
}

/**
 * Registers this pcvsc object to another pcvsc attribute
 * in another JFrame.
 * This method will save the current pcvsc object
 * to avoid create new data of pcvsc.
 *
 * @param pcvscObj the current pcvsc object to save.
 */
public void setPCVS(PCVS pcvscObj) {
    pcvsc = pcvscObj;
}

```

```
}

/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
 */
public void setFile(File directory) {
    file = directory;
}

}
```

VaccinationInformationGUI

```
package frame;
1
import pcvs.PCVS;
import table.VaccinationTableModel;

import javax.swing.*;
12 import java.awt.*;
import javax.swing.border.EmptyBorder;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;

import static java.awt.Color.*;
import static java.awt.Font.*;
import static javax.swing.JFrame.*;
import static javax.swing.SwingConstants.*;

/**
 * VaccinationInformationGUI class allow pcvs GUI application to
 * display detail vaccination information in table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
2 */
public class VaccinationInformationGUI extends JFrame {

    private final JPanel content_pnl;
    private final VaccinationTableModel vaccinationTableModel;
    private JTable vaccInfo_table;
    private PCVS pcvs;
    private File file = null;

    /**
     * Create the vaccination information frame.
     */
    public VaccinationInformationGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display detail Vaccination information.
             * @param e the event to process.
             */
            @Override
            public void windowOpened(WindowEvent e) {
                vaccinationTableModel.clear();
                pcvs.getPCVSVaccines()
                    // traverse vaccination obj in batch collection
                    .forEach(vaccine -> vaccine.getBatches().stream()
                        .flatMap(batch -> batch.getVaccinations().stream())
                        // avoid duplicate
                        .filter(vaccination -> !vaccinationTableModel
```

```

        .contains(vaccination))
        .forEach(vaccinationTableModel::add));
    });
setDefaultCloseOperation(EXIT_ON_CLOSE);
setBounds(250, 250, 1000, 600);
content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel sideBarAdminMenu_pnl = new JPanel();
sideBarAdminMenu_pnl.setLayout(null);
sideBarAdminMenu_pnl.setBackground(new Color(40, 143, 148));
sideBarAdminMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(sideBarAdminMenu_pnl);

JPanel vaccInfoMenu_pnl = new JPanel();
vaccInfoMenu_pnl.setLayout(null);
vaccInfoMenu_pnl.setBackground(new Color(79, 189, 194));
vaccInfoMenu_pnl.setBounds(0, 315, 300, 40);
sideBarAdminMenu_pnl.add(vaccInfoMenu_pnl);

JLabel vaccInfo_lbl = new JLabel("Vaccination Information");
vaccInfo_lbl.setForeground(WHITE);
vaccInfo_lbl.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 16));
vaccInfo_lbl.setBounds(36, 6, 258, 28);
vaccInfoMenu_pnl.add(vaccInfo_lbl);

JPanel vaccInfoSel_pnl = new JPanel();
vaccInfoSel_pnl.setBackground(new Color(253, 210, 155));
vaccInfoSel_pnl.setBounds(0, 0, 5, 40);
vaccInfoMenu_pnl.add(vaccInfoSel_pnl);

JPanel vaccInfo_pnl = new JPanel();
vaccInfo_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(vaccInfo_pnl);
vaccInfo_pnl.setLayout(null);

JPanel vaccInfoTitle_pnl = new JPanel();
vaccInfoTitle_pnl.setLayout(null);
vaccInfoTitle_pnl.setBackground(PINK);
vaccInfoTitle_pnl.setBounds(0, 0, 700, 70);
vaccInfo_pnl.add(vaccInfoTitle_pnl);

JLabel vaccInfoTitle_lbl = new JLabel("Vaccination Information");
vaccInfoTitle_lbl.setHorizontalTextPosition(
    CENTER);
vaccInfoTitle_lbl.setForeground(WHITE);
vaccInfoTitle_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 20));
vaccInfoTitle_lbl.setBounds(6, 6, 688, 58);
vaccInfoTitle_pnl.add(vaccInfoTitle_lbl);

```

```

JPanel vaccInfoTable_pnl = new JPanel();
vaccInfoTable_pnl.setBounds(0, 70, 700, 464);
vaccInfo_pnl.add(vaccInfoTable_pnl);
vaccInfoTable_pnl.setLayout(null);

JScrollPane vaccInfo_scrollPane = new JScrollPane(vaccInfo_table);
vaccInfo_scrollPane.setBorder(null);
vaccInfo_scrollPane.setBounds(0, 0, 700, 464);
vaccInfoTable_pnl.add(vaccInfo_scrollPane);

String[] colHeaderVaccination = {"Vaccination ID",
    "Appointment Date", "Status", "Remarks"};
vaccinationTableModel =
    new VaccinationTableModel(colHeaderVaccination, 0);

vaccInfo_table = new JTable(vaccinationTableModel);
vaccInfo_table.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 12));
vaccInfo_table.setBorder(null);
vaccInfo_scrollPane.setViewportView(vaccInfo_table);

JPanel vaccInfoBtn_pnl = new JPanel();
vaccInfoBtn_pnl.setBackground(PINK);
vaccInfoBtn_pnl.setBounds(0, 533, 700, 39);
vaccInfo_pnl.add(vaccInfoBtn_pnl);
vaccInfoBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

JButton back_btn = new JButton("Back");
back_btn.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 13));
back_btn.addActionListener(e -> {
    AdminMenuGUI adminMenu = new AdminMenuGUI();
    adminMenu.pack();
    adminMenu.setBounds(0, 0, 1000, 600);
    adminMenu
        .setLocationRelativeTo(VaccinationInformationGUI.this);
    adminMenu.setPCVS(pcvS);
    adminMenu.setFile(file);
    adminMenu.setVisible(true);
    setVisible(false);
});
vaccInfoBtn_pnl.add(back_btn);
}

/**
 * Registers this pcvS object to another pcvS attribute
 * in another JFrame.
 * This method will save the current pcvS object
 * to avoid create new data of pcvS.
 *
 * @param pcvSObj the current pcvS object to save.
 */
public void setPCVS(PCVS pcvSObj) {
    pcvS = pcvSObj;
}

/**

```

```
* Registers this file directory to another
* file attribute in another JFrame.
* This method will save the current file directory
* to avoid current file lost.
*
* @param directory the directory file to save.
*/
public void setFile(File directory) {
    file = directory;
}

}
```

PatientMenuGUI

```
package frame;

import dialog.AppointmentDialog;
import pcvs.*;
import table.BatchTableModel;
35
import javax.swing.*;
import java.awt.*;
4import javax.swing.border.EmptyBorder;
import java.awt.event.MouseEvent;
import java.awt.event.WindowEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.WindowAdapter;
import java.util.ArrayList;
import java.util.Calendar;
import java.io.File;
import java.util.stream.IntStream;

import static javax.swing.JOptionPane.*;
import static javax.swing.JOptionPane.showMessageDialog;
import static javax.swing.SwingConstants.*;

/**
 * PatientMenuGUI class allow patient to request an vaccination
 * appointment and view the appointment status.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
45/
public class PatientMenuGUI extends JFrame {

    private static Integer vID = 0;
    private JTable batch_table;
    private PCVS pcvs;
    private File file = null;
    private HealthcareCentre selHealthCentre;
    private Vaccine selectedVaccine;
    private BatchTableModel batchTableModel;
    private int iHealthCentre;
    private int iVaccine;
    private int iPatient;

    /**
     * Create the patient menu frame.
     */
    public PatientMenuGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.

```

```

* This method display the available vaccine batch
* after patient select the available vaccine.
* @param e the event to process.
34
@Override
public void windowOpened(WindowEvent e) {
    try {
        for (int i = 0; i < selHealthCentre
            .getBatches().size(); i++) {
            // Get the expiryDate attribute from Batch object
            // and split from String to array
            int[] splitExpiryDate = PCVS.splitToArray(
                selHealthCentre.getBatches().get(i)
                .getExpiryDate());
            // Real time
            Calendar today = Calendar.getInstance();
            Calendar expires = Calendar.getInstance();
            // Set expires by expiry date attribute
            expires.set(splitExpiryDate[2],
                splitExpiryDate[0], splitExpiryDate[1]);
            // Traverse Batch object in Vaccine collection
            for (int j = 0; j <
                selectedVaccine.getBatches().size(); j++) {

                // Compare Batch from HealthcareCentre
                // and Vaccine
                if (selHealthCentre.getBatches().get(i)
                    .equals(selectedVaccine.getBatches()
                        .get(j)) &&
                    selHealthCentre.getBatches().get(i)
                    .getQuantityAvailable() > 0 &&
                    !today.after(expires)) {
                    if (!batchTableModel
                        .contains(selectedVaccine.getBatches()
                            .get(j))) {
                        batchTableModel
                            .add(selectedVaccine.getBatches()
                                .get(j));
                    }
                }
            }
        }
    } catch (NullPointerException ignored) {
    }
}
7;
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(250, 150, 1000, 600);
JPanel conte2t_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvs = new PCVS();

JPanel patientMenu_pnl = new JPanel();
patientMenu_pnl.setLayout(null);

```

```

patientMenu_pnl.setBackground(new Color(40, 143, 148));
patientMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(patientMenu_pnl);

JPanel appointment_pnl = new JPanel();
appointment_pnl.setLayout(null);
appointment_pnl.setBackground(new Color(79, 189, 194));
appointment_pnl.setBounds(0, 225, 300, 40);
patientMenu_pnl.add(appointment_pnl);

16
JLabel appointmentme21_lbl = new JLabel("Vaccination Appointment");
appointment_lbl.setForeground(Color.WHITE);
appointment_lbl.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 16));
appointment_lbl.setBounds(36, 6, 258, 28);
appointment_pnl.add(appointment_lbl);

JPanel appointmentSel_pnl = new JPanel();
appointmentSel_pnl.setBackground(new Color(253, 210, 155));
appointmentSel_pnl.setBounds(0, 0, 5, 40);
appointment_pnl.add(appointmentSel_pnl);

JPanel appointmentStatus_pnl = new JPanel();
appointmentStatus_pnl.addMouseListener(
    new PanelMenuButtonMouseAdapter(appointmentStatus_pnl) {
        @Override
        public void mouseClicked(MouseEvent e) {
            ArrayList<Patient> patients = new ArrayList<>();
            ArrayList<String> statusPatients = new ArrayList<>();
            IntStream.range(0, pcv.getPCVSUsers().size())
                .filter(x -> pcv.getPCVSUsers().get(x)
                    instanceof Patient)
                .mapToObj(x -> (Patient)
                    pcv.getPCVSUsers().get(iPatient))
                .forEach(pt -> {
                    pt.getVaccinations().stream()
                        .map(Vaccination::getStatus)
                        // avoid duplicate
                        .filter(status -> !statusPatients
                            .contains(status))
                        .forEach(status -> {
                            // Not in statusPatients collection
                            showMessageDialog(
                                PatientMenuGUI.this,
                                "Your appointment status: " +
                                status,
                                null, PLAIN_MESSAGE);

                            // Add to statusPatients collection
                            statusPatients.add(status);
                        });
                if (!patients.contains(pt)) {
                    // Patient not yet aggregation
                    // with Vaccination
                    if (pt.getVaccinations().size() == 0) {
                        showMessageDialog(
                            PatientMenuGUI.this,

```

```

        "Sorry, you haven't made an " +
        "vaccination appointment",
        null, WARNING_MESSAGE);

        // Add to patients collection
        patients.add(pt);
    }
}
});

appointmentStatus_pnl.setLayout(null);
appointmentStatus_pnl.setBackground(new Color(40, 143, 148));
appointmentStatus_pnl.setBounds(0, 265, 300, 40);
patientMenu_pnl.add(appointmentStatus_pnl);

JLabel appointmentStatus_lbl =
    new JLabel("Vaccination Appointment Status");
appointmentStatus_lbl.setForeground(Color.WHITE);
appointmentStatus_lbl.setFont(
    new Font(".AppleSystemUIFont", Font.PLAIN, 16));
appointmentStatus_lbl.setBounds(36, 6, 264, 28);
appointmentStatus_pnl.add(appointmentStatus_lbl);

JButton logOut_btn = new JButton("Log Out");
logOut_btn.addActionListener(e -> {

    int choice = showConfirmDialog(PatientMenuGUI.this,
        "Are you sure want to log out?",
        "Warning", YES_NO_CANCEL_OPTION,
        WARNING_MESSAGE);
    if (choice == YES_OPTION) {
        PCVSGUI pcvs_gui = new PCVSGUI();
        pcvs_gui.pack();
        pcvs_gui.setBounds(0, 0, 1000, 600);
        pcvs_gui.setLocationRelativeTo(PatientMenuGUI.this);
        pcvs_gui.setPCVS(pcv);
        pcvs_gui.setFileDir(file);
        pcvs_gui.setVisible(true);
        setVisible(false);
    }
});
logOut_btn.setForeground(Color.DARK_GRAY);
logOut_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 16));
logOut_btn.setBounds(6, 460, 288, 40);
patientMenu_pnl.add(logOut_btn);

JPanel vaccAppointment_pnl = new JPanel();
vaccAppointment_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(vaccAppointment_pnl);
vaccAppointment_pnl.setLayout(null);

JPanel vaccAppointmentTitle_pnl = new JPanel();
vaccAppointmentTitle_pnl.setLayout(null);
vaccAppointmentTitle_pnl.setBackground(Color.PINK);
vaccAppointmentTitle_pnl.setBounds(0, 0, 700, 70);

```

```

vaccAppointment_pnl.add(vaccAppointmentTitle_pnl);

JLabel vaccAppointment_lbl =
    new JLabel("Request Vaccination Appointment");
vaccAppointment_lbl.setHorizontalAlignment(CENTER);
vaccAppointment_lbl.setForeground(Color.WHITE);
vaccAppointment_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", Font.PLAIN, 20));
vaccAppointment_lbl.setBounds(6, 6, 688, 58);
vaccAppointmentTitle_pnl.add(vaccAppointment_lbl);

 JPanel vaccAppointmentBtn_pnl = new JPanel();
vaccAppointmentBtn_pnl.setBackground(Color.PINK);
vaccAppointmentBtn_pnl.setBounds(0, 533, 700, 39);
vaccAppointment_pnl.add(vaccAppointmentBtn_pnl);
vaccAppointmentBtn_pnl.setLayout
    (new FlowLayout(FlowLayout.RIGHT));

 JButton view_btn = new JButton("View Available Vaccine");
view_btn.setFont(new Font(".AppleSystemUIFont", Font.PLAIN, 13));
view_btn.addActionListener(e -> {

    ListAvailableVaccinesGUI availableVaccine =
        new ListAvailableVaccinesGUI();
    availableVaccine.pack();
    availableVaccine.setBounds(0, 0, 1000, 600);
    availableVaccine.setLocationRelativeTo(
        PatientMenuGUI.this);
    availableVaccine.setPCVS(pcvs);
    availableVaccine.setFile(file);
    availableVaccine.setIndexPatient(iPatient);
    availableVaccine.setVisible(true);
    setVisible(false);
    batchTableModel.clear();
});

 JButton request_btn = new JButton("Request Appointment");
request_btn.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 13));
request_btn.addActionListener(e -> {
    int selRow = batch_table.getSelectedRow();
    if (selRow != -1) {
        Object expiryDate = batchTableModel
            .getValueAt(selRow, 1);
        int qty = pcvs.getNumOfPendingByPatient() +
            pcvs.getQuantityAdministered();
        Object batchNumObject = batchTableModel
            .getValueAt(selRow, 0);
        int batchNum = (Integer) batchNumObject;
        int iBatch = -1;
        HealthcareCentre hc = pcvs.getPCVSHealthcareCentres()
            .get(iHealthCentre);
        Vaccine vc = pcvs.getPCCSVaccines().get(iVaccine);

    30 get selected batch number.
    for (int x = 0; x < hc.getBatches().size(); x++) {
        for (int y = 0; y < vc.getBatches().size(); y++) {
            if (batchNum == vc.getBatches().get(y)

```

```

        .getBatchNo() ) {
    iBatch = y;
}
}

showMessageDialog(PatientMenuGUI.this,
    "Expiry date: " + expiryDate +
    "\nQuantity available: " + qty,
    null, PLAIN_MESSAGE);
AppointmentDialog appointmentDialog =
    new AppointmentDialog(PatientMenuGUI.this);
appointmentDialog.pack();
appointmentDialog.setSize(700, 440);
appointmentDialog
    .setLocationRelativeTo(PatientMenuGUI.this);
appointmentDialog.setVisible(true);

String appointmentDate =
    appointmentDialog.getUpcomingDate();
try {
    boolean appointment =
        pcvs.appointmentDate(expiryDate, appointmentDate);
    while (!appointment) {
        showMessageDialog(PatientMenuGUI.this,
            "Appointment date after " +
            "the batch expiration date!",
            null, WARNING_MESSAGE);
        appointmentDialog.setVisible(true);
        appointmentDate =
            appointmentDialog.getUpcomingDate();
        appointment = pcvs
            .appointmentDate(expiryDate, appointmentDate);
    }
} catch (ArrayIndexOutOfBoundsException io) {
    showMessageDialog(PatientMenuGUI.this,
        "Invalid date!",
        null, WARNING_MESSAGE);
}
// create object vaccination for patient.
Vaccination vaccination = new Vaccination();
vaccination.setAppointmentDate(appointmentDate);
vaccination.setVaccinationID(generateVaccinationID());
vaccination.setStatus("pending");

// Registers vaccination.
Patient patient = (Patient) pcvs.getPCVSUsers()
    .get(iPatient);
patient.setVaccinations(vaccination);
selectedVaccine.getBatches().get(iBatch)
    .setVaccinations(vaccination);

showMessageDialog(PatientMenuGUI.this,
    "The vaccination is recorded for " +
    "the patient and the batch",
    null, PLAIN_MESSAGE);
batchTableModel.clear();

```

```

        } else {
            showMessageDialog(PatientMenuGUI.this,
                "Please select available vaccine " +
                "and then select batch!",
                null, WARNING_MESSAGE);
        }
    });
vaccAppointmentBtn_pnl.add(request_btn);
vaccAppointmentBtn_pnl.add(view_btn);

JPanel vaccAppointmentTable_pnl = new JPanel();
vaccAppointmentTable_pnl.setBounds(0, 70, 700, 464);
vaccAppointment_pnl.add(vaccAppointmentTable_pnl);
vaccAppointmentTable_pnl.setLayout(null);

JSscrollPane batch_scrollPane = new JSscrollPane(batch_table);
batch_scrollPane.setBorder(null);
batch_scrollPane.setBounds(0, 0, 700, 464);
vaccAppointmentTable_pnl.add(batch_scrollPane);

String[] batchColHeader = {"Batch Number", "Expiry Date",
    "Quantity Available"};
batchTableModel = new BatchTableModel(batchColHeader, 0);

batch_table = new JTable(batchTableModel);
batch_table.setFont(new Font(".AppleSystemUIFont",
    Font.PLAIN, 12));
batch_table.setBorder(null);
batch_scrollPane.setViewportView(batch_table);
}

/**
 * Calculates the length of the Vaccination ID.
 * Returns two positive integers + id, zero , or one integer + id.
 *
 * @return a String with the increment of id.
 */
public static String generateVaccinationID() {
    // if length id 1 digit add with 00 + id
    String id = (vID.toString().length() == 1) ? ("00" + vID)
        // if length id 2 digit add with 0 + id
        : ((vID.toString().length() == 2) ? ("0" + vID)
            : vID.toString());
    vID++; // increment
    return "V" + id;
}

/**
 * Registers this pcvs object to another pcvs attribute
 * in another JFrame.
 * This method will save the current pcvs object
 * to avoid create new data of pcvs.
 *
 * @param pcvsObj the current pcvs object to save.
 */
public void setPCVS(PCVS pcvsObj) {
    pcvs = pcvsObj;
}

```

```

}

/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
 */
public void setFile(File directory) {
    file = directory;
}

/**
 * Registers this healthcare centre attribute to
 * display batches of vaccine.
 *
 * @param healthcareCentre selected healthcare centre.
 */
public void setSelHealthCentre(HealthcareCentre healthcareCentre) {
    selHealthCentre = healthcareCentre;
}

/**
 * Registers this vaccine attribute to display
 * batches of vaccine.
 *
 * @param vaccine selected vaccine.
 */
public void setSelectedVaccine(Vaccine vaccine) {
    selectedVaccine = vaccine;
}

/**
 * Registers this index Healthcare Centre to
 * gets healthcare centre object by index.
 *
 * @param idx selected index healthcare centre.
 */
public void setIndexHealthcareCentre(int idx) {
    iHealthCentre = idx;
}

/**
 * Registers this index vaccine to gets
 * vaccine object by index.
 *
 * @param idx Vaccine object to search.
 */
public void setIndexVaccine(int idx) {
    iVaccine = idx;
}

/**
 * Registers this index patient to get
 * the Patient object by index.
 */

```

```

    * This method allow recording vaccination appointment
    * to Patient.
    *
    * @param idx Patient object to search.
    */
public void setIndexPatient(int idx) {
    iPatient = idx;
}

/**
 * PanelButtonMouseAdapter inner class to display a different color
 * when the mouse entered the panel.
 */
private static class PanelMenuButtonMouseListener
    extends MouseAdapter {
    JPanel panel;

    /**
     * PanelButtonMouseAdapter constructor for invocation in listeners.
     *
     * @param panel determine which panel receives the action.
     */
    public PanelMenuButtonMouseListener(JPanel panel) {
        this.panel = panel;
    }

    /**
     * Change this color when a mouse button
     * has been entered on a component. 5
     *
     * @param e the event to process.
6
@Override
public void mouseEntered(MouseEvent e) {
    panel.setBackground(new Color(79, 189, 194));
}

/**
 * Change this color when a mouse button
5
 * has been exited on a component.
*
* @param e the event to process.
6
@Override
public void mouseExited(MouseEvent e) {
    panel.setBackground(new Color(40, 143, 148));
}

/**
 * Change this color when a mouse button
5
 * has been pressed on a component.
*
* @param e the event to process.
14
@Override
public void mousePressed(MouseEvent e) {
    panel.setBackground(new Color(40, 143, 148));
}

```

```
}

/**
 * Change this color when mouse button
 * has been released.
 *
 * @param e the event to process.
6/
@Override
public void mouseReleased(MouseEvent e) {
    panel.setBackground(new Color(79, 189, 194));
}
}
```

ListAvailableVaccinesGUI

```
package frame;
1 import pcvs.HealthcareCentre;
import pcvs.PCVS;
import pcvs.Vaccine;
3 import java.awt.*;
import javax.swing.*;
import java.awt.event.ActionEvent;
import javax.swing.border.EmptyBorder;
import java.io.File;
import java.awt.event.ActionListener;
import java.util.stream.IntStream;

import static java.awt.Color.*;
import static 24va.awt.Font.*;
import static javax.swing.ListSelectionModel.*;
import static javax.swing.SwingConstants.*;

/**
 * ListAvailableVaccinesGUI class allow pcvs GUI application to
 * display list available vaccine and manufacturer.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
23/
public class ListAvailableVaccinesGUI extends JFrame {

    private final DefaultListModel<Vaccine> listModelAvailableVaccines;
    private final JList<Vaccine> availableVacc_lst;

    27ivate PCVS pcvS;
    private File file = null;
    private int iPatient;
    private int vaccineRecorded;
    private Vaccine selectedVaccine;

    /**
     * Create the list available vaccine frame.
     */
    public ListAvailableVaccinesGUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(250, 150, 1000, 600);
        JPanel content_pnl = new JPanel();
        content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(content_pnl);
        content_pnl.setLayout(null);

        pcvS = new PCVS();

        listModelAvailableVaccines = new DefaultListModel<>();
```

```

IntStream.range(0, pcvS.getPCVSVaccines().size())
    .forEach(i -> listModelAvailableVaccines
        .addElement(pcvS.getPCVSVaccines().get(i)));

JPanel patientMenu_pnl = new JPanel();
patientMenu_pnl.setLayout(null);
patientMenu_pnl.setBackground(new Color(40, 143, 148));
patientMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(patientMenu_pnl);

JPanel appointment_pnl = new JPanel();
appointment_pnl.setLayout(null);
appointment_pnl.setBackground(new Color(79, 189, 194));
appointment_pnl.setBounds(0, 225, 300, 40);
patientMenu_pnl.add(appointment_pnl);

JLabel appointment_lbl = new JLabel("Vaccination Appointment");
appointment_lbl.setForeground(WHITE);
appointment_lbl.setFont(
    new Font(".AppleSystemUIFont", PLAIN, 16));
appointment_lbl.setBounds(36, 6, 258, 28);
appointment_pnl.add(appointment_lbl);

JPanel appointmentSel_pnl = new JPanel();
appointmentSel_pnl.setBackground(new Color(253, 210, 155));
appointmentSel_pnl.setBounds(0, 0, 5, 40);
appointment_pnl.add(appointmentSel_pnl);

JPanel availableVacc_pnl = new JPanel();
availableVacc_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(availableVacc_pnl);
availableVacc_pnl.setLayout(null);

JPanel availableVaccTitle_pnl = new JPanel();
availableVaccTitle_pnl.setLayout(null);
availableVaccTitle_pnl.setBackground(PINK);
availableVaccTitle_pnl.setBounds(0, 0, 700, 70);
availableVacc_pnl.add(availableVaccTitle_pnl);

JLabel availableVacc_lbl = new JLabel("Available Vaccines");
availableVacc_lbl.setHorizontalTextPosition(CENTER);
availableVacc_lbl.setForeground(WHITE);
availableVacc_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 20));
availableVacc_lbl.setBounds(6, 6, 688, 58);
availableVaccTitle_pnl.add(availableVacc_lbl);

JPanel availableVaccBtn_pnl = new JPanel();
availableVaccBtn_pnl.setBackground(PINK);
availableVaccBtn_pnl.setBounds(0, 533, 700, 39);
availableVacc_pnl.add(availableVaccBtn_pnl);
availableVaccBtn_pnl.setLayout(new FlowLayout(FlowLayout.RIGHT));

 JButton record_btn = new JButton("Record"); 3
record_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
record_btn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

```

```

vaccineRecorded = availableVacc_lst.getSelectedIndex();
selectedVaccine = pcvs.getPCVSVaccines()
    .get(vaccineRecorded);

DefaultListModel<HealthcareCentre>
    healthcareCentresListModel;
healthcareCentresListModel = new DefaultListModel<>();

// traverse batch collection based on
// selected vaccine
selectedVaccine.getBatches().stream()
    .mapToInt(batch -> pcvs.equalsBatch(batch))
    .filter(iBatch -> !healthcareCentresListModel
        .contains(pcvs.getPCVSHCHealthcareCentres()
            .get(iBatch)))
    .forEach(iBatch -> healthcareCentresListModel
        .addElement(pcvs.getPCVSHCHealthcareCentres()
            .get(iBatch)));

ListHealthcareCentresGUI listHealthcareCentres =
    new ListHealthcareCentresGUI();
listHealthcareCentres.pack();
listHealthcareCentres.setBounds(0, 0, 1000, 600);
listHealthcareCentres
    .setLocationRelativeTo(ListAvailableVaccinesGUI.this);

// Pass value to ListHealthcareCentresGUI.
listHealthcareCentres.setPCVS(pcvs);
listHealthcareCentres.setFile(file);
listHealthcareCentres.setSelectedVaccine(selectedVaccine);
listHealthcareCentres.setIndexVaccine(vaccineRecorded);
listHealthcareCentres.setIndexPatient(iPatient);

listHealthcareCentres.setVisible(true);
setVisible(false);
}
});
availableVaccBtn_pnl.add(record_btn);

JPanel availableVaccList_pnl = new JPanel();
availableVaccList_pnl.setBounds(0, 70, 700, 464);
availableVacc_pnl.add(availableVaccList_pnl);
availableVaccList_pnl.setLayout(null);

availableVacc_lst = new JList<>(listModelAvailableVaccines);
availableVacc_lst.setFont(new Font(".AppleSystemUIFont",
    PLAIN, 12));
availableVacc_lst.setBounds(0, 0, 700, 464);
availableVaccList_pnl.add(availableVacc_lst);
availableVacc_lst.setSelectionMode(SINGLE_SELECTION);
}

/**
 * Registers this pcvs object to another pcvs attribute
 * in another JFrame.
 * This method will save the current pcvs object
 * to avoid create new data of pcvs.

```

```
* @param pcvsObj the current pcvs object to save.  
*/  
public void setPCVS(PCVS pcvsObj) {  
    this.pcvs = pcvsObj;  
}  
  
/**  
 * Registers this file directory to another  
 * file attribute in another JFrame.  
 * This method will save the current file directory  
 * to avoid current file lost.  
 *  
 * @param directory the directory file to save.  
 */  
public void setFile(File directory) {  
    file = directory;  
}  
  
/**  
 * Registers this index patient to get  
 * the Patient object by index.  
 * This method allow recording vaccination appointment  
 * to Patient.  
 *  
 * @param idx patient want to appointment.  
 */  
public void setIndexPatient(int idx) {  
    iPatient = idx;  
}  
}
```

ListHealthcareCentresGUI

```
package frame;
1
import pcvs.HealthcareCentre;
import pcvs.PCVS;
import pcvs.Vaccine;

import javax.swing.*;
import java.awt.*;
import javax.swing.border.EmptyBorder;
import java.awt.event.WindowEvent;
import java.io.File;
import java.awt.event.WindowAdapter;

import static java.awt.Color.*;
import static 24va.awt.Font.*;
import static javax.swing.ListSelectionModel.*;
import static javax.swing.SwingConstants.*;

/**
 * ListHealthcareCentresGUI class allow pcvs GUI application to
 * display healthcare centre name and addresses offering selected vaccine;
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class ListHealthcareCentresGUI extends JFrame {

    private final DefaultListModel<HealthcareCentre>
        healthcareCentresListModel;
    private JList<HealthcareCentre> healthcareCentres_lst;
    private PCVS pcvs;
    private File file = null;
    private int iPatient;
    private HealthcareCentre selectedHealthcareCentre;
    private int iHealthCentre;
    private Vaccine selectedVaccine;
    private int iVaccine;

    /**
     * Create the list healthcare centre frame.
     */
    public ListHealthcareCentresGUI() {
        addWindowListener(new WindowAdapter() {
            /**
             * Invoke when JFrame is opened.
             * This method display healthcare centre name
             * and addresses offering selected vaccine.
             * @param e the event to process.
             */
            @Override
```

```

public void windowOpened(WindowEvent e) {
    healthcareCentresListModel.clear();

    // Traverse batch collection in vaccine.
    selectedVaccine.getBatches().stream()

        // Equals batch at healthcare centre.
        .mapToInt(batch -> pcvS.equalsBatch(batch))

        // Avoid duplicate.
        .filter(iBatch -> !healthcareCentresListModel
            .contains(pcvS.getPCVSHCHealthcareCentres()
                .get(iBatch)))

        // Add to table.
        .forEach(iBatch -> healthcareCentresListModel
            .addElement(pcvS.getPCVSHCHealthcareCentres()
                .get(iBatch)));
}

7;
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(250, 150, 1000, 600);
JPanel content_pnl = new JPanel();
content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(content_pnl);
content_pnl.setLayout(null);

pcvS = new PCVS();
healthcareCentresListModel = new DefaultListModel<>();

JPanel patientMenu_pnl = new JPanel();
patientMenu_pnl.setLayout(null);
patientMenu_pnl.setBackground(new Color(40, 143, 148));
patientMenu_pnl.setBounds(0, 0, 300, 572);
content_pnl.add(patientMenu_pnl);

16
JPanel appointment_pnl = new JPanel();
appointment_pnl.setLayout(null);
appointment_pnl.setBackground(new Color(79, 189, 194));
appointment_pnl.setBounds(0, 225, 300, 40);
patientMenu_pnl.add(appointment_pnl);

JLabel appointment_lbl = new JLabel("Vaccination Appointment");
appointment_lbl.setForeground(WHITE);
appointment_lbl.setFont(new Font(".AppleSystemUIFont", PLAIN, 16));
appointment_lbl.setBounds(36, 6, 258, 28);
appointment_pnl.add(appointment_lbl);

JPanel appointmentSel_pnl = new JPanel();
appointmentSel_pnl.setBackground(new Color(253, 210, 155));
appointmentSel_pnl.setBounds(0, 0, 5, 40);
appointment_pnl.add(appointmentSel_pnl);

JPanel healthcareCentres_pnl = new JPanel();
healthcareCentres_pnl.setBounds(300, 0, 700, 572);
content_pnl.add(healthcareCentres_pnl);

```

```

healthcareCentres_pnl.setLayout(null);

JPanel healthcareCentresTitle_pnl = new JPanel();
healthcareCentresTitle_pnl.setLayout(null);
healthcareCentresTitle_pnl.setBackground(PINK);
healthcareCentresTitle_pnl.setBounds(0, 0, 700, 70);
healthcareCentres_pnl.add(healthcareCentresTitle_pnl);

JLabel healthcareCentresTitle_lbl =
    new JLabel("Healthcare Centres");
healthcareCentresTitle_lbl.setHorizontalAlignment(CENTER);
healthcareCentresTitle_lbl.setForeground(WHITE);
healthcareCentresTitle_lbl.setFont(
    new Font("Neue Haas Grotesk Text Pro", PLAIN, 20));
healthcareCentresTitle_lbl.setBounds(6, 6, 688, 58);
healthcareCentresTitle_pnl.add(healthcareCentresTitle_lbl);

JPanel healthcareCentresBtn_pnl = new JPanel();
healthcareCentresBtn_pnl.setBackground(PINK);
healthcareCentresBtn_pnl.setBounds(0, 533, 700, 39);
healthcareCentres_pnl.add(healthcareCentresBtn_pnl);
healthcareCentresBtn_pnl.setLayout(
    new FlowLayout(FlowLayout.RIGHT));

JButton select_btn = new JButton("Select");
select_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
select_btn.addActionListener(e -> {
    iHealthCentre = healthcareCentres_lst.getSelectedIndex();
    selectedHealthcareCentre = pcvs.getPCVSHealthcareCentres()
        .get(iHealthCentre);

    PatientMenuGUI patientMenu = new PatientMenuGUI();
    patientMenu.pack();
    patientMenu.setBounds(0, 0, 1000, 600);
    patientMenu
        .setLocationRelativeTo(ListHealthcareCentresGUI.this);

    // Pass value to PatientMenuGUI.
    patientMenu.setPCVS(pcvs);
    patientMenu.setFile(file);
    patientMenu.setSelHealthCentre(selectedHealthcareCentre);
    patientMenu.setSelectedVaccine(selectedVaccine);
    patientMenu.setIndexHealthcareCentre(iHealthCentre);
    patientMenu.setIndexVaccine(iVaccine);
    patientMenu.setIndexPatient(iPatient);

    patientMenu.setVisible(true);
    setVisible(false);
}

19: JButton back_btn = new JButton("Back");
back_btn.setFont(new Font(".AppleSystemUIFont", PLAIN, 13));
back_btn.addActionListener(e -> {
    ListAvailableVaccinesGUI listAvailableVaccines =
        new ListAvailableVaccinesGUI();
    listAvailableVaccines.pack();
    listAvailableVaccines.setBounds(0, 0, 1000, 600);
}

```

```

        listAvailableVaccines
            .setLocationRelativeTo(ListHealthcareCentresGUI.this);

        listAvailableVaccines.setPCVS(pcv);
        listAvailableVaccines.setFile(file);
        listAvailableVaccines.setIndexPatient(iPatient);

        listAvailableVaccines.setVisible(true);
        setVisible(false);
    });
    healthcareCentresBtn_pnl.add(back_btn);
    healthcareCentresBtn_pnl.add(select_btn);

    JPanel healthcareCentresList_pnl = new JPanel();
    healthcareCentresList_pnl.setLayout(null);
    healthcareCentresList_pnl.setBounds(0, 70, 700, 463);
    healthcareCentres_pnl.add(healthcareCentresList_pnl);

    healthcareCentres_lst = new JList<>(healthcareCentresListModel);
    healthcareCentres_lst.setFont(new Font(".AppleSystemUIFont",
        PLAIN, 12));
    healthcareCentres_lst.setSelectionMode(SINGLE_SELECTION);
    healthcareCentres_lst.setBounds(0, 0, 700, 463);
    healthcareCentresList_pnl.add(healthcareCentres_lst);
}

/**
 * Registers this pcv object to another pcv attribute
 * in another JFrame.
 * This method will save the current pcv object
 * to avoid create new data of pcv.
 *
 * @param pcvObj the current pcv object to save.
 */
public void setPCVS(PCVS pcvObj) {
    this.pcv = pcvObj;
}

/**
 * Registers this file directory to another
 * file attribute in another JFrame.
 * This method will save the current file directory
 * to avoid current file lost.
 *
 * @param directory the directory file to save.
 */
public void setFile(File directory) {
    file = directory;
}

/**
 * Registers this vaccine based on selected vaccine in
 * ListAvailableVaccines GUI to display the available
 * batches of vaccine.
 *
 * @param selVaccine the selected value.
 */

```

```
public void setSelectedVaccine(Vaccine selVaccine) {
    selectedVaccine = selVaccine;
}

/**
 * Registers this index vaccine to gets the object vaccine
 * based on index.
 *
 * @param idx the value to get.
 */
public void setIndexVaccine(int idx) {
    iVaccine = idx;
}

/**
 * Registers this index patient to get
 * the Patient object by index.
 * This method allow recording vaccination appointment
 * to Patient.
 *
 * @param idx patient want to appointment.
 */
public void setIndexPatient(int idx) {
    iPatient = idx;
}

}
```

AppointmentDialog

```
package dialog;
18
import java.awt.*;
import javax.swing.border.EmptyBorder;
import javax.swing.*;
import javax.swing.border.LineBorder;

import static java.awt.BorderLayout.*;
import static java.awt.Color.*;
import static java.awt.Font.*;
import static javax.swing.JOptionPane.*;

/**
 * AppointmentDialog class allow pcvs GUI application to get
 * upcoming date from Patient.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
1
public class AppointmentDialog extends JDialog {

    private final JPanel content_pnl = new JPanel();
    private final JTextField date_tf;

    private String upcomingDate;

    /**
     * Create the appointment dialog.
     */
    public AppointmentDialog(JFrame parent) {
        super(parent, true);
        setTitle("Request Vaccination Appointment");
        setBounds(0, 0, 700, 440);
        getContentPane().setLayout(new BorderLayout());
        content_pnl.setBackground(WHITE);
        content_pnl.setBorder(new EmptyBorder(5, 5, 5, 5));
        getContentPane().add(content_pnl, CENTER);
        content_pnl.setLayout(null);
        {
            JLabel date_lbl = new JLabel("Upcoming Date (mm dd yyyy)");
            date_lbl.setToolTipText("");
            date_lbl.setForeground(DARK_GRAY);
            date_lbl.setFont(
                new Font("Neue Haas Grotesk Text Pro", PLAIN, 14));
            date_lbl.setBounds(219, 136, 199, 18);
            content_pnl.add(date_lbl);
        }
        {
            JPanel date_pnl = new JPanel();
            date_pnl.setLayout(null);
            date_pnl.setOpaque(false);
```

```

        date_pnl.setBorder(new LineBorder(
            new Color(238, 238, 238), 2, true));
        date_pnl.setBounds(219, 166, 262, 40);
        content_pnl.add(date_pnl);
    {
        date_tf = new JTextField();
        date_tf.setForeground(DARK_GRAY);
        date_tf.setFont(new Font(".AppleSystemUIFont",
            PLAIN, 13));
        date_tf.setColumns(10);
        date_tf.setBorder(null);
        date_tf.setBackground(WHITE);
        date_tf.setBounds(6, 6, 250, 28);
        date_pnl.add(date_tf);
    }
}

{
    JPanel btn_pnl = new JPanel();
    btn_pnl.setLayout(new FlowLayout(
        FlowLayout.RIGHT));
    getContentPane().add(btn_pnl, SOUTH);
{
    JButton request_btn = new JButton("Request");
    request_btn.setFont(new Font(".AppleSystemUIFont",
        PLAIN, 13));
    request_btn.addActionListener(e -> {
        upcomingDate = date_tf.getText().trim();
        if (upcomingDate.equals("")) {
            showMessageDialog(
                AppointmentDialog.this,
                "Please fill up all fields",
                null, WARNING_MESSAGE);
            date_tf.requestFocus();
        } else {
            date_tf.setText("");
            setVisible(false);
        }
    });
    btn_pnl.add(request_btn);
    getRootPane().setDefaultButton(request_btn);
}
}

/**
 * Returns valid upcoming date to
 * record the vaccination appointment.
 */
24
* @return value of date in format (mm dd yyyy)
*/
public String getUpcomingDate() {
    return upcomingDate;
}
}

```

AdministratorTableModel

```
package table;
9
import java.util.ArrayList;
import pcvs.Administrator;
import javax.swing.table.AbstractTableModel;

/**
 * AdministratorTableModel class to display
 * this Administrator collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
 */
public class AdministratorTableModel extends AbstractTableModel {

    private static final String[] AdmCol_header = {"Username",
        "Password", "Email", "Full Name", "Staff ID"};
    private final ArrayList<Administrator> administrators;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public AdministratorTableModel(String[] colHeader, int row) {
        administrators = new ArrayList<>();
    }

    /**
     * Returns Administrator in this collection.
     *
     * @return Administrator collection.
     */
    public ArrayList<Administrator> getAdministrators() {
        return administrators;
    }

    /**
     * Returns the number of row in this table.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return administrators.size();
    }

    /**
     * Returns the number of column in this table.
     */
```

```

    * Determine how many column to crate and display by default.
    *
    * @return the num of columns.
    */
@Override
public int getColumnCount() {
    return AdmCol_header.length;
}

/**
 * Returns the attribute value of administrator
 * at column index and row index.
 *
 * @param rowIdx      the queried row.
 * @param colIdx      the queried column.
 * @return Object value of administrator.
15
@Override
public Object getValueAt(int rowIdx, int colIdx) {
    15Administrator adm = getAdministrators().get(rowIdx);
    return switch (colIdx) {
        case 0 -> adm.getUsername();
        case 1 -> adm.getPassword();
        case 2 -> adm.getEmail();
        case 3 -> adm.getFullName();
        case 4 -> adm.getStaffID();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return AdmCol_header[clm];
}

/**
 * Registers administrator object to this collection.
 *
 * @param administrator the object value to registered.
 */
public void add(Administrator administrator) {
    getAdministrators().add(administrator);
    fireTableDataChanged();
}

/**
 * Return administrator object based on row index.
 *
 * @param rowIdx the row value of administrator.
 * @return the administrator object.
 */

```

```
public Administrator getAdministrator(int rowIdx) {
    return getAdministrators().get(rowIdx);
}

/**
 * Remove administrator object in table model
 * based on rowIndex index.
 *
 * @param rowIndex the row value of to remove.
 */
public void remove(int rowIndex) {
    getAdministrators().remove(rowIndex);
    fireTableDataChanged();
}

/**
 * Remove all administrator from table.
 */
public void clear() {
    getAdministrators().clear();
    fireTableDataChanged();
}

/**
 * Returns true if administrator in this collection,
 * false if not in this collection.
 *
 * @param administrator the object value to compare.
 * @return condition value.
 */
public boolean contains(Administrator administrator) {
    return getAdministrators().contains(administrator);
}
}
```

BatchTableModel

```
package table;
13
import javax.swing.table.AbstractTableModel;
import pcvs.Batch;
import java.util.ArrayList;

/**
 * BatchTableModel class defined to display
 * this Batch collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
9*/
public class BatchTableModel extends AbstractTableModel {

    private static final String[] batchCol_header = {"Batch Number",
        "Expiry Date", "Quantity Available"};
    private final ArrayList<Batch> batches;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public BatchTableModel(String[] colHeader, int row) {
        batches = new ArrayList<>();
    }

    /**
     * Returns Batch in this collection.
     *
     * @return Batch collection.
     */
    public ArrayList<Batch> getBatches() {
        return batches;
    }

    /**
     * Returns the number of row in this table model.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return batches.size();
    }

    /**
     * Returns the number of column in this table model.
```

```

    * Determine how many column to crate and display by default.
    *
    * @return the num of columns.
    */
@Override
public int getColumnCount() {
    return batchCol_header.length;
}

/**
 * Returns the attribute value of batch
 * at column index and row index.
 *
 * @param rowIdx      the queried row.
 * @param columnIdx   the queried column.
 * @return Object value of administrator.
 */
@Override
public Object getValueAt(int rowIdx, int columnIdx) {
    Batch bth = getBatches().get(rowIdx);
    return switch (columnIdx) {
        case 0 -> bth.getFirstBatchNo();
        case 1 -> bth.getExpiryDate();
        case 2 -> bth.getQuantityAvailable();
        case 3 -> bth.getQuantityAdministered();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return batchCol_header[clm];
}

/**
 * Registers batch object to this collection.
 *
 * @param batchObj the object value to registered.
 */
public void add(Batch batchObj) {
    getBatches().add(batchObj);
    fireTableDataChanged();
}

/**
 * Remove all batch from table.
 */
public void clear() {
    getBatches().clear();
    fireTableDataChanged();
}

```

```
/**  
 * Returns true if batch in this collection,  
 * false if not in this collection.  
 *  
 * @param batch the object value to compare.  
 * @return condition value.  
 */  
public boolean contains(Batch batch) {  
    return getBatches().contains(batch);  
}  
}
```

PatientTableModel

```
package table;

13 import javax.swing.table.AbstractTableModel;
import pcvs.Patient;
import java.util.ArrayList;

/**
 * PatientTableModel class to display
 * this Patient collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
9 */
public class PatientTableModel extends AbstractTableModel {

    private static final String[] patientCol_header = {"Username",
        "Password", "Email", "Full Name", "IC or Passport"};
    private final ArrayList<Patient> patients;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public PatientTableModel(String[] colHeader, int row) {
        patients = new ArrayList<>();
    }

    public ArrayList<Patient> getPatients() {
        return patients;
    }

    /**
     * Returns the number of row in this table.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return patients.size();
    }

    /**
     * Returns the number of column in this table.
     * Determine how many column to create and display by default.
     *
     * @return the num of columns.
     */
}
```

```

@Override
public int getColumnCount() {
    return patientCol_header.length;
}

/**
 * Returns the attribute value of administrator
 * at column index and row index.
 *
 * @param rowIndex      the queried row.
 * @param columnIndex   the queried column.
 * @return Object value of administrator.
 */
@Override
public Object getValueAt(int rowIndex, int columnIndex) {
    Patient pt = getPatients().get(rowIndex);
    return switch (columnIndex) {
        case 0 -> pt.getUsername();
        case 1 -> pt.getPassword();
        case 2 -> pt.getEmail();
        case 3 -> pt.getFullName();
        case 4 -> pt.getICPassport();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return patientCol_header[clm];
}

/**
 * Registers patient object to this collection.
 *
 * @param patient the object value to registered.
 */
public void add(Patient patient) {
    getPatients().add(patient);
    fireTableDataChanged();
}

/**
 * Return administrator object based on row index.
 *
 * @param rowIdx the row value of administrator.
 * @return the administrator object.
 */
public Patient getPatient(int rowIdx) {
    return getPatients().get(rowIdx);
}

```

```
/**  
 * Remove administrator object in table model  
 * based on rowIndex index.  
 *  
 * @param rowIdx the row value of to remove.  
 */  
public void remove(int rowIdx) {  
    getPatients().remove(rowIdx);  
    fireTableDataChanged();  
}  
  
/**  
 * Remove all patient from table.  
 */  
public void clear() {  
    getPatients().clear();  
    fireTableDataChanged();  
}  
  
/**  
 * Returns true if patient in this collection,  
 * false if not in this collection.  
 *  
 * @param patient the object value to compare.  
 * @return condition value.  
 */  
public boolean contains(Patient patient) {  
    return getPatients().contains(patient);  
}  
}
```

VaccinationListTableModel

```
package table;
13
import javax.swing.table.AbstractTableModel;
import pcvs.Vaccination;
import java.util.ArrayList;

/**
 * VaccinationListTableModel class to display
 * this Vaccination collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
9*/
public class VaccinationListTableModel extends AbstractTableModel {

    private static final String[] vacColHeader = {"Vaccination ID",
        "Appointment Date", "Status"};
    private final ArrayList<Vaccination> vaccinations;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public VaccinationListTableModel(String[] colHeader, int row) {
        vaccinations = new ArrayList<>();
    }

    /**
     * Returns vaccination in this collection.
     *
     * @return vaccination collection.
     */
    public ArrayList<Vaccination> getVaccinations() {
        return vaccinations;
    }

    /**
     * Returns the number of row in this table.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return vaccinations.size();
    }

    /**

```

```

    * Returns the number of column in this table.
    * Determine how many column to create and display by default.
    *
    * @return the num of columns.
    */
@Override
public int getColumnCount() {
    return vacColHeader.length;
}

/**
 * Returns the attribute value of Vaccination
 * at column index and row index.
 *
 * @param rowIdx the queried row.
 * @param colIdx the queried column.
 * @return Object value of vaccination.
 */
@Override
public Object getValueAt(int rowIdx, int colIdx) {
    Vaccination vacc = getVaccinations().get(rowIdx);
    return switch (colIdx) {
        case 0 -> vacc.getVaccinationID();
        case 1 -> vacc.getAppointmentDate();
        case 2 -> vacc.getStatus();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return vacColHeader[clm];
}

/**
 * Registers vaccination object to this collection.
 *
 * @param vaccination the object value to registered.
 */
public void add(Vaccination vaccination) {
    getVaccinations().add(vaccination);
    fireTableDataChanged();
}

/**
 * Remove all vaccination from table.
 */
public void clear() {
    getVaccinations().clear();
    fireTableDataChanged();
}

```

```
/**  
 * Returns true if vaccination in this collection,  
 * false if not in this collection.  
 *  
 * @param vaccination the object value to compare.  
 * @return condition value.  
 */  
public boolean contains(Vaccination vaccination) {  
    return getVaccinations().contains(vaccination);  
}  
}
```

VaccinationTableModel

```
package table;
13
import javax.swing.table.AbstractTableModel;
import pcvs.Vaccination;
import java.util.ArrayList;

/**
 * VaccinationListTableModel class to display
 * this Vaccination collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
9*/
public class VaccinationTableModel extends AbstractTableModel {

    private static final String[] vacColHeader = {"Vaccination ID",
        "Appointment Date", "Status", "Remarks"};
    private final ArrayList<Vaccination> vaccinations;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public VaccinationTableModel(String[] colHeader, int row) {
        vaccinations = new ArrayList<>();
    }

    /**
     * Returns vaccination in this collection.
     *
     * @return vaccination collection.
     */
    public ArrayList<Vaccination> getVaccinations() {
        return vaccinations;
    }

    /**
     * Returns the number of row in this table.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return vaccinations.size();
    }

    /**

```

```

    * Returns the number of column in this table.
    * Determine how many column to crate and display by default.
    *
    * @return the num of columns.
    */
@Override
public int getColumnCount() {
    return vacColHeader.length;
}

/**
 * Returns the attribute value of Vaccination
 * at column index and row index.
 *
 * @param rowIdx the queried row.
 * @param colIdx the queried column.
 * @return Object value of vaccination.
 */
@Override
public Object getValueAt(int rowIdx, int colIdx) {
    Vaccination vacc = getVaccinations().get(rowIdx);
    return switch (colIdx) {
        case 0 -> vacc.getVaccinationID();
        case 1 -> vacc.getAppointmentDate();
        case 2 -> vacc.getStatus();
        case 3 -> vacc.getRemarks();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return vacColHeader[clm];
}

/**
 * Registers vaccination object to this collection.
 *
 * @param vaccination the object value to registered.
 */
public void add(Vaccination vaccination) {
    getVaccinations().add(vaccination);
    fireTableDataChanged();
}

/**
 * Remove all vaccination from table.
 */
public void clear() {
    getVaccinations().clear();
    fireTableDataChanged();
}

```

```
}

/**
 * Returns true if vaccination in this collection,
 * false if not in this collection.
 *
 * @param vaccination the object value to compare.
 * @return condition value.
 */
public boolean contains(Vaccination vaccination) {
    return getVaccinations().contains(vaccination);
}
```

VaccineBatchTableModel

```
package table;
13
import javax.swing.table.AbstractTableModel;
import pcvs.Batch;
import java.util.ArrayList;

/**
 * VaccineBatchTableModel class defined to display
 * this Batch collection to table model.
 *
 * @author I Nyoman Surya Pradipta
 * Student ID: E1900344
 * Date: 01-06-2021
 * Java version: java 17 2021-09-14 LTS
 * IDE : IntelliJ IDEA & Eclipse
9*/
public class VaccineBatchTableModel extends AbstractTableModel {
    private static final String[] batchColHeader = {"Batch Number",
        "Expiry Date", "Quantity Available"};
    private final ArrayList<Batch> batches;

    /**
     * Constructor specifying attribute of table model to create.
     * Allow to enter a title in the table model.
     *
     * @param colHeader the number of column.
     * @param row       the value of row count.
     */
    public VaccineBatchTableModel(String[] colHeader, int row) {
        batches = new ArrayList<>();
    }

    /**
     * Returns Batch in this collection.
     *
     * @return Batch collection.
     */
    public ArrayList<Batch> getBatches() {
        return batches;
    }

    /**
     * Returns the number of row in this table model.
     *
     * @return the num of rows.
     */
    @Override
    public int getRowCount() {
        return batches.size();
    }

    /**
     * Returns the number of column in this table model.
     * Determine how many column to create and display by default.
     */
```

```

/*
 * @return the num of columns.
 */
@Override
public int getColumnCount() {
    return batchColHeader.length;
}

/**
 * Returns the attribute value of batch
 * at column index and row index.
 *
 * @param rowIdx      the queried row.
 * @param columnIdx   the queried column.
 * @return Object value of administrator.
 */
@Override
public Object getValueAt(int rowIdx, int columnIdx) {
    Batch bth = getBatches().get(rowIdx);
    return switch (columnIdx) {
        case 0 -> bth.getBatchNo();
        case 1 -> bth.getExpiryDate();
        case 2 -> bth.getQuantityAvailable();
        case 3 -> bth.getQuantityAdministered();
        default -> "";
    };
}

/**
 * Returns the column header using spreadsheet conventions.
 *
 * @param clm the queried column.
 * @return a string of column header.
 */
@Override
public String getColumnName(int clm) {
    return batchColHeader[clm];
}

/**
 * Registers batch object to this collection.
 *
 * @param batch the object value to registered.
 */
public void add(Batch batch) {
    getBatches().add(batch);
    fireTableDataChanged();
}

/**
 * Remove all batch from table.
 */
public void clear() {
    getBatches().clear();
    fireTableDataChanged();
}

```

```
    /**
     * Returns true if batch in this collection,
     * false if not in this collection.
     *
     * @param batch the object value to compare.
     * @return condition value.
     */
    public boolean contains(Batch batch) {
        return getBatches().contains(batch);
    }
}
```

Interface Design

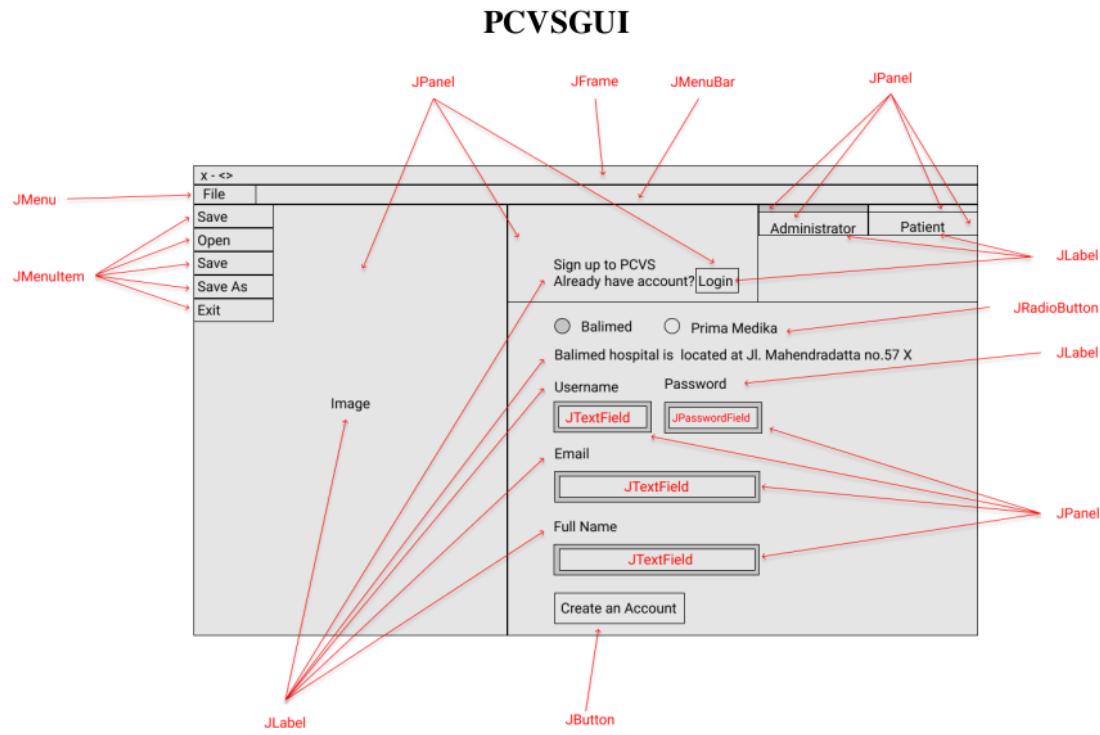


Figure 1: SignUpGUI for Administrator Interface Design Sketch

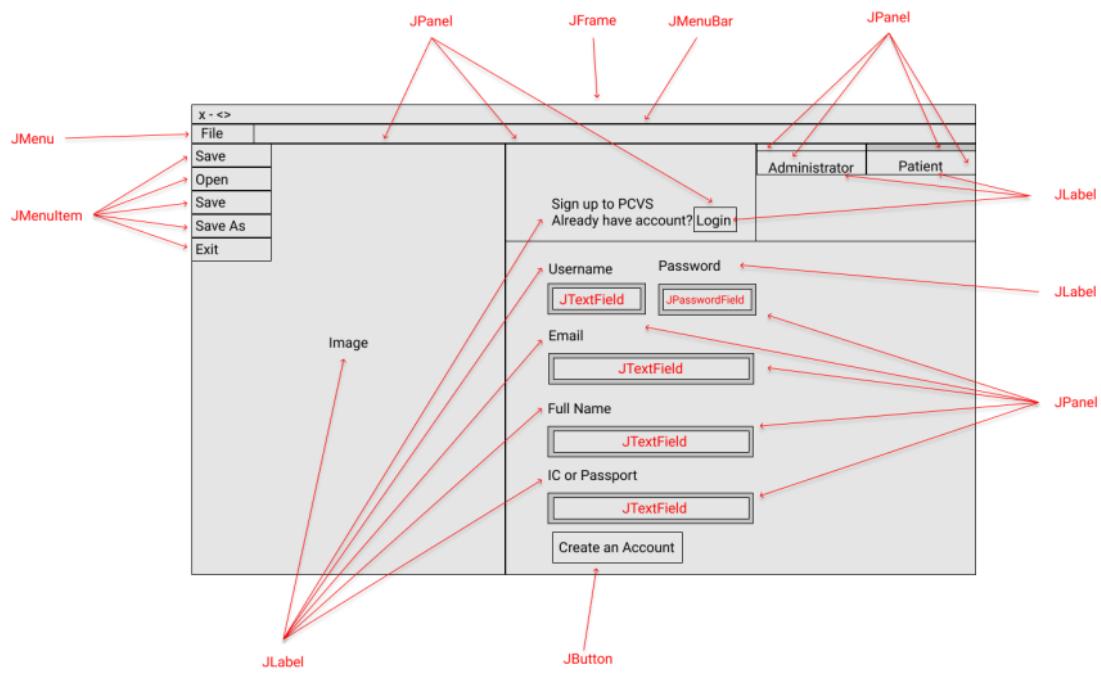


Figure 2: SignUpGUI for Patient Interface Design Sketch

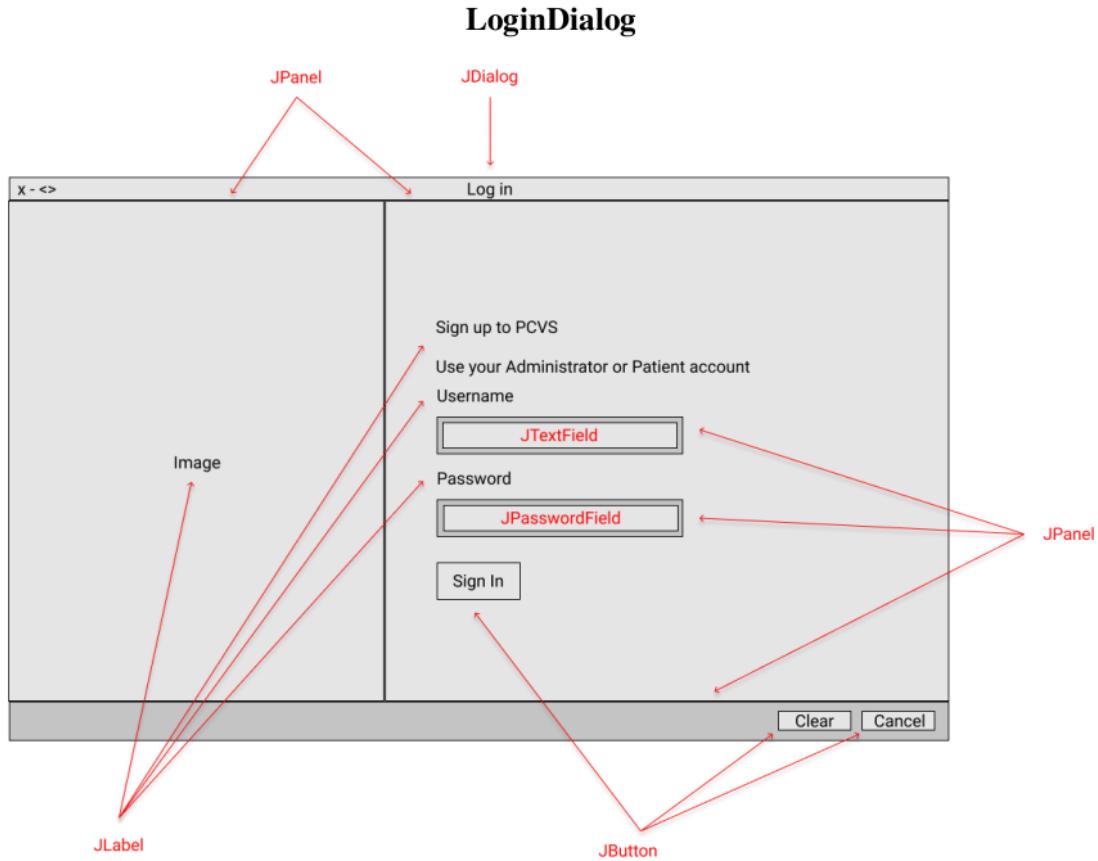


Figure 1: LoginDialog Interface Design Sketch

AdminMenuGUI

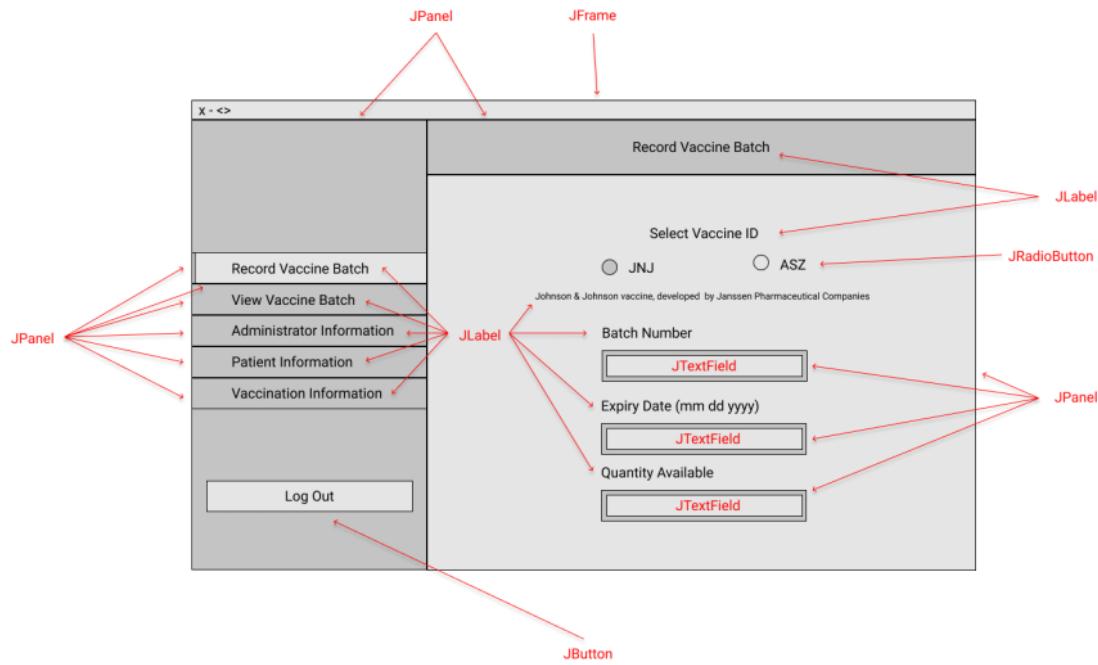


Figure 1: AdminMenuGUI Interface Design Sketch

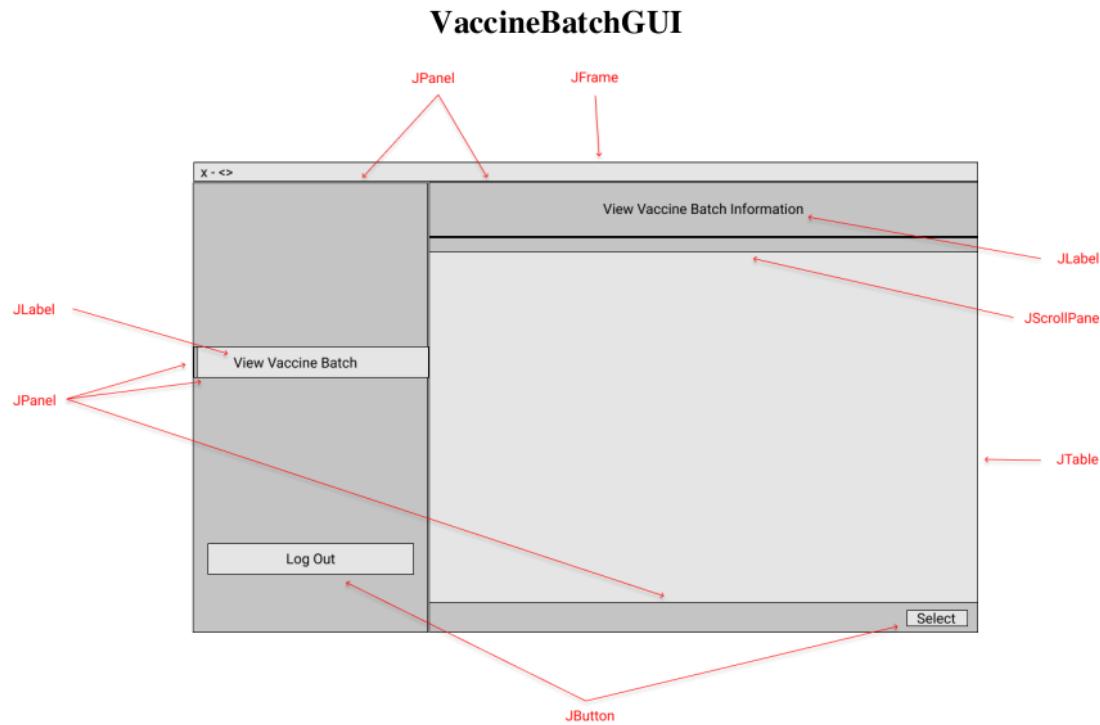


Figure 1: VaccineBatchGUI Interface Design Sketch

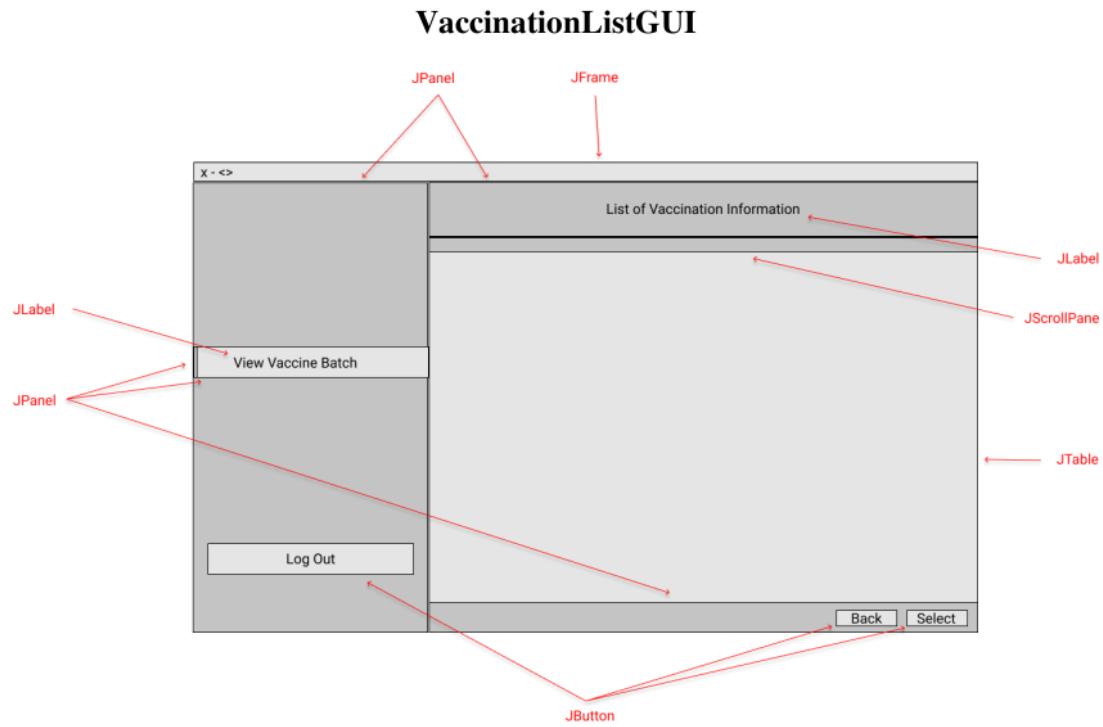


Figure 1: VaccinationListGUI Interface Design Sketch

AppointmentInformationGUI

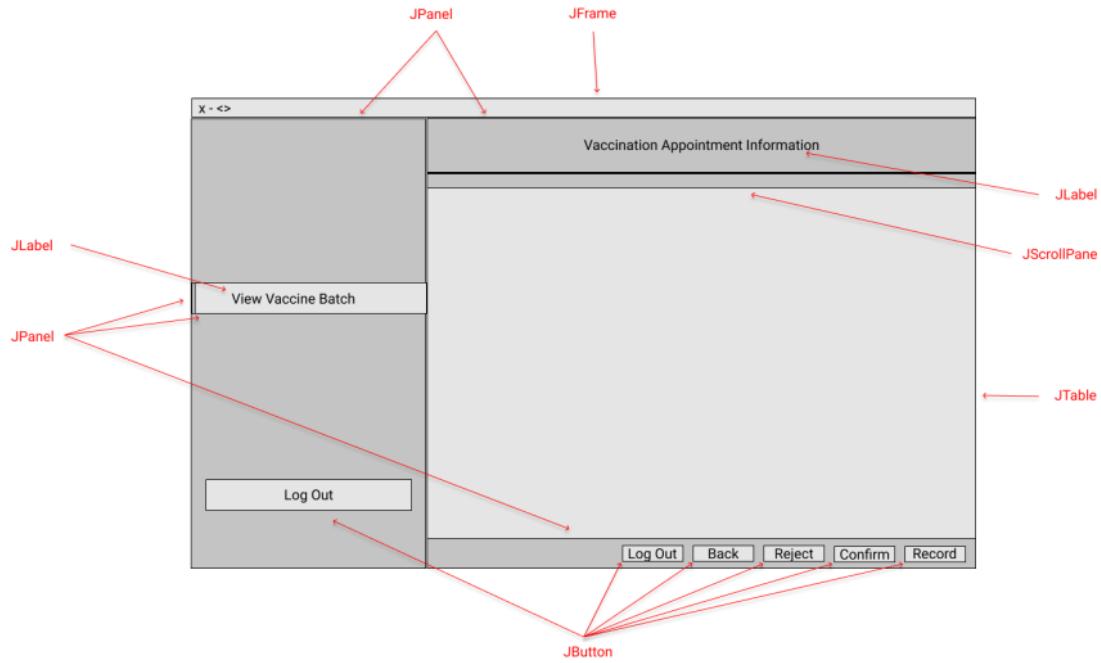


Figure 1: AppointmentInformationGUI Interface Design Sketch

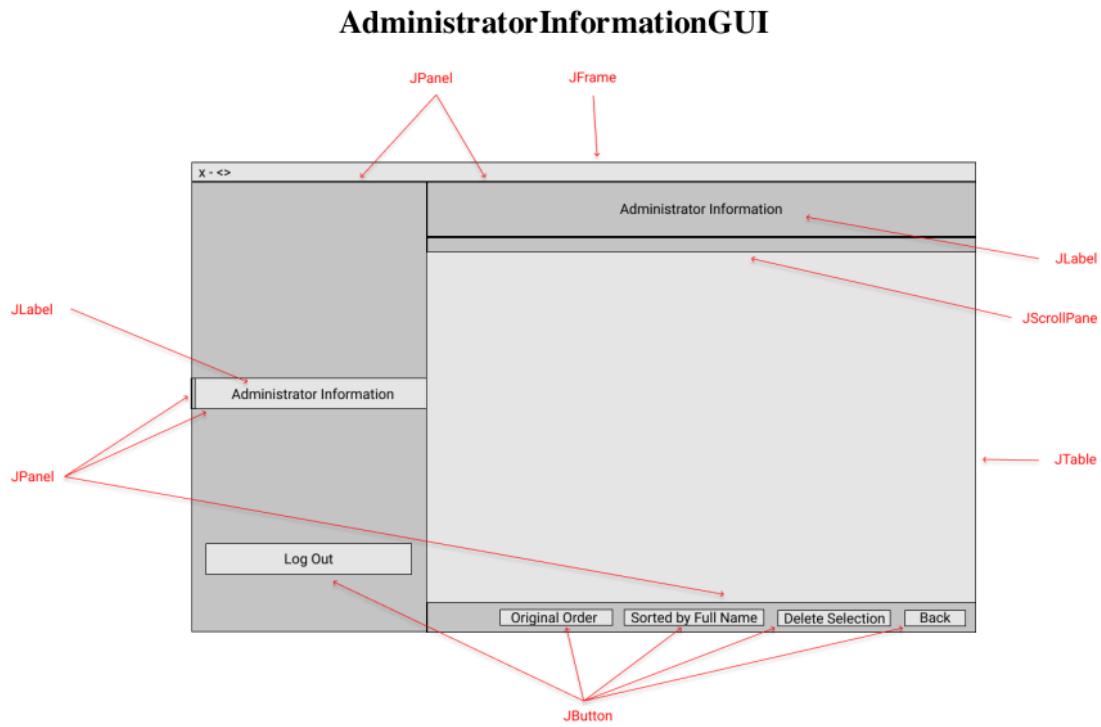


Figure 1: AdministratorInformationGUI Interface Design Sketch

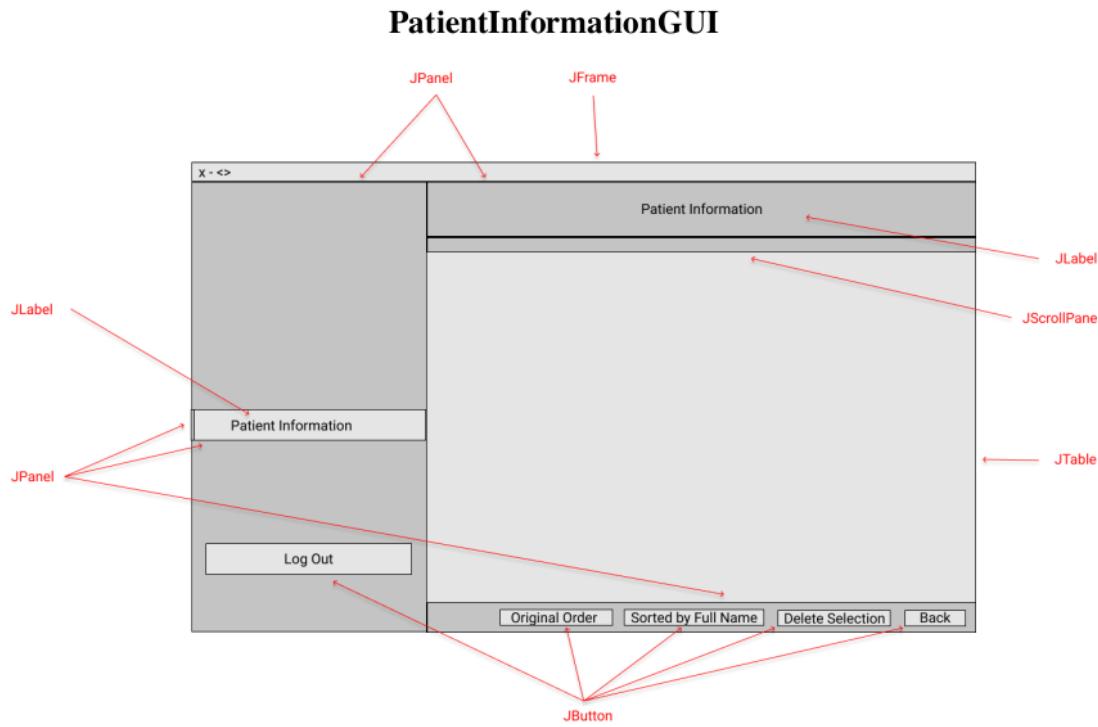


Figure 1: PatientInformationGUI Interface Design Sketch

VaccinationInformationGUI

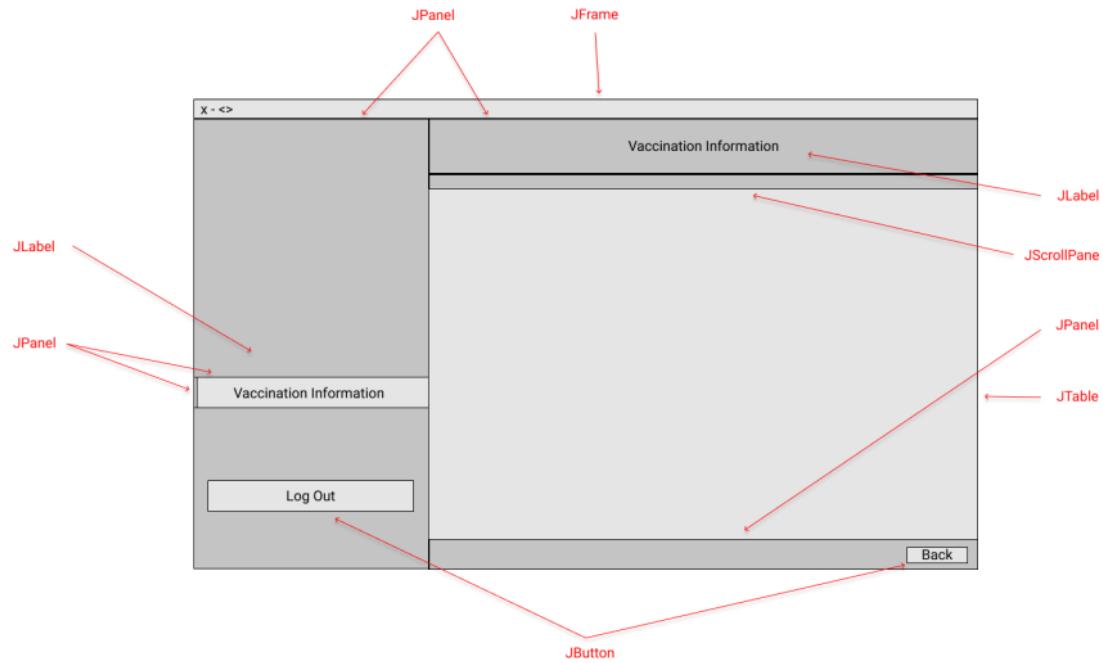


Figure 1: VaccinationInformationGUI Interface Design Sketch

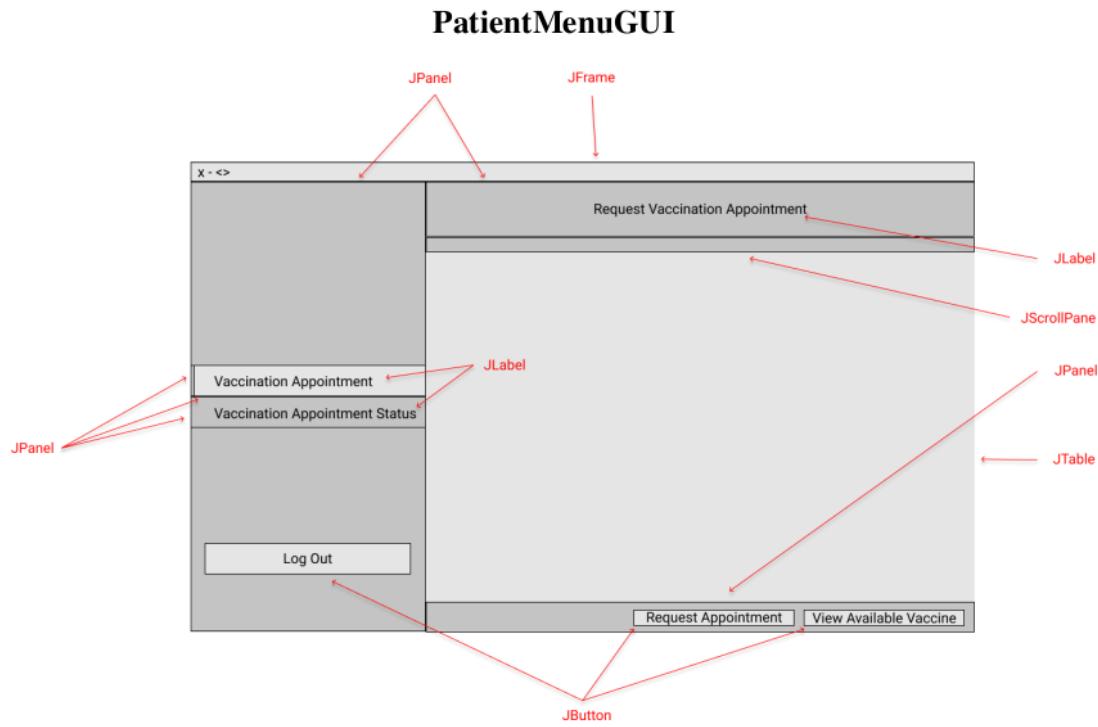


Figure 1: PatientMenuGUI Interface Design Sketch

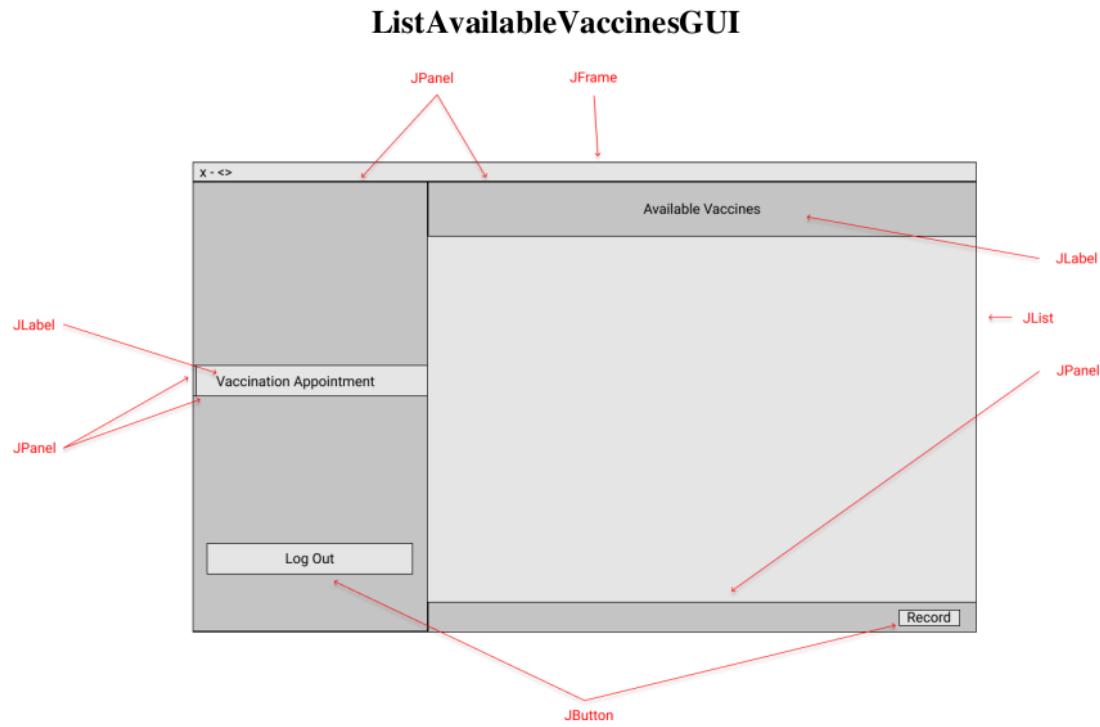


Figure 1: ListAvailableVaccinesGUI Interface Design Sketch

ListHealthcareCentresGUI

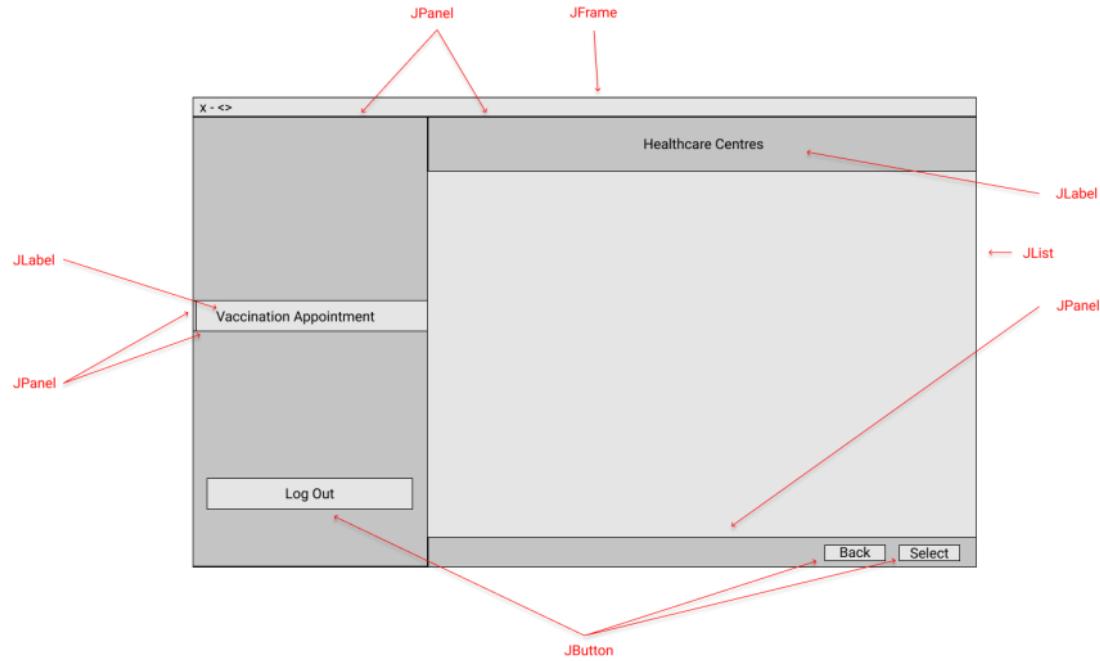


Figure 1: ListHealthcareCentresGUI Interface Design Sketch

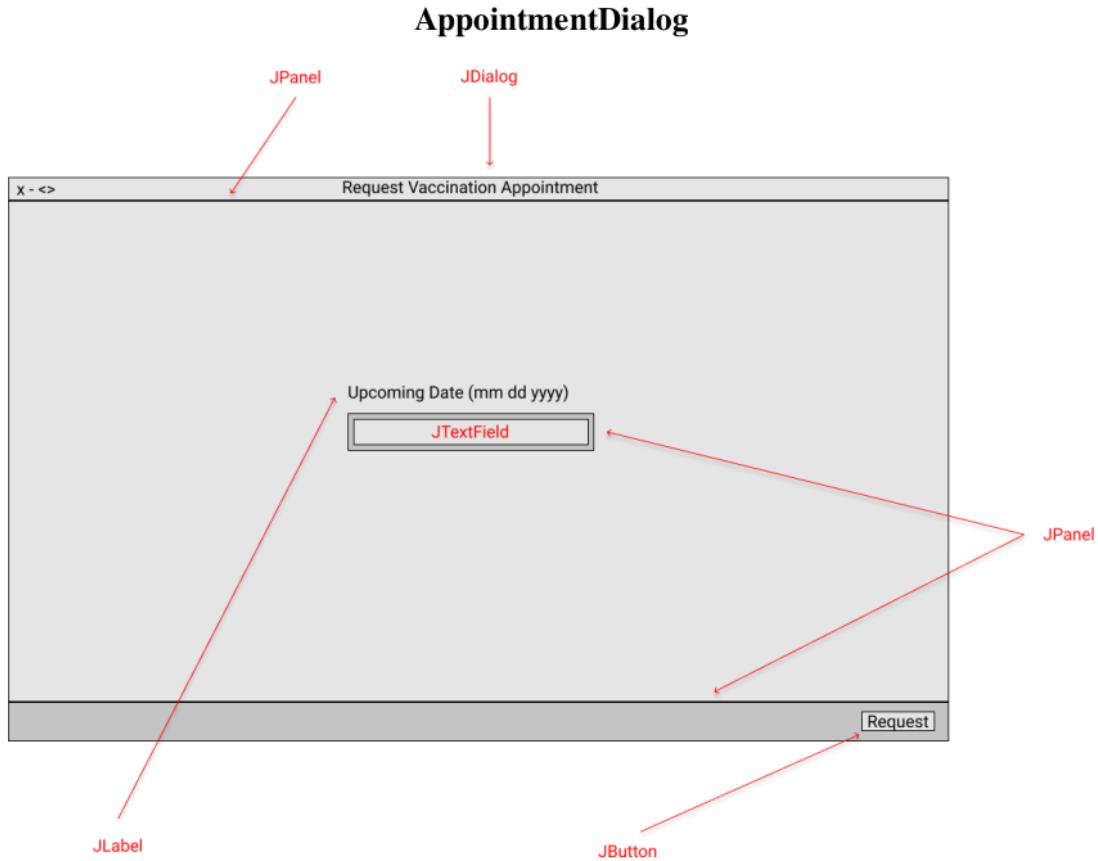
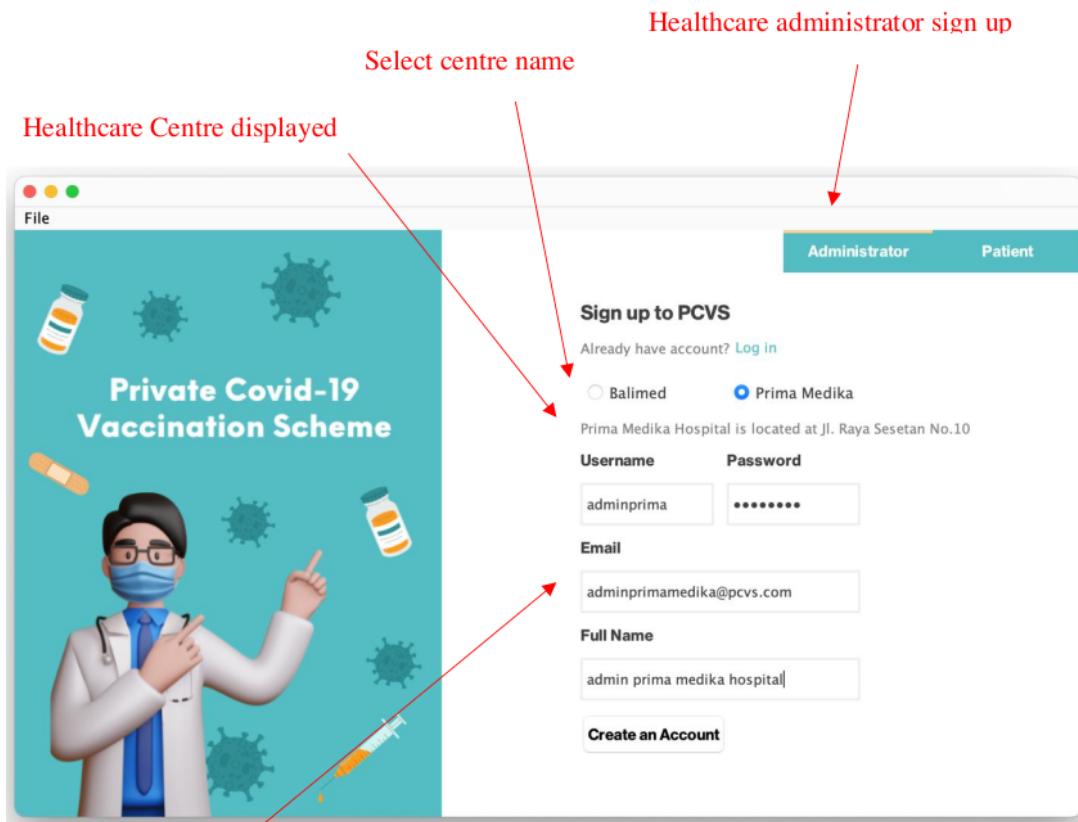


Figure 1: AppointmentDialog Interface Design Sketch

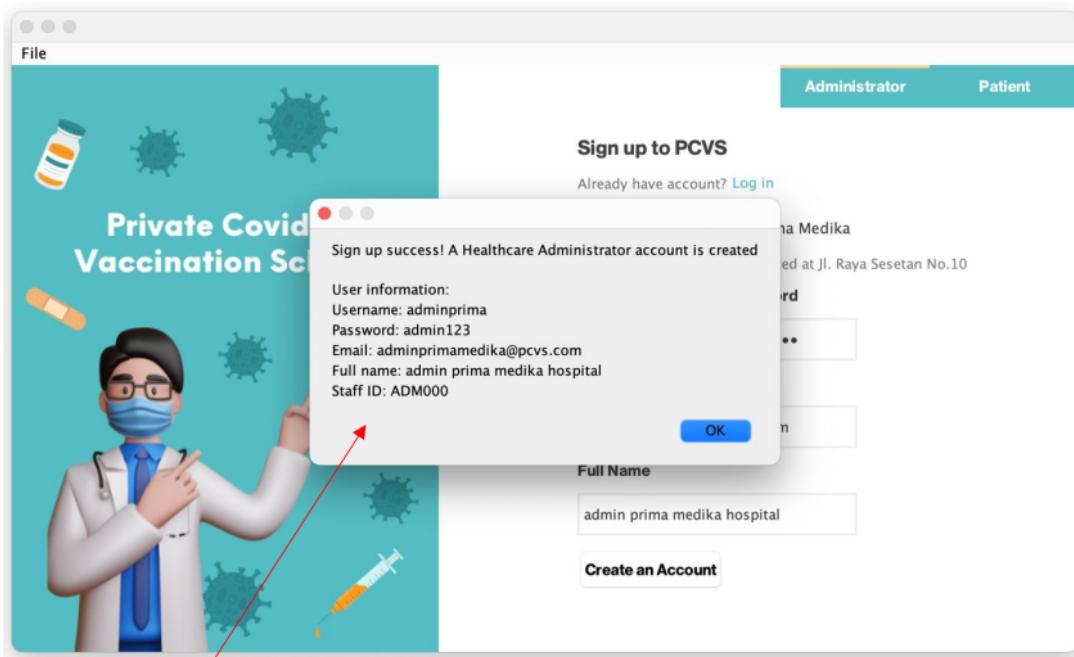
Sample Output

Use Case 1 Sign Up



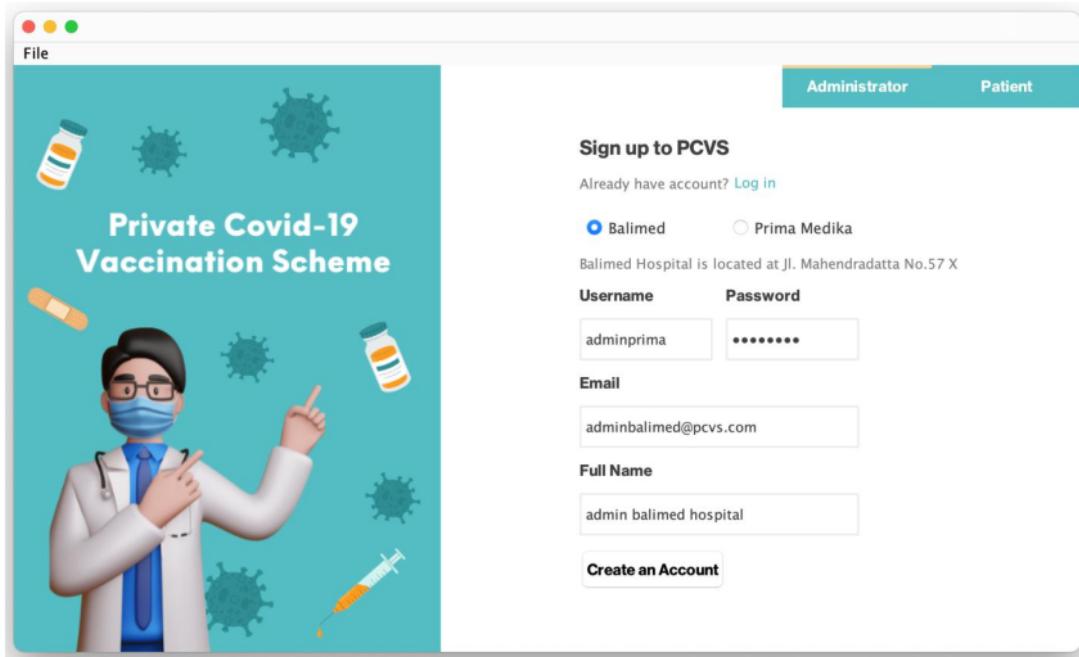
Picture 1: Administrator Sign Up

Administrator enter information



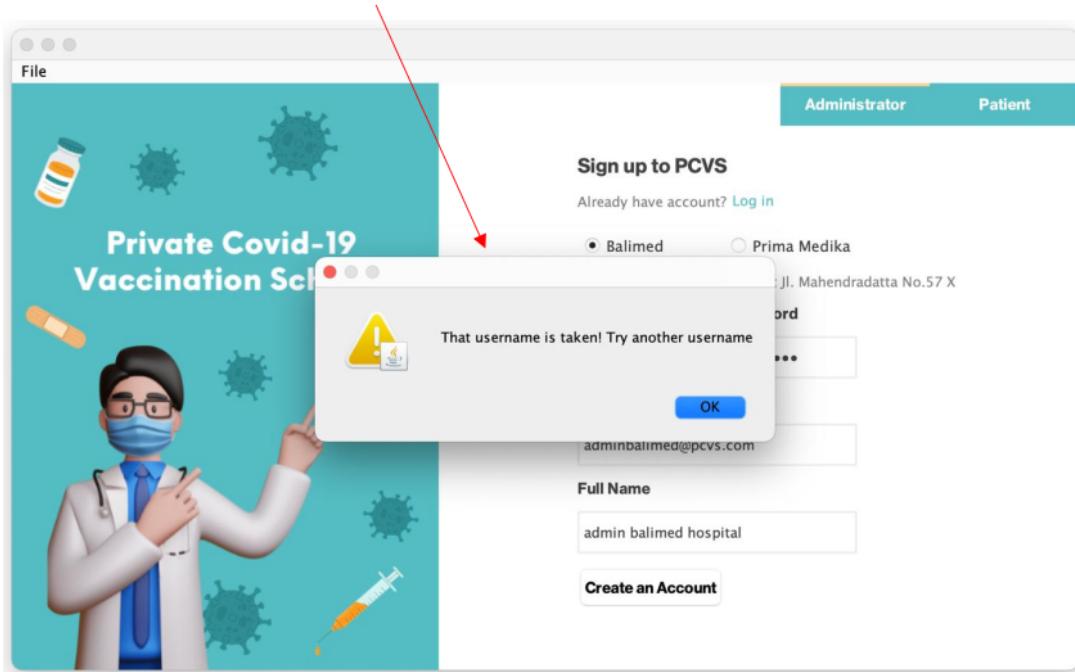
Picture 1.1: Administrator Sign Up

Administrator sign up success and staff ID automatically generated.

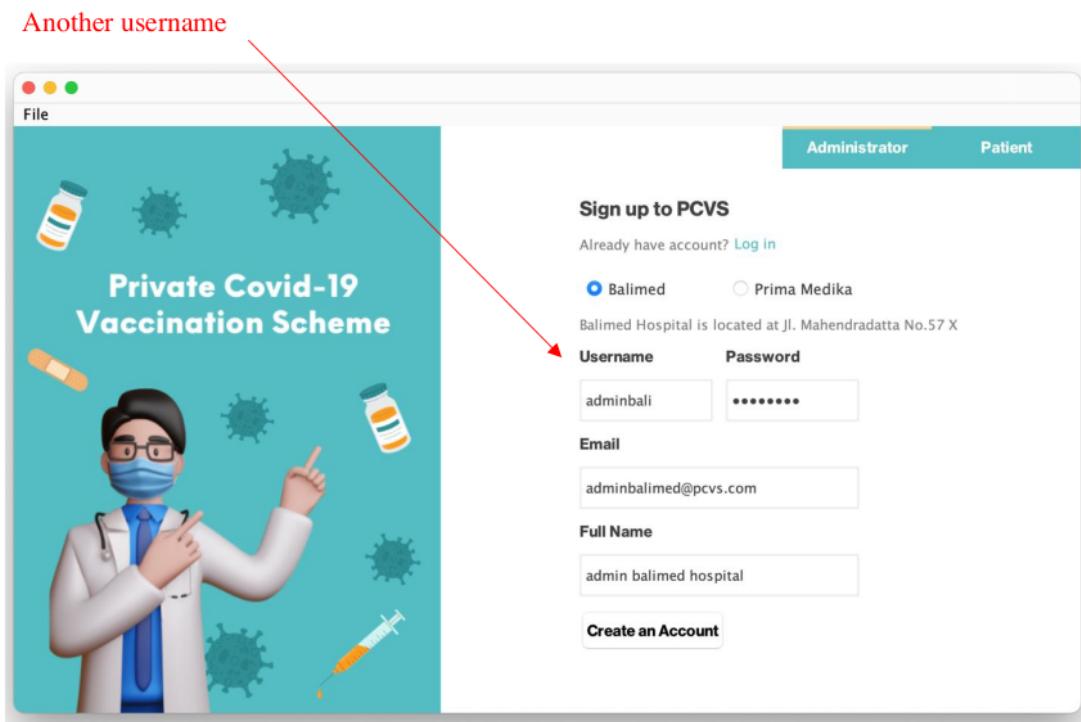


Picture 2: Administrator Sign Up Username Validation

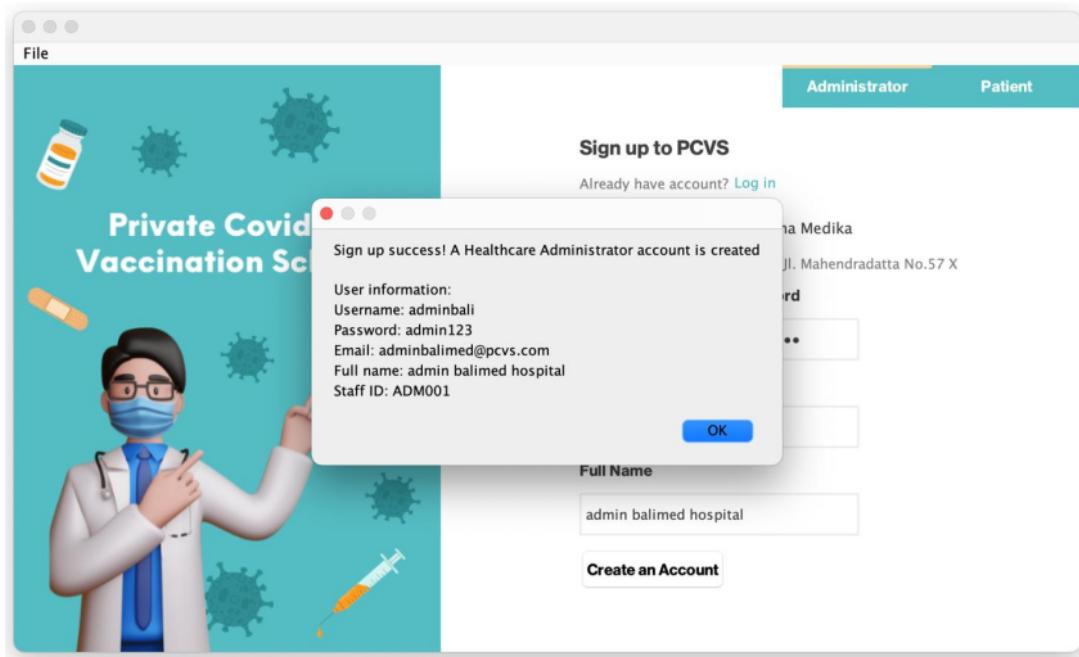
Checking duplicate username



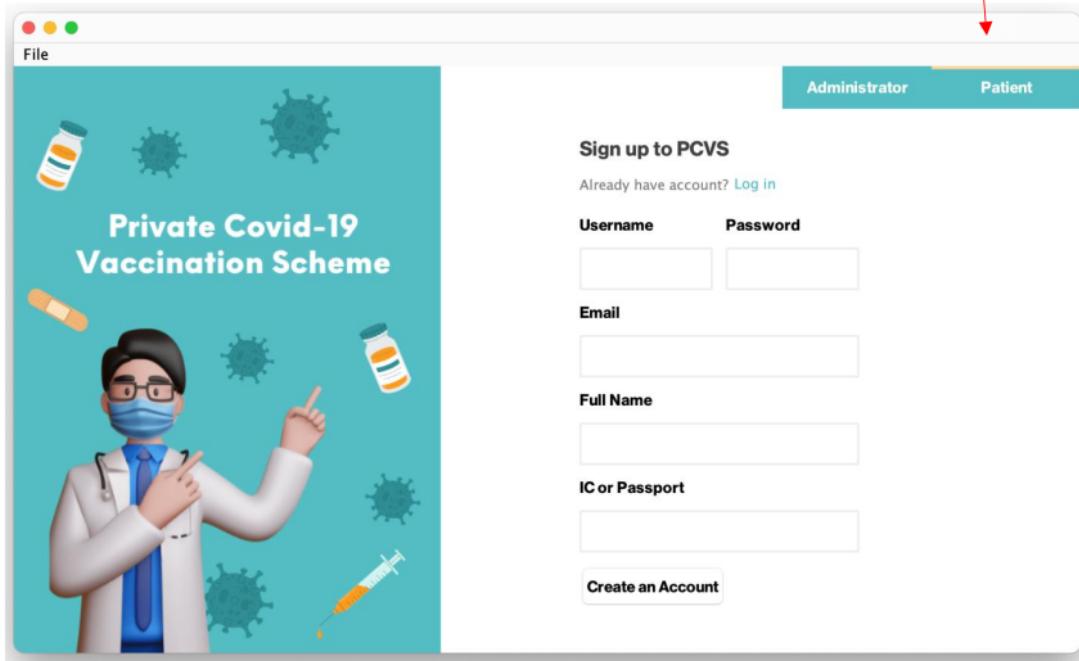
Picture 2.1: Administrator Sign Up Username Validation



Picture 2.2: Administrator Sign Up

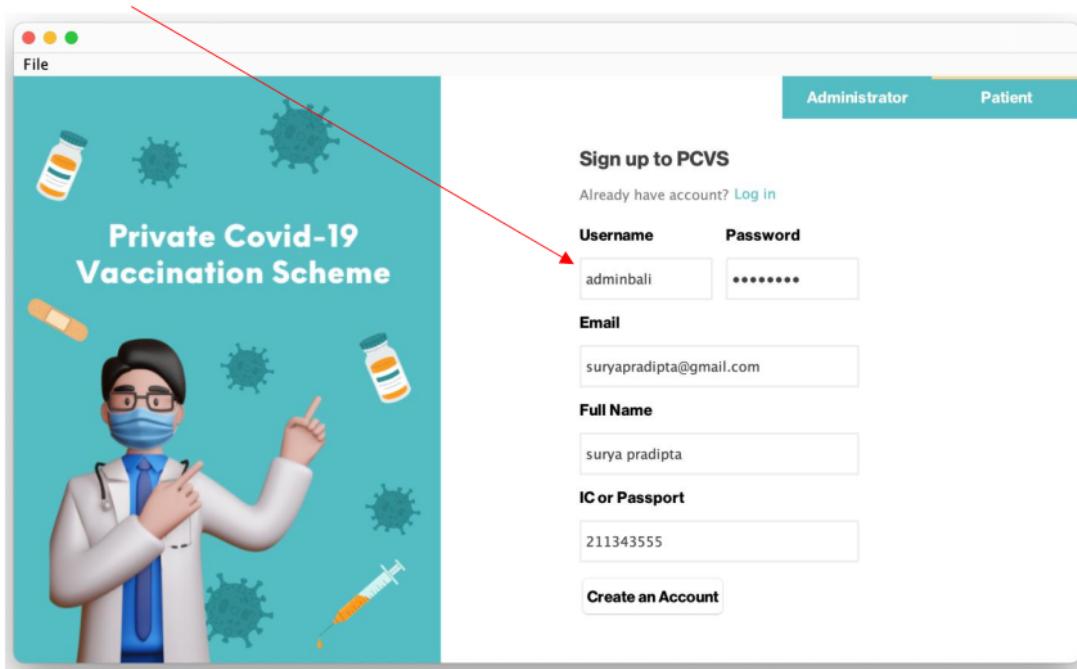


Picture 2.3: Administrator Sign Up

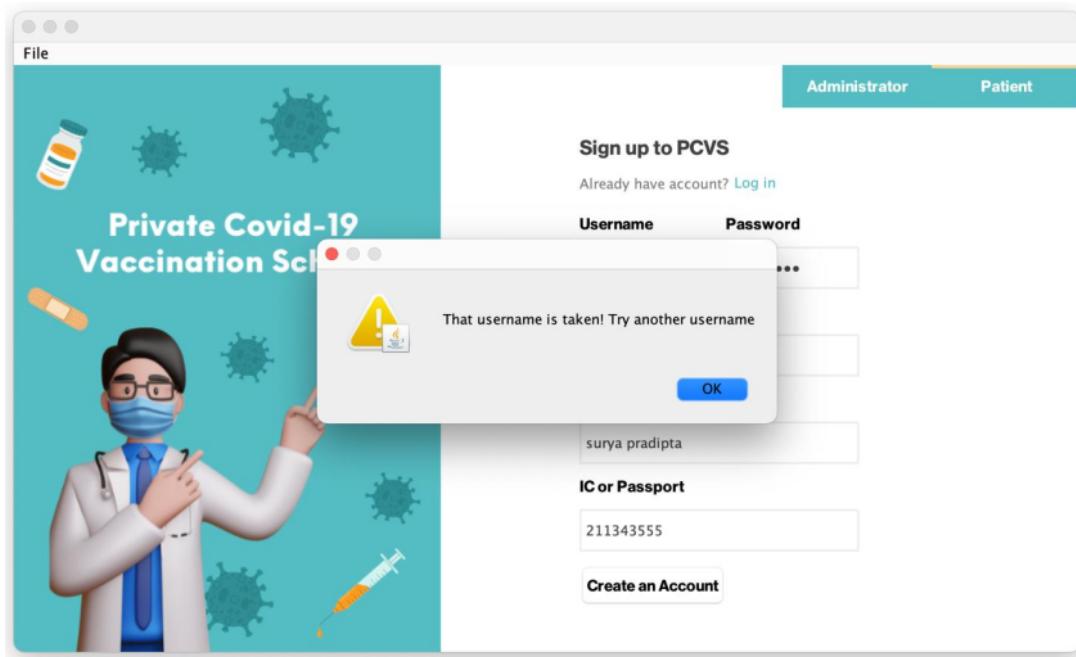


Picture 3: Patient Sign Up

Checking duplicate username

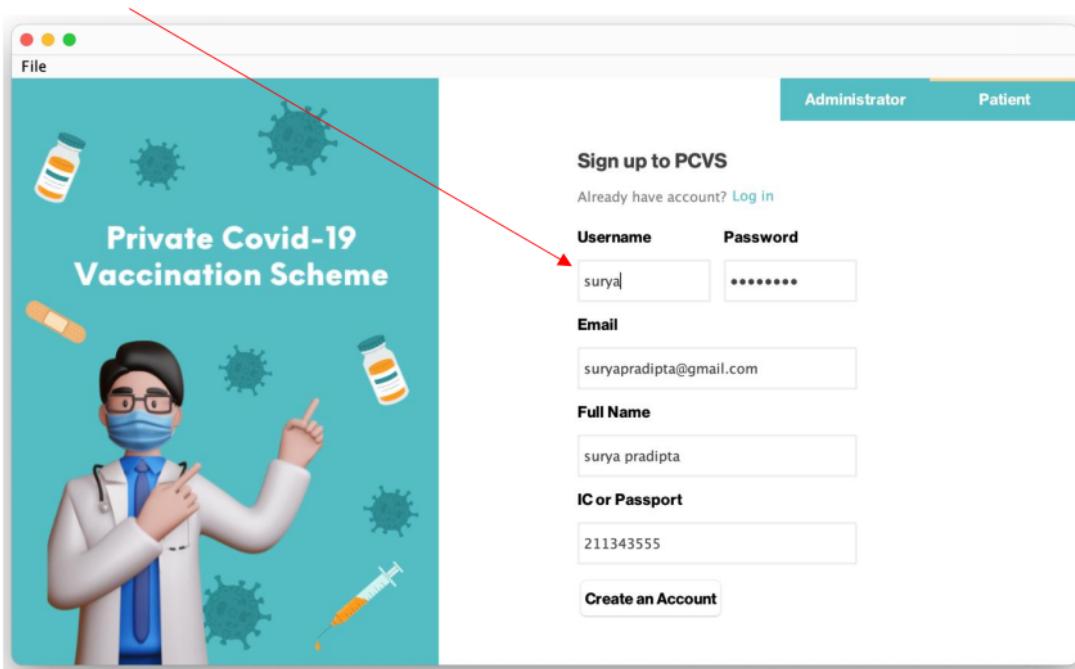


Picture 3.1: Patient Sign Up Username Validation

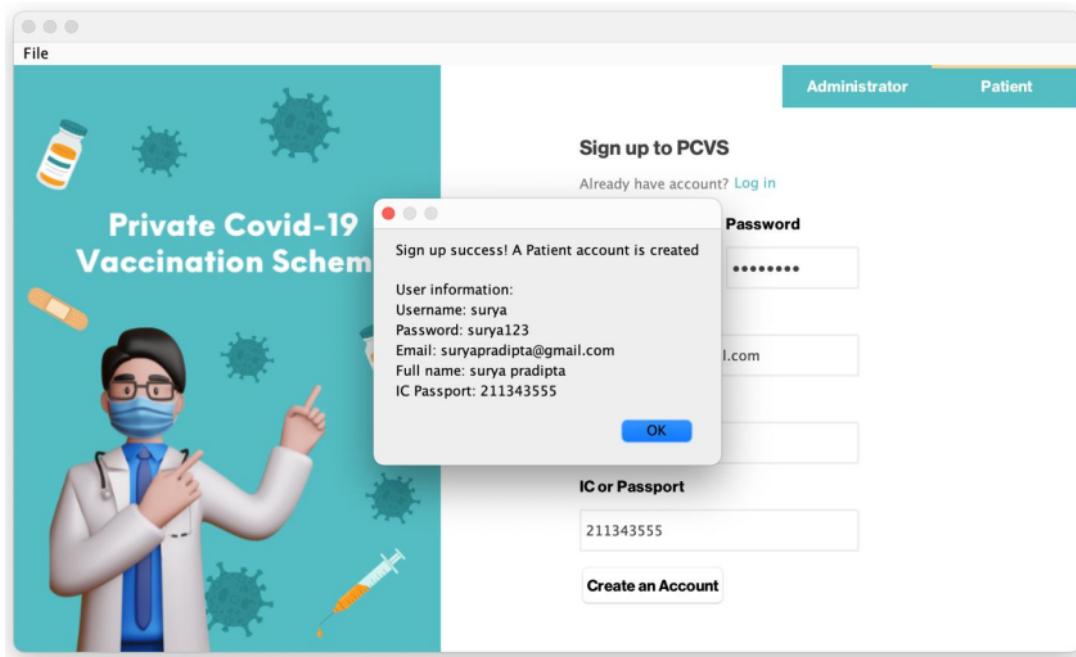


Picture 3.2: Patient Sign Up Username Validation

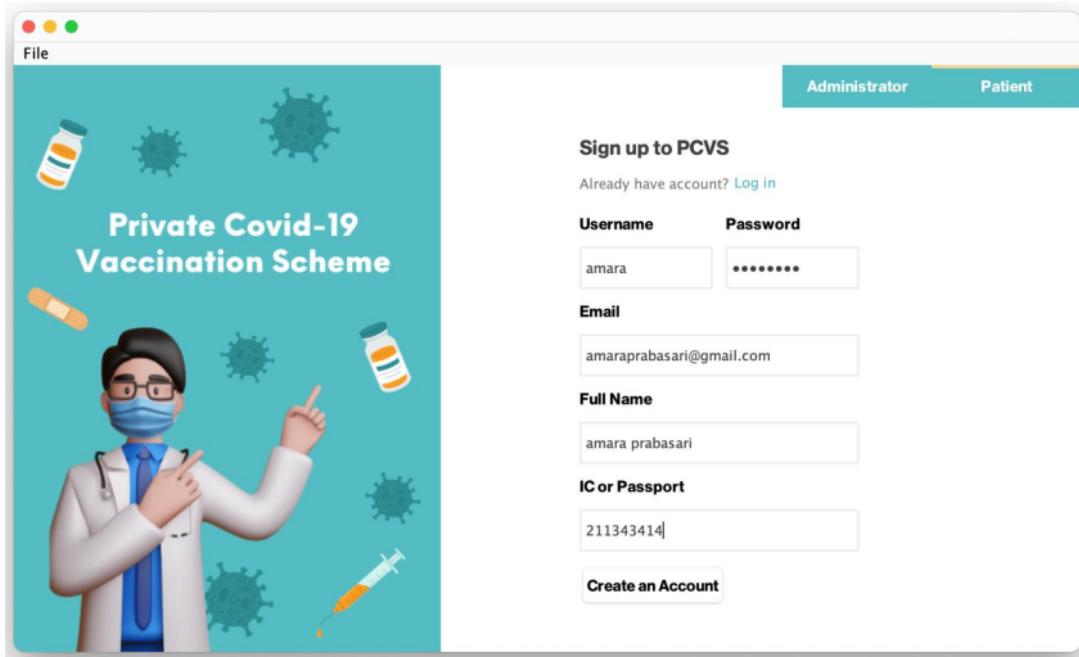
Another username



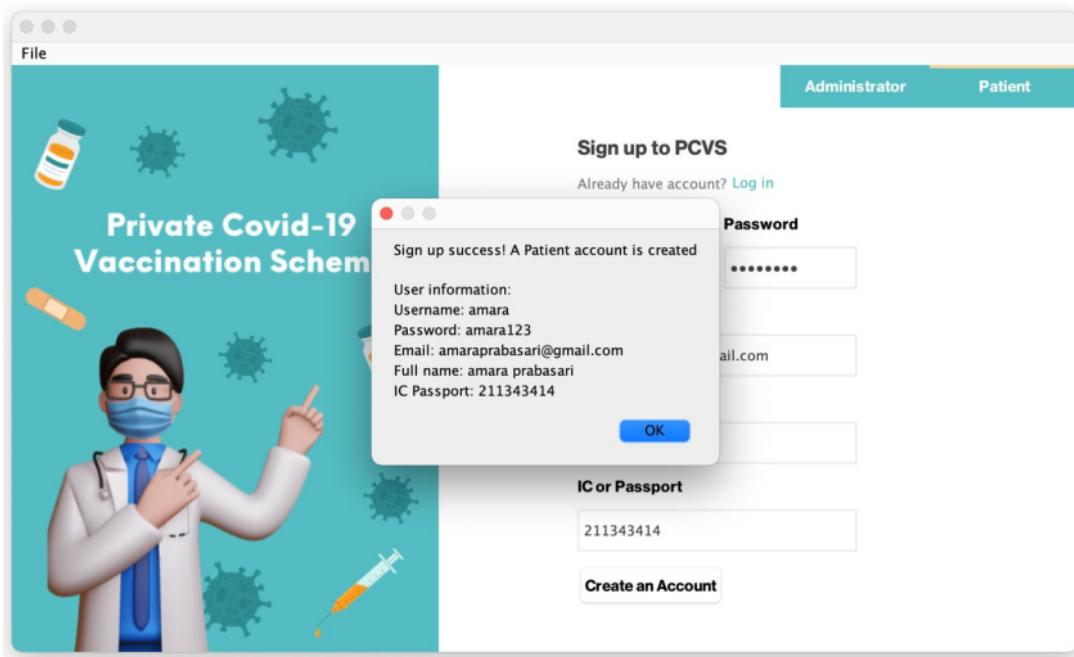
Picture 3.3: Patient Sign Up Username Validation



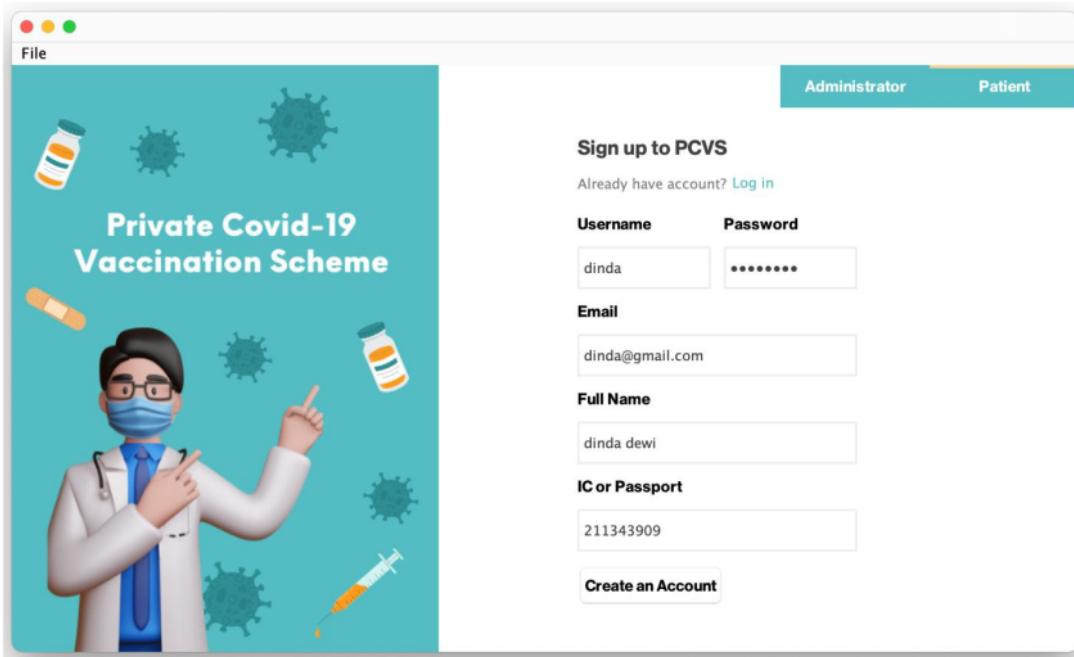
Picture 3.4: Patient Sign Up Username Validation



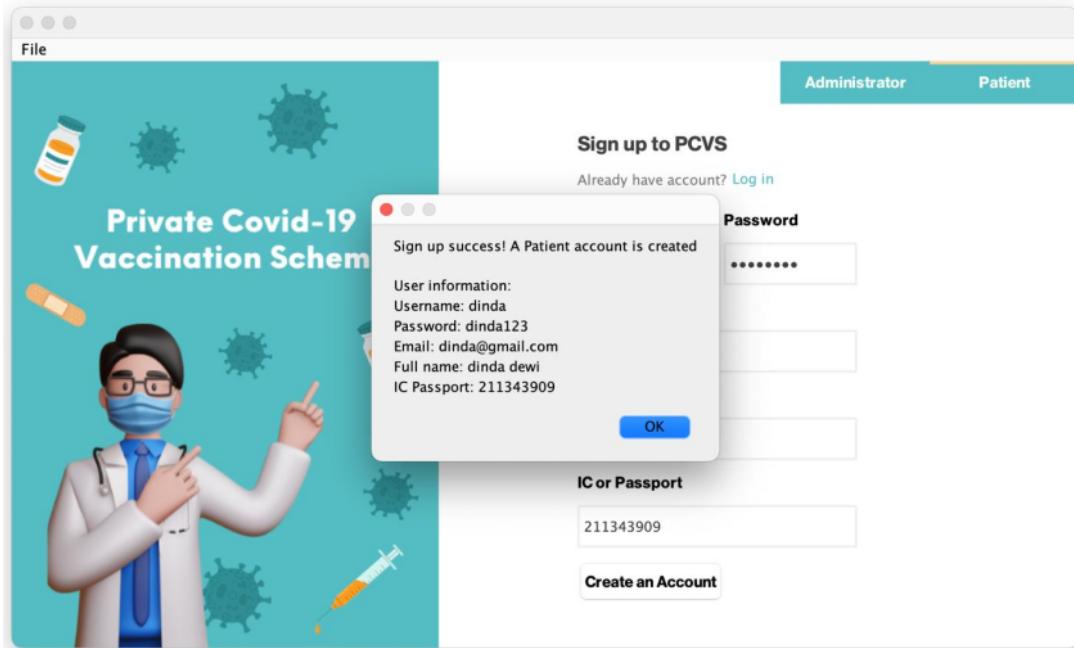
Picture 4: Patient Sign Up



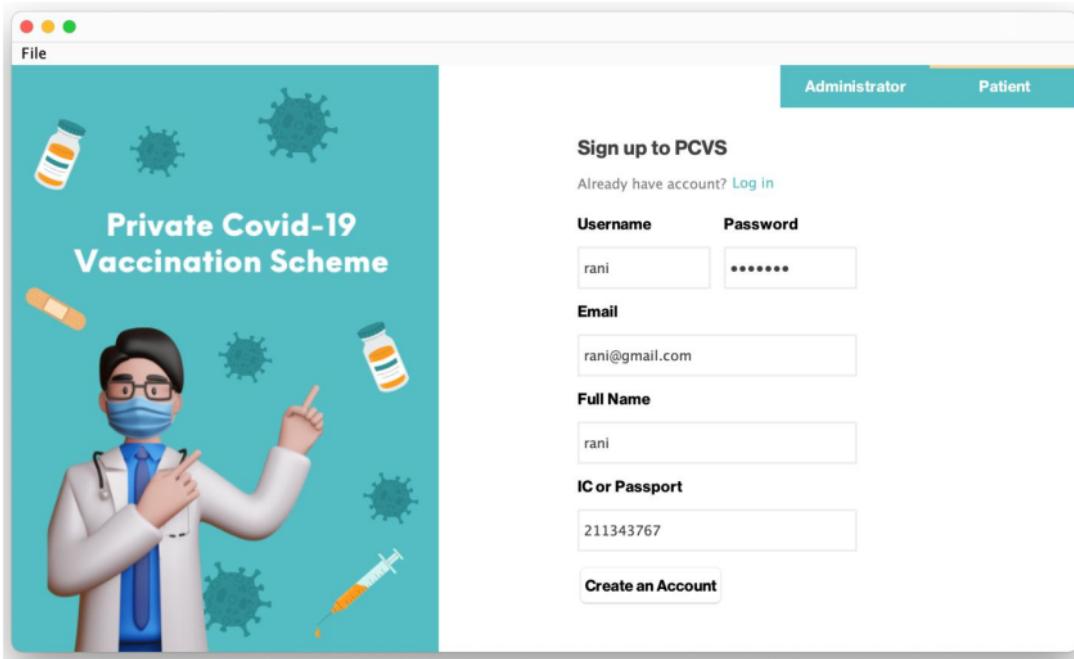
Picture 4.1: Patient Sign Up



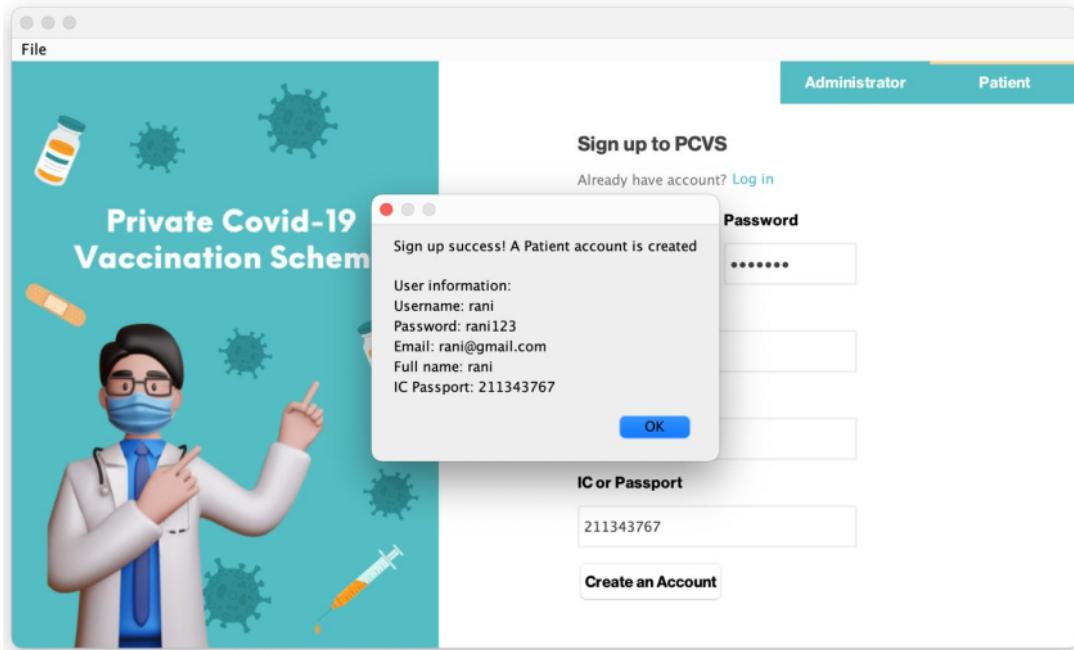
Picture 5: Patient Sign Up



Picture 5.1: Patient Sign Up



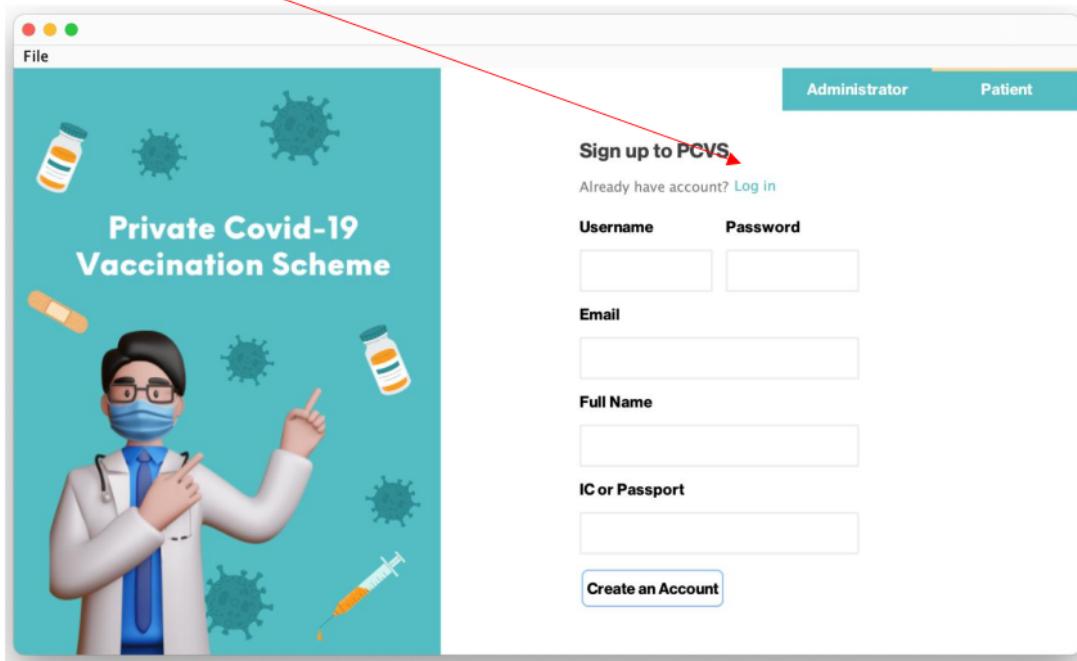
Picture 6: Patient Sign Up



Picture 6.1: Patient Sign Up

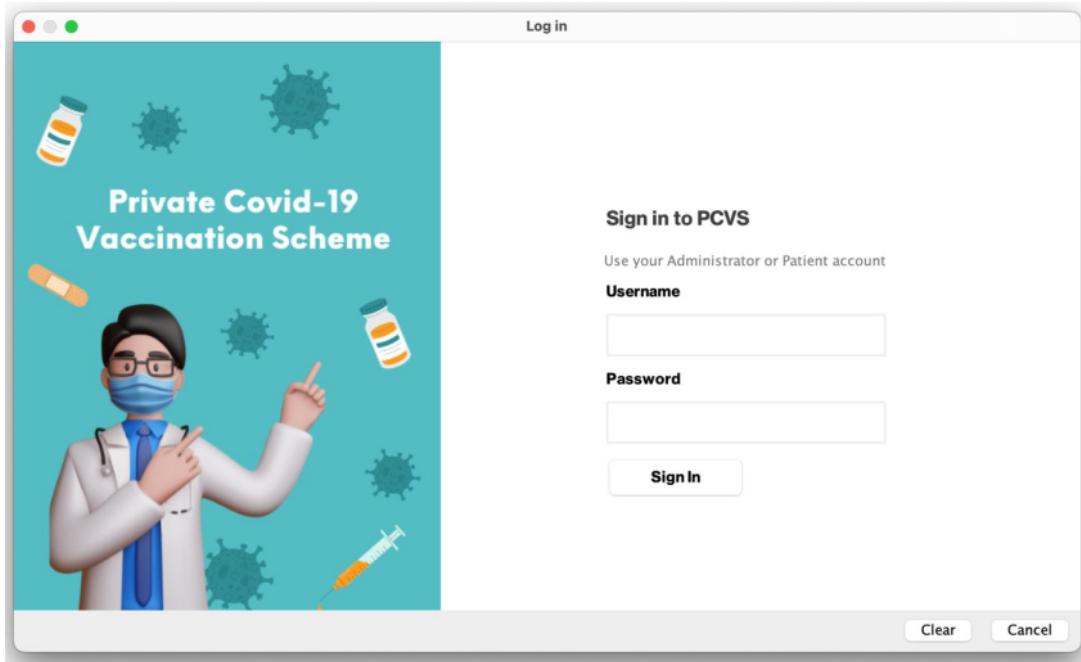
1
Use Case 2 Record New Vaccine Batch

Log in clicked



1
Picture 1: Record New Vaccine Batch

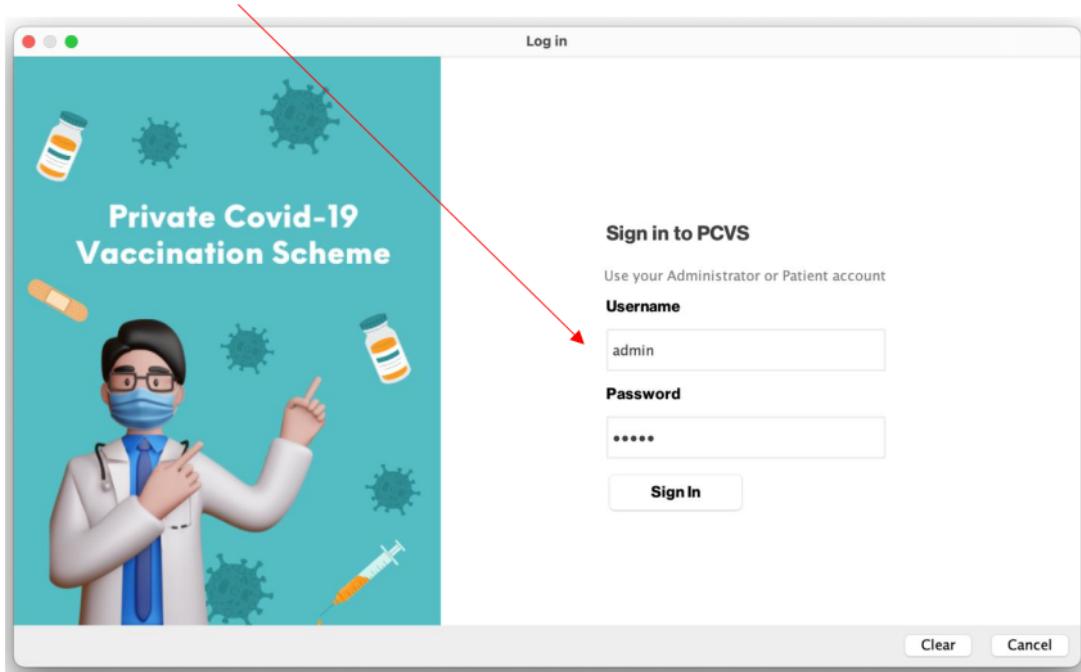
Administrator Login



Picture 2: Record New Vaccine Batch

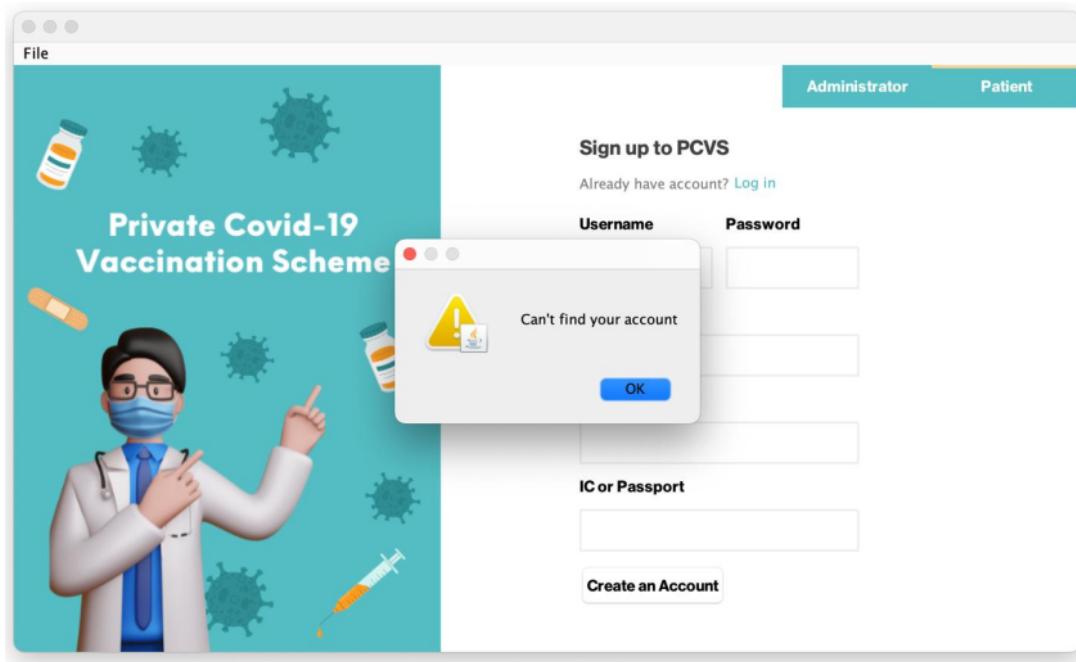
Administrator Login

Administrator log in with a valid account



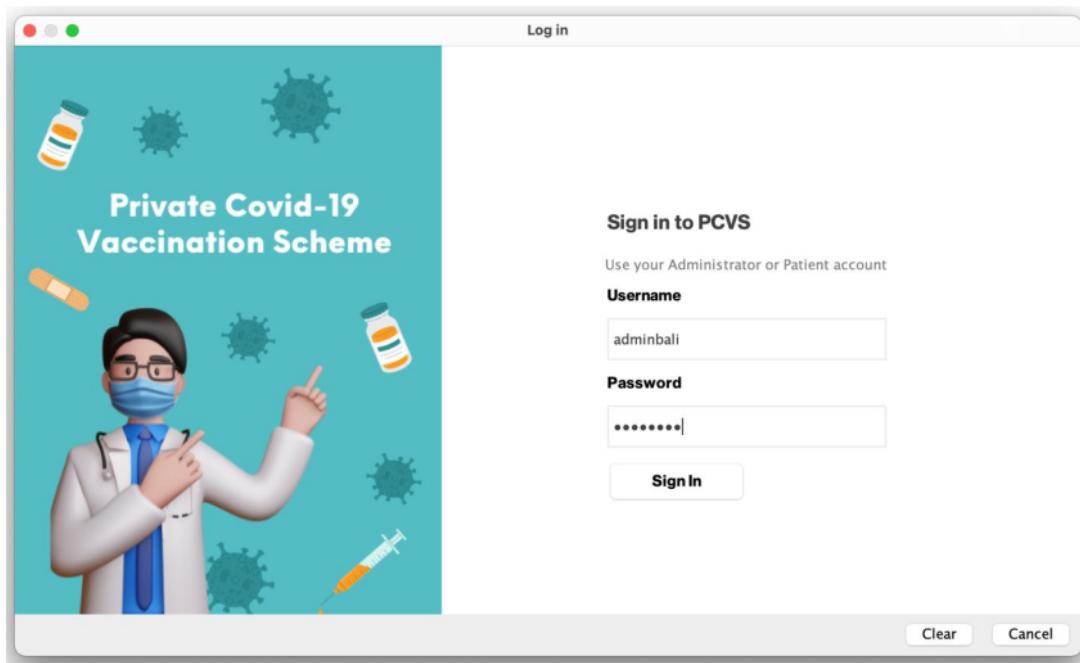
1
Picture 2.1: Record New Vaccine Batch

Administrator Login



Picture 2.2: Record New Vaccine Batch

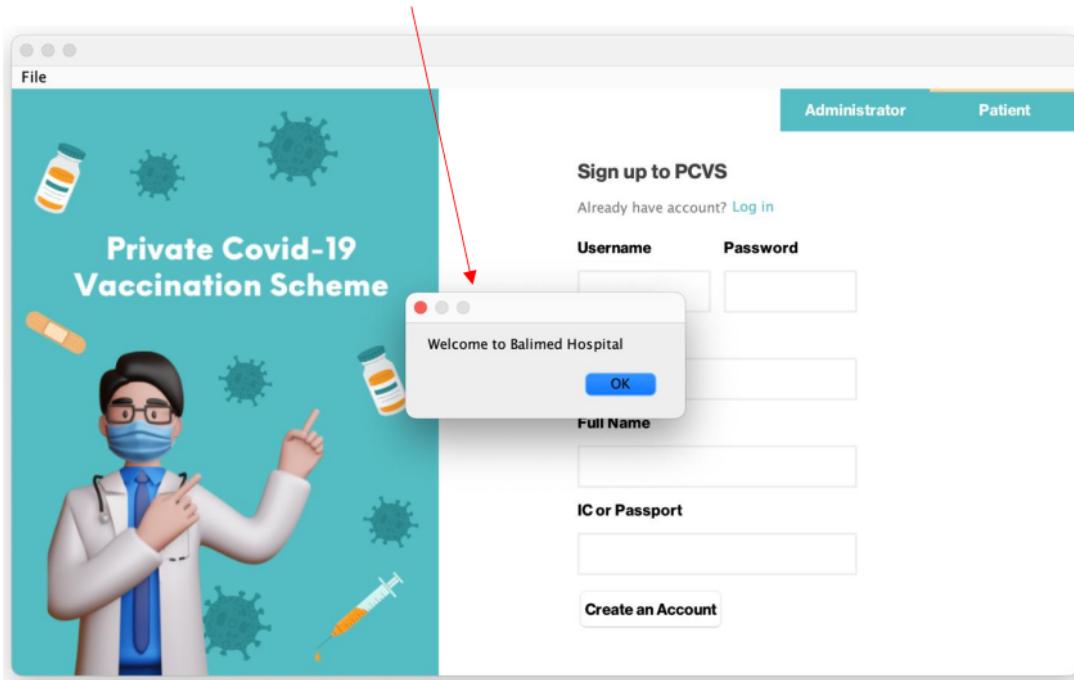
Administrator Login



Picture 3: Record New Vaccine Batch

Administrator Login

Healthcare Centre name displayed



Picture 3.1: Record New Vaccine Batch

Administrator Login

Vaccine information displayed

Select the vaccine ID

Record Vaccine Batch

Select Vaccine ID

JNJ ASZ

Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies

Batch Number

Expiry Date (mm dd yyyy)

Quantity Available

Record

Picture 4: Record New Vaccine Batch

Enter Batch Information

Enter information

The screenshot shows a web-based application interface for managing vaccine batches. On the left, a sidebar has a teal background and contains the following buttons:

- Record Vaccine Batch (highlighted by a red arrow)
- View Vaccine Batch
- Administrator Information
- Patient Information
- Vaccination Information
- Log Out

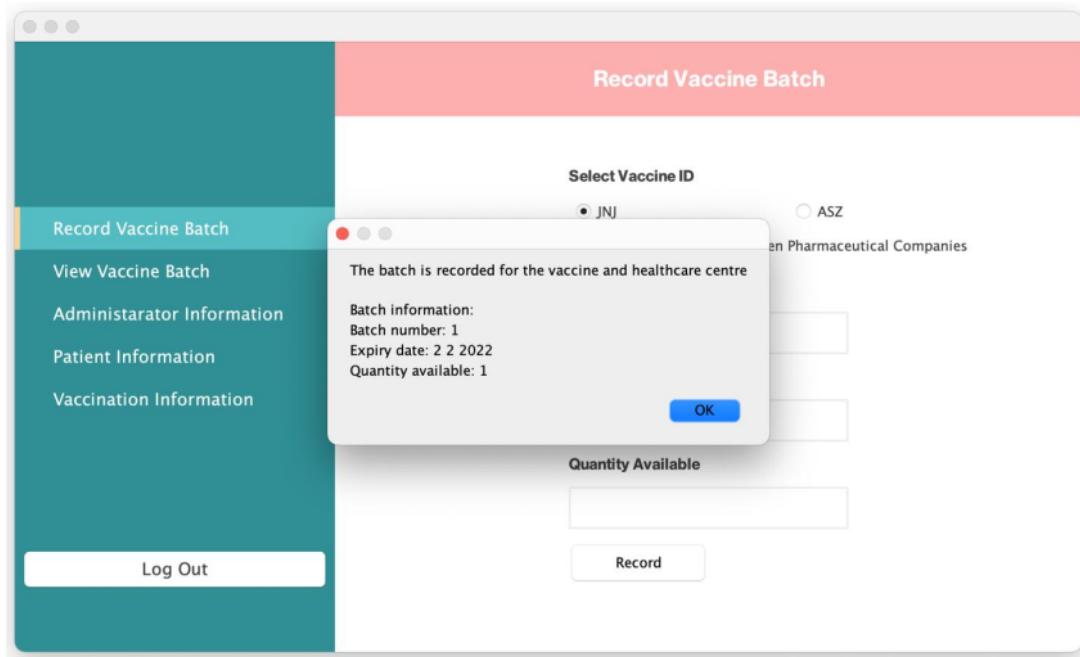
The main content area has a pink header bar with the text "Record Vaccine Batch". Below the header, there is a section titled "Select Vaccine ID" with two radio buttons: "JNJ" (selected) and "ASZ". A note below the radio buttons states: "Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies".

The form fields include:

- Batch Number:** An input field containing the value "1".
- Expiry Date (mm dd yyyy):** An input field containing the value "22 2022".
- Quantity Available:** An input field containing the value "1".
- Record:** A button at the bottom right of the form.

Picture 4.1: Record New Vaccine Batch

Enter Batch Information



Picture 4.2: Record New Vaccine Batch

Enter Batch Information

Another vaccine ID selected

The screenshot shows a web-based application interface. On the left, there is a vertical sidebar with a teal background containing the following menu items:

- Record Vaccine Batch (selected)
- View Vaccine Batch
- Administrator Information
- Patient Information
- Vaccination Information

At the bottom of the sidebar is a white button labeled "Log Out".

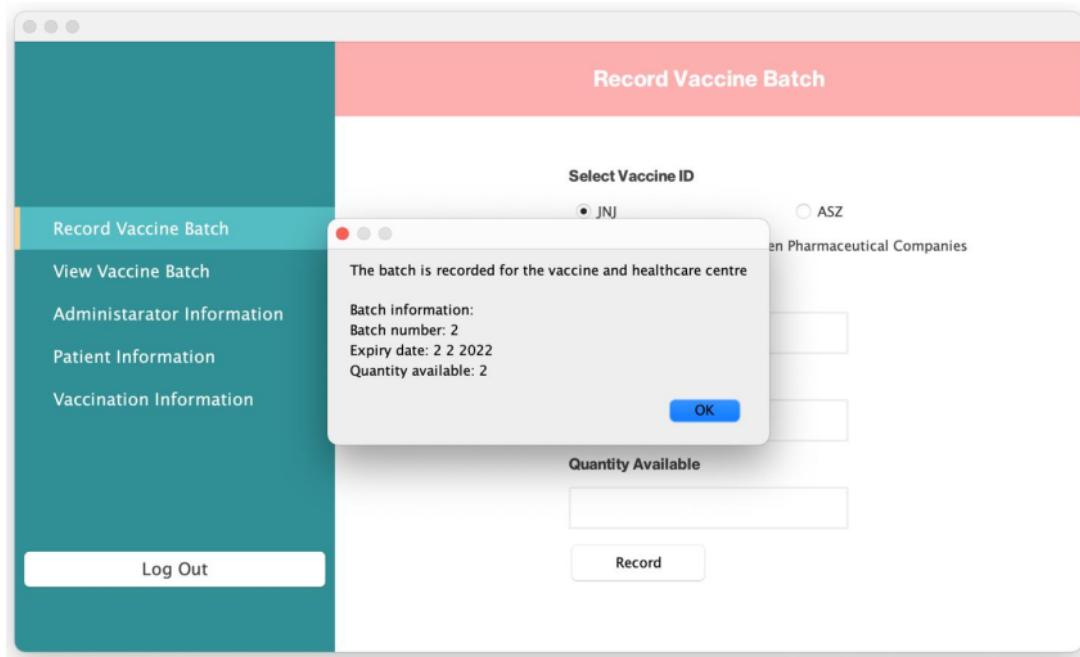
The main content area has a pink header bar with the text "Record Vaccine Batch". Below the header, the text "Select Vaccine ID" is displayed, followed by two radio buttons: "JNJ" (unselected) and "ASZ" (selected). A note below the radio buttons states: "AstraZeneca vaccine, developed by AstraZeneca, University of Oxford".

The form fields include:

- Batch Number:** An input field containing the value "2".
- Expiry Date (mm dd yyyy):** An input field containing the value "2 2 2022".
- Quantity Available:** An input field containing the value "2".
- Record:** A white button with black text.

Picture 5: Record New Vaccine Batch

Enter Batch Information



Picture 5.1: Record New Vaccine Batch

Enter Batch Information

The screenshot shows a software application window titled "Record Vaccine Batch". On the left is a sidebar with a teal background containing the following menu items:

- Record Vaccine Batch
- View Vaccine Batch
- Administrator Information
- Patient Information
- Vaccination Information

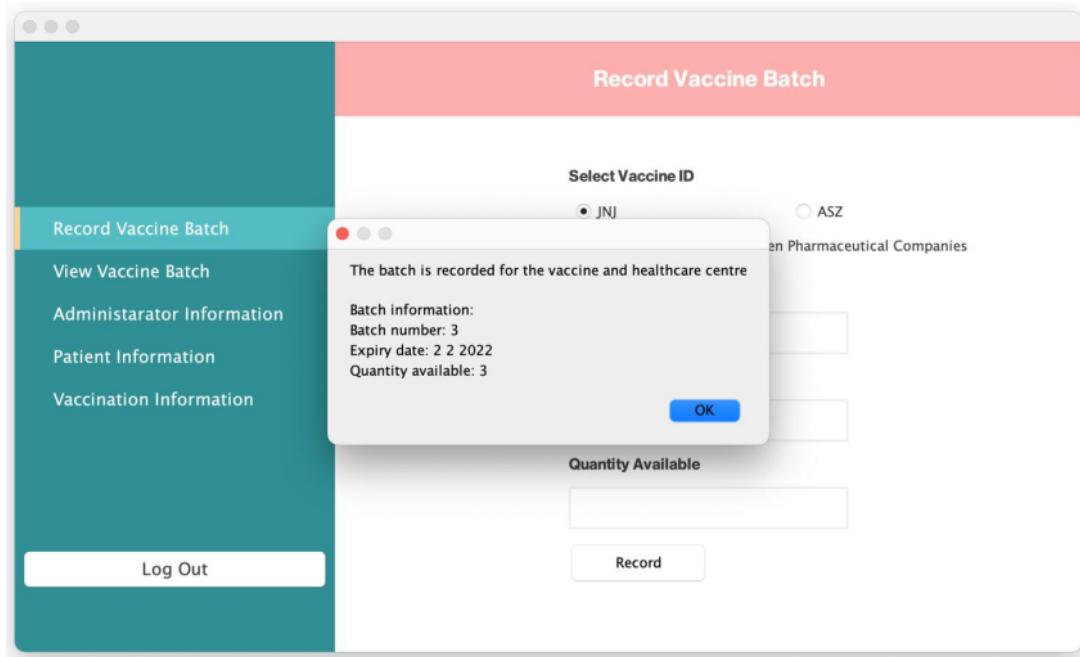
At the bottom of the sidebar is a "Log Out" button.

The main content area has a white background with a pink header bar containing the title "Record Vaccine Batch". Below the header are several input fields:

- Select Vaccine ID**:
Two radio buttons are shown: JNJ and ASZ. A small note below says "Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies".
- Batch Number**: An input field containing the value "3".
- Expiry Date (mm dd yyyy)**: An input field containing the value "2 2 2022".
- Quantity Available**: An input field containing the value "3".
- Record**: A button located at the bottom right of the form.

Picture 6: Record New Vaccine Batch

Enter Batch Information



Picture 6.1: Record New Vaccine Batch

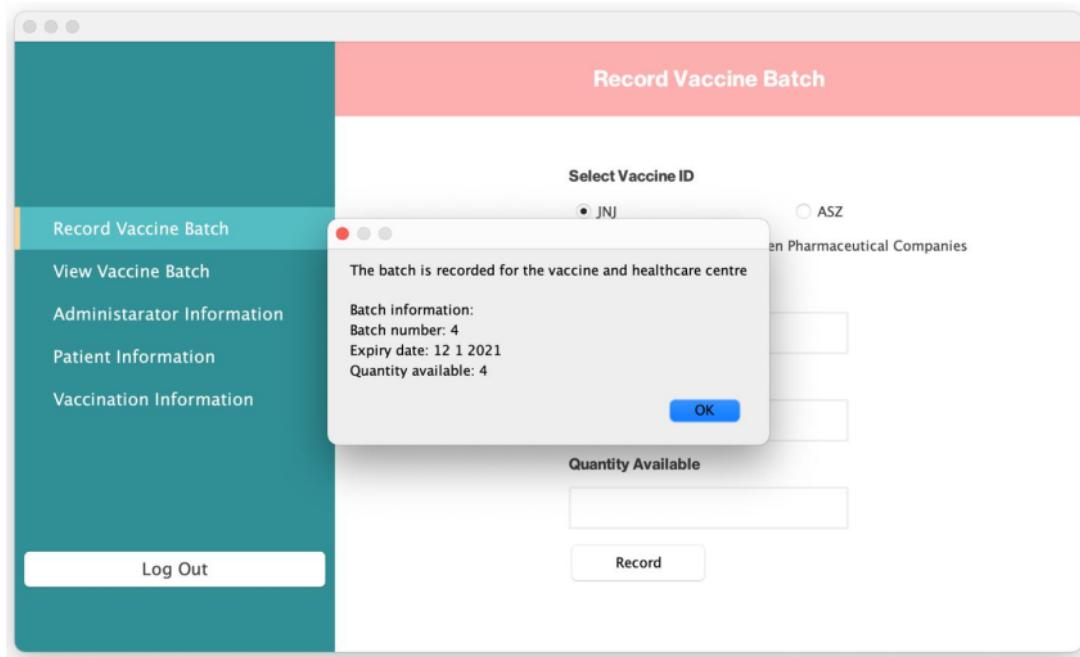
Enter Batch Information

Expired batch

The screenshot shows a user interface for managing vaccine batches. On the left, a sidebar has a teal header with the title 'Record Vaccine Batch'. Below this are several menu items: 'View Vaccine Batch', 'Administrator Information', 'Patient Information', and 'Vaccination Information'. At the bottom of the sidebar is a white button labeled 'Log Out'. The main content area has a pink header bar with the title 'Record Vaccine Batch'. Below this, there's a section titled 'Select Vaccine ID' with two radio buttons: 'JNJ' (unselected) and 'ASZ' (selected). A note below the radio buttons states: 'AstraZeneca vaccine, developed by AstraZeneca, University of Oxford'. The form includes fields for 'Batch Number' (containing '4'), 'Expiry Date (mm dd yyyy)' (containing '12 1 2021'), and 'Quantity Available' (containing '4'). A 'Record' button is located at the bottom right of the form.

Picture 7: Record New Vaccine Batch

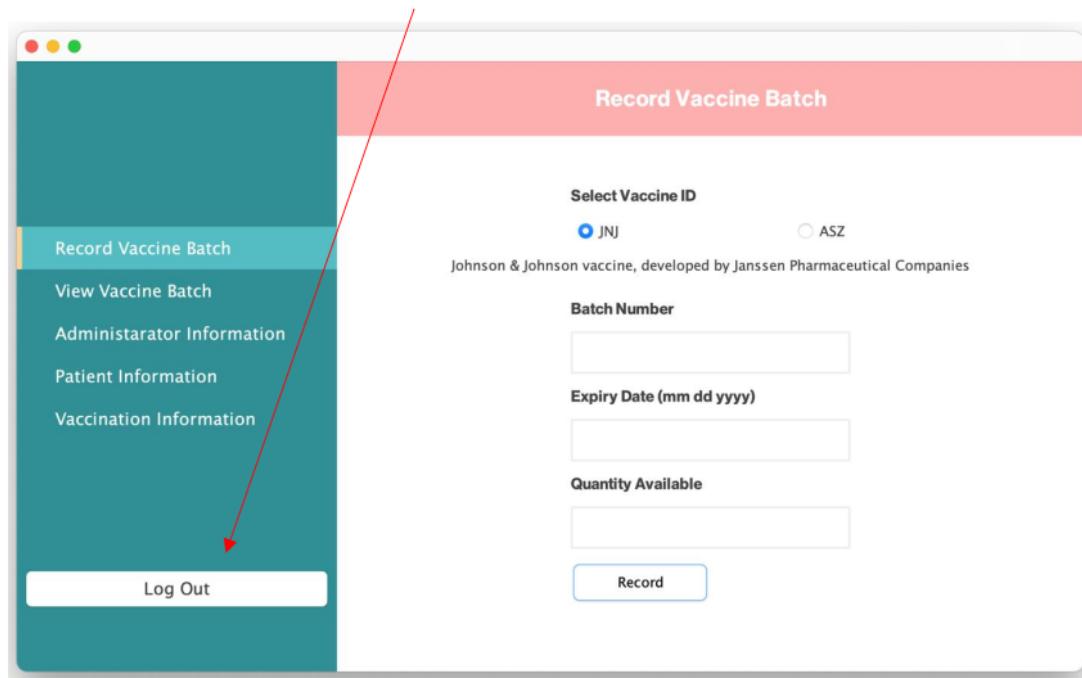
Enter Batch Information



Picture 7.1: Record New Vaccine Batch

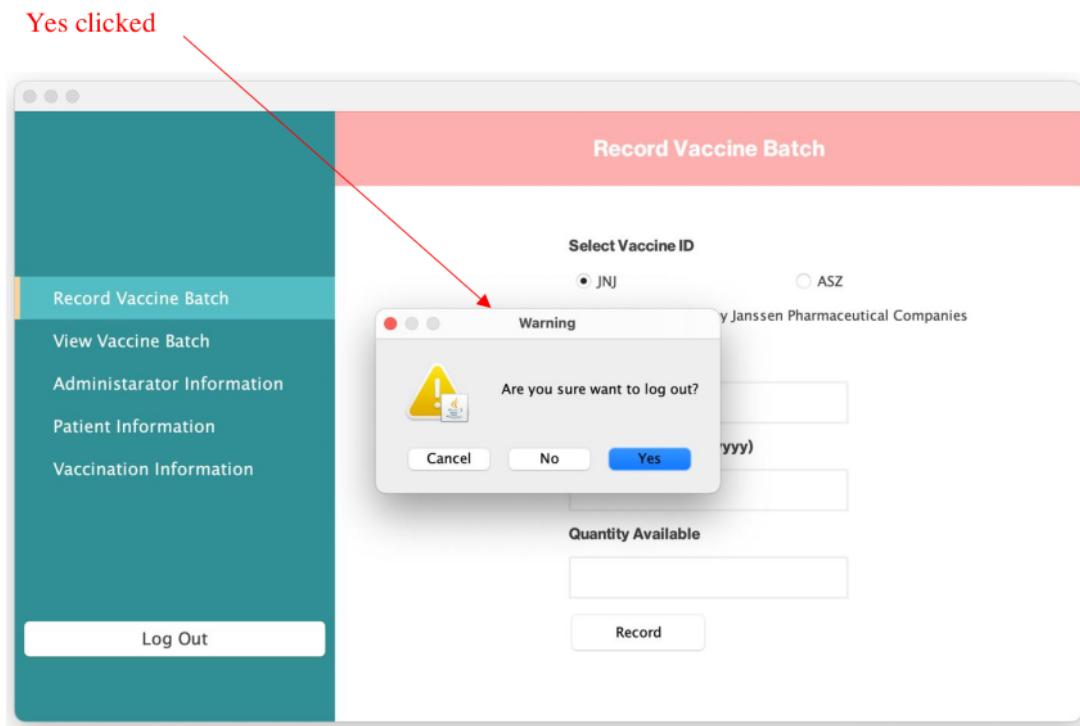
Enter Batch Information

Log out clicked



Picture 8: Record New Vaccine Batch

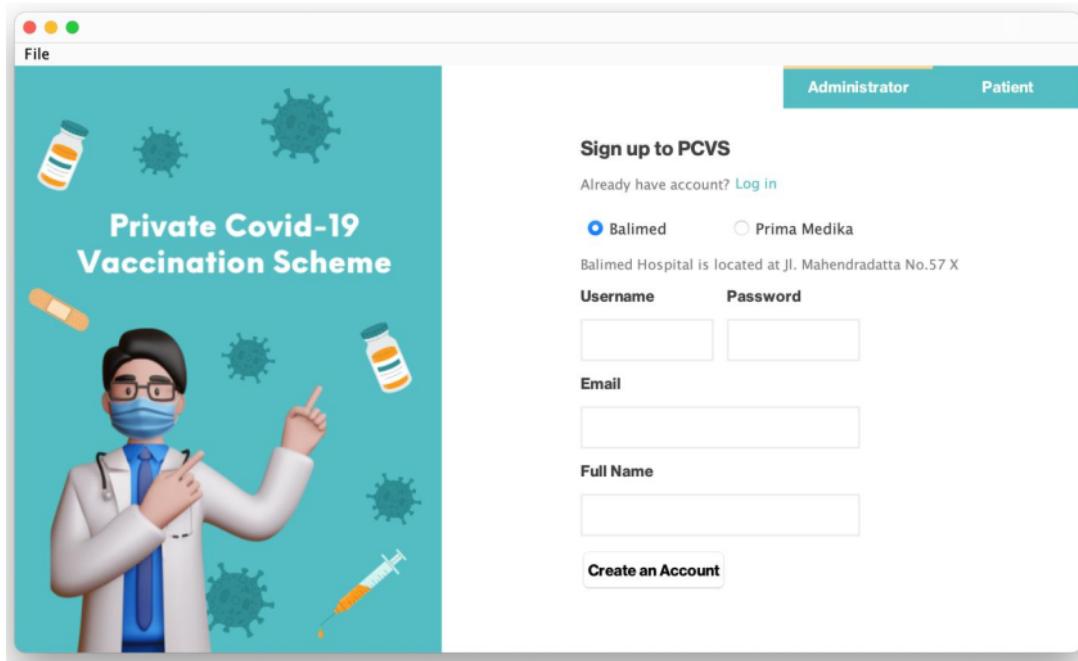
Log Out



Picture 8.1: Record New Vaccine Batch

Log Out

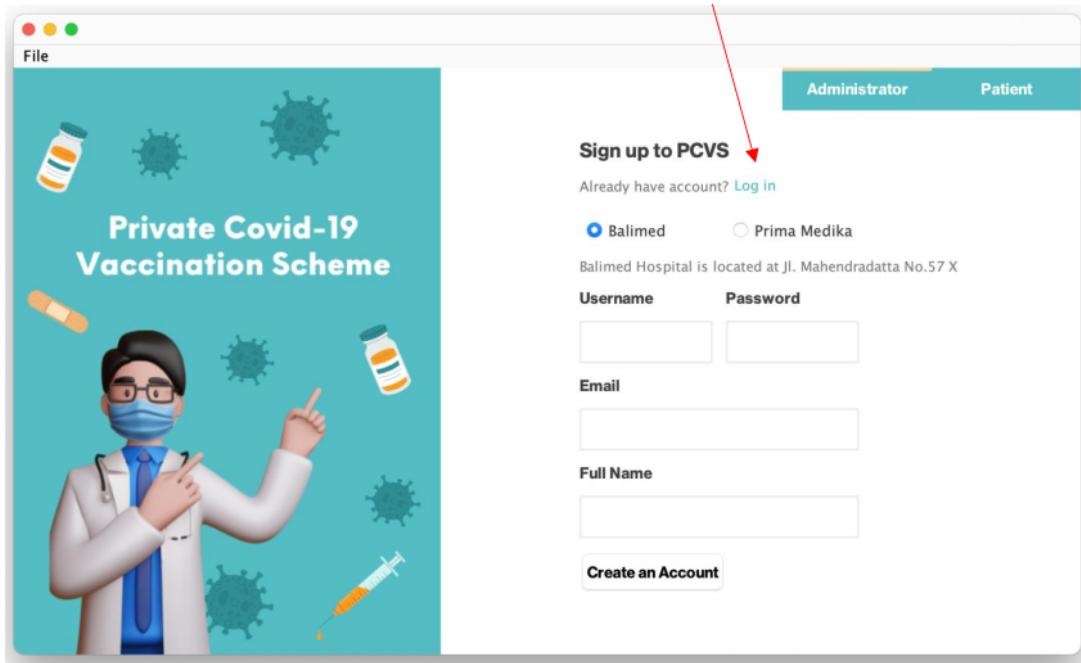
[Back to SignUpGUI](#)



Picture 8.2: Record New Vaccine Batch

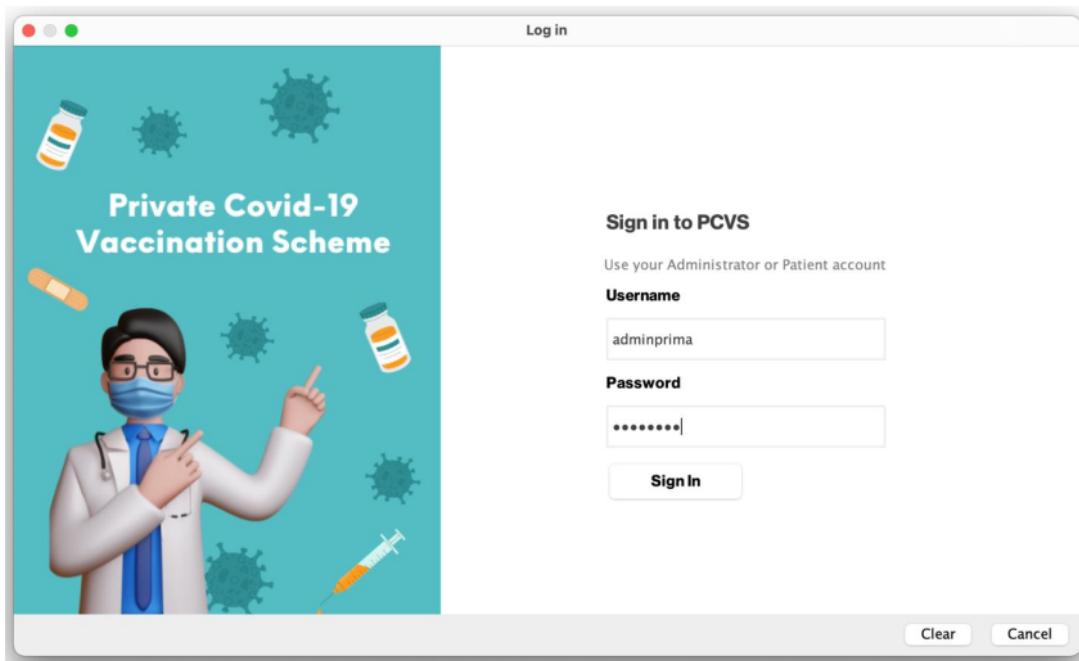
[Log Out](#)

Log in clicked



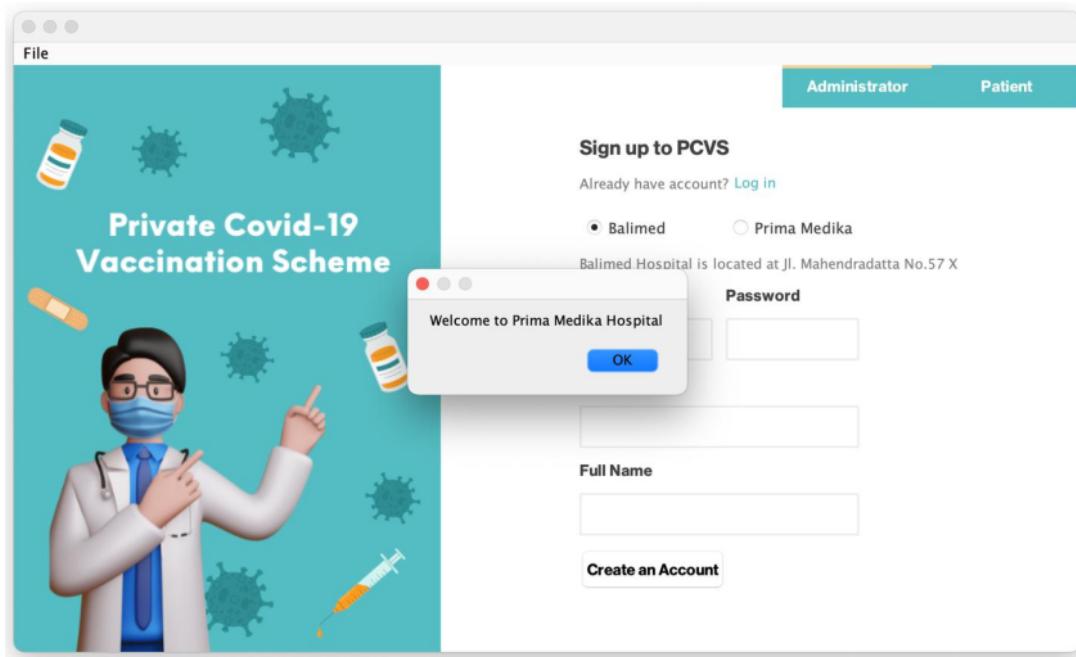
1
Picture 9: Record New Vaccine Batch

[Administrator Login](#)



Picture 9.1: Record New Vaccine Batch

Administrator Login



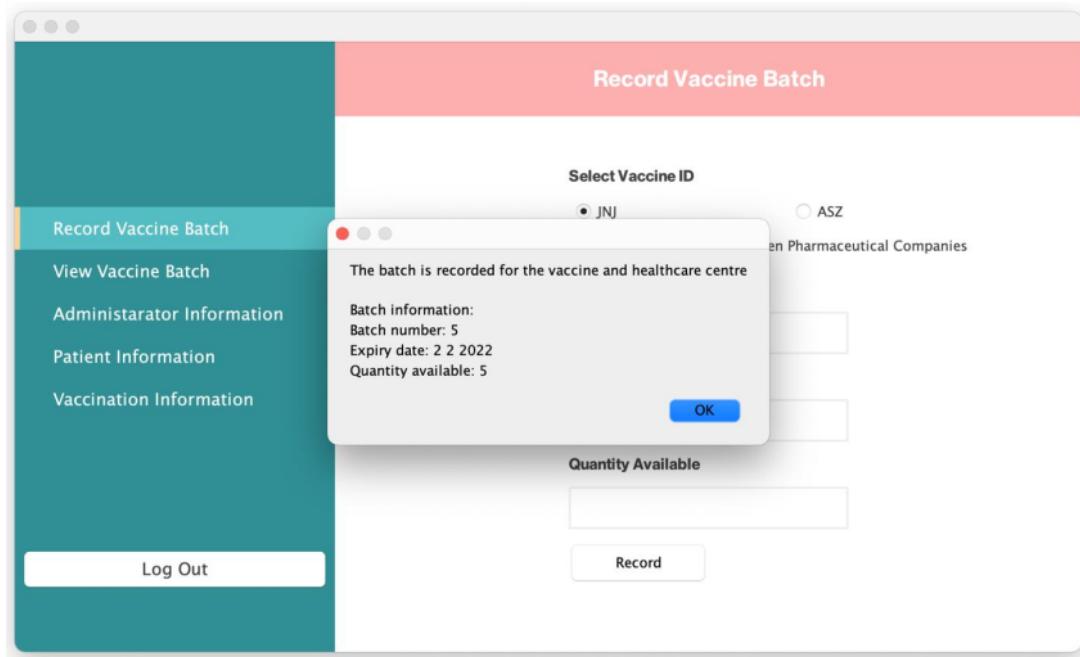
1
Picture 9.2: Record New Vaccine Batch

Administrator Login

The screenshot shows a web-based application interface for managing vaccine batches. The left sidebar has a teal background with white text, listing navigation options: Record Vaccine Batch (which is selected and highlighted in orange), View Vaccine Batch, Administrator Information, Patient Information, and Vaccination Information. At the bottom of the sidebar is a 'Log Out' button. The main content area has a light gray background. At the top right, a red header bar contains the title 'Record Vaccine Batch'. Below this, the text 'Select Vaccine ID' is displayed, followed by two radio buttons: 'JNJ' (selected) and 'ASZ'. A note below the radio buttons states 'Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies'. The next section is labeled 'Batch Number' with an input field containing the value '5'. Below this is a section for 'Expiry Date (mm dd yyyy)' with an input field containing '2 2 2022'. The final section is labeled 'Quantity Available' with an input field containing the value '5'. At the bottom right of the form is a 'Record' button.

Picture 9.3: Record New Vaccine Batch

Enter Batch Information in Different Administrator Account

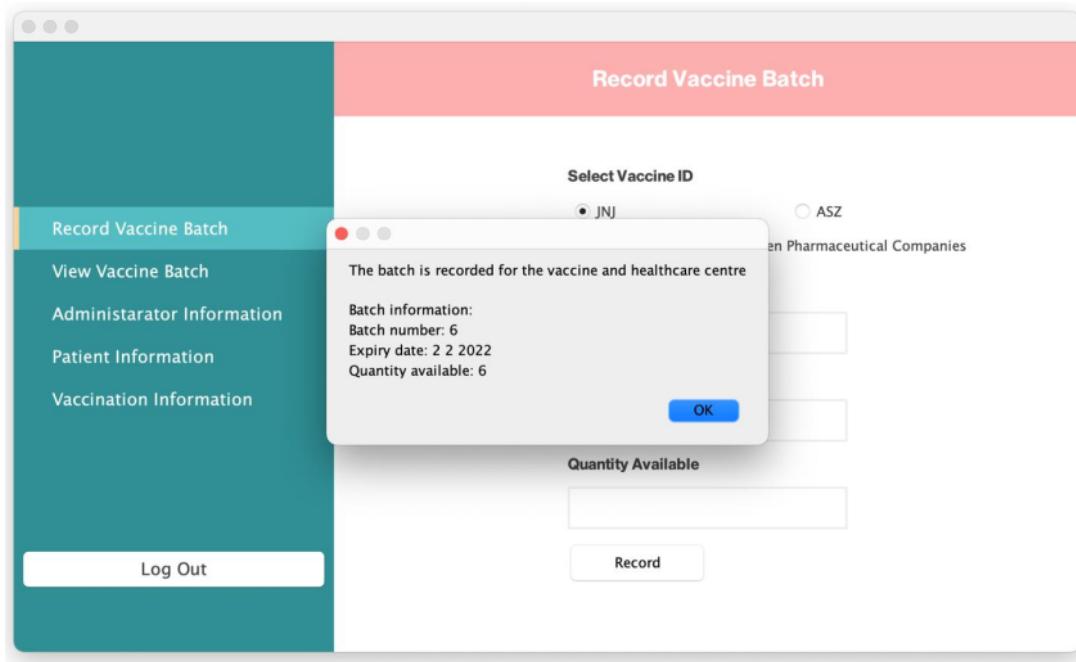


Picture 9.4: Record New Vaccine Batch

Enter Batch Information in Different Administrator Account

The screenshot shows a computer application window titled "Record Vaccine Batch". The window has a teal sidebar on the left containing links: "Record Vaccine Batch" (which is active), "View Vaccine Batch", "Administrator Information", "Patient Information", and "Vaccination Information". At the bottom of the sidebar is a "Log Out" button. The main content area has a pink header bar with the title. Below it, there's a section for "Select Vaccine ID" with radio buttons for "JNJ" and "ASZ" (which is selected). A note below says "AstraZeneca vaccine, developed by AstraZeneca, University of Oxford". There are four input fields: "Batch Number" (containing "6"), "Expiry Date (mm dd yyyy)" (containing "2 2 2022"), "Quantity Available" (containing "6"), and a "Record" button at the bottom right.

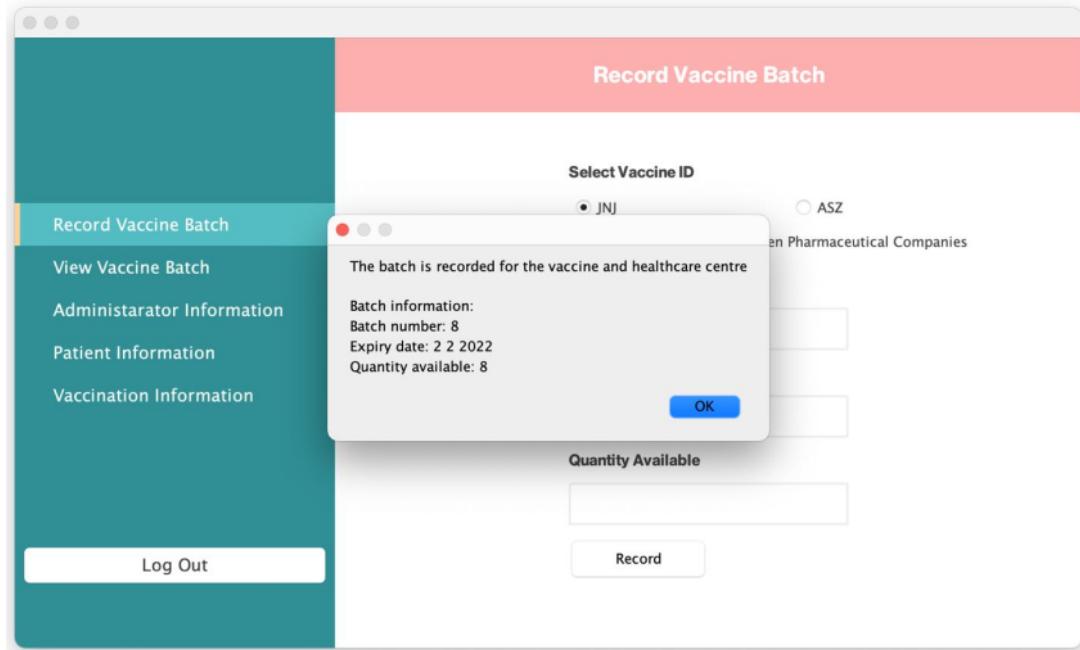
Picture 10: Record New Vaccine Batch



Picture 10.1: Record New Vaccine Batch

The screenshot shows a software application window titled "Record Vaccine Batch". On the left is a vertical navigation bar with the following options: Record Vaccine Batch (selected), View Vaccine Batch, Administrator Information, Patient Information, and Vaccination Information. At the bottom of the navigation bar is a "Log Out" button. The main content area has a pink header bar with the title "Record Vaccine Batch". Below the header, there is a section titled "Select Vaccine ID" with two radio buttons: JNJ (unselected) and ASZ (selected). A note below the radio buttons states: "AstraZeneca vaccine, developed by AstraZeneca, University of Oxford". There are three input fields: "Batch Number" containing "8", "Expiry Date (mm dd yyyy)" containing "2 2 2022", and "Quantity Available" containing "8". At the bottom right is a "Record" button.

Picture 11: Record New Vaccine Batch



Picture 11.1: Record New Vaccine Batch

The screenshot shows a computer application window titled "Record Vaccine Batch". The window has a teal sidebar on the left containing links: "Record Vaccine Batch" (selected), "View Vaccine Batch", "Administrator Information", "Patient Information", and "Vaccination Information". At the bottom of the sidebar is a "Log Out" button. The main content area has a pink header bar with the title. Below it, there's a section for selecting a vaccine ID with radio buttons for "JNJ" (selected) and "ASZ", followed by a note about the Johnson & Johnson vaccine. There are three input fields for "Batch Number", "Expiry Date (mm dd yyyy)", and "Quantity Available". A "Record" button is located at the bottom right of these fields.

Record Vaccine Batch

Select Vaccine ID

JNJ ASZ
Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies

Batch Number

Expiry Date (mm dd yyyy)

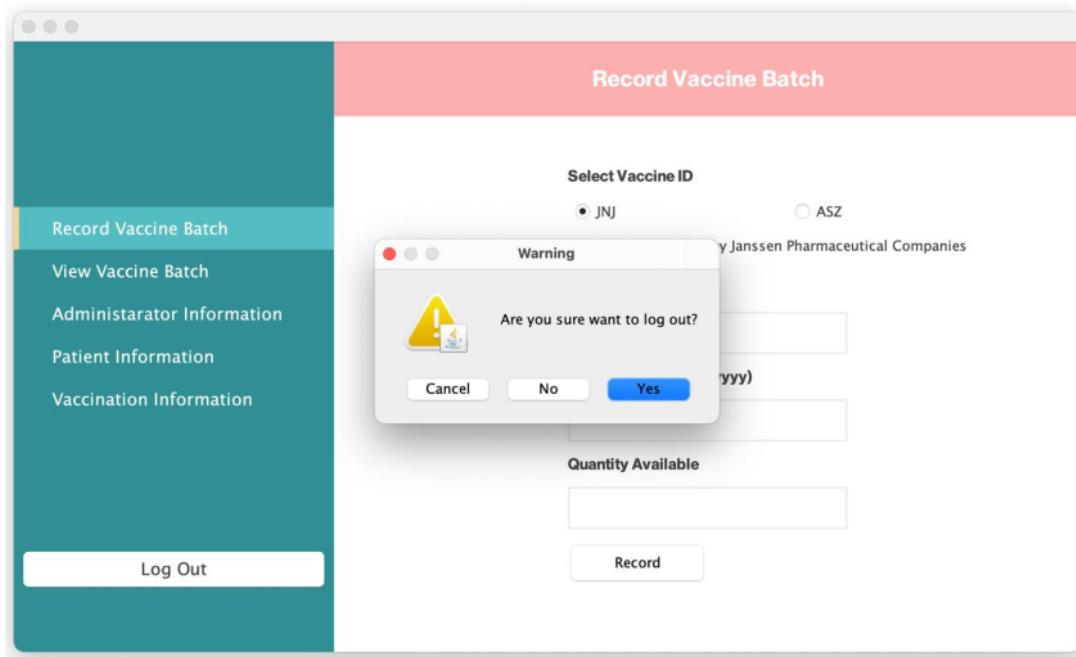
Quantity Available

Record

Log Out

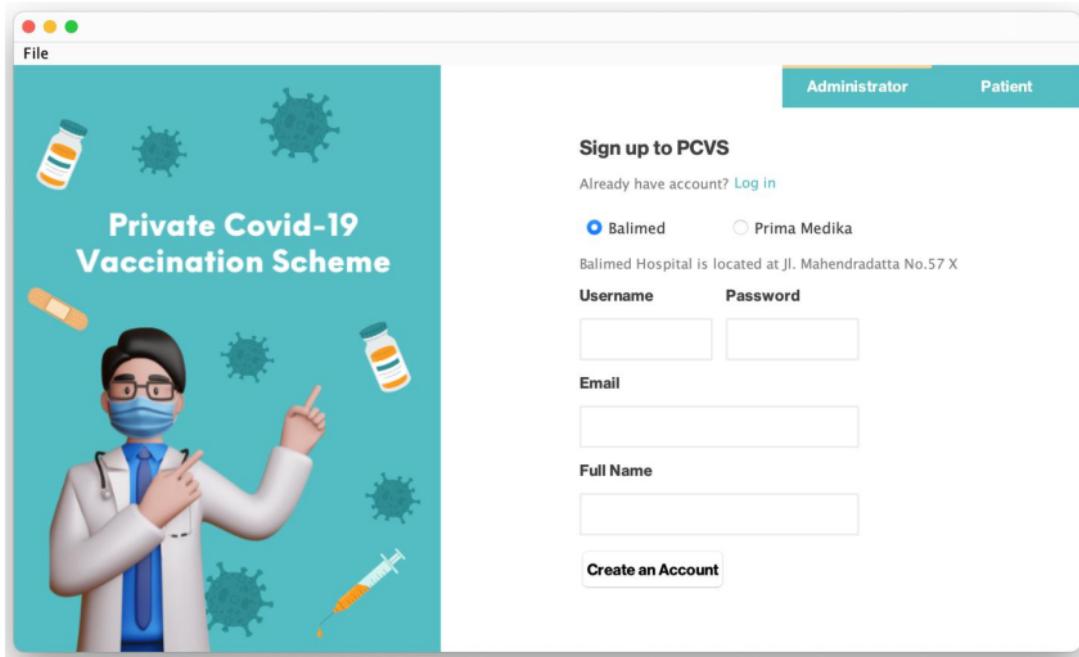
Picture 12: Record New Vaccine Batch

Log Out



Picture 12.1: Record New Vaccine Batch

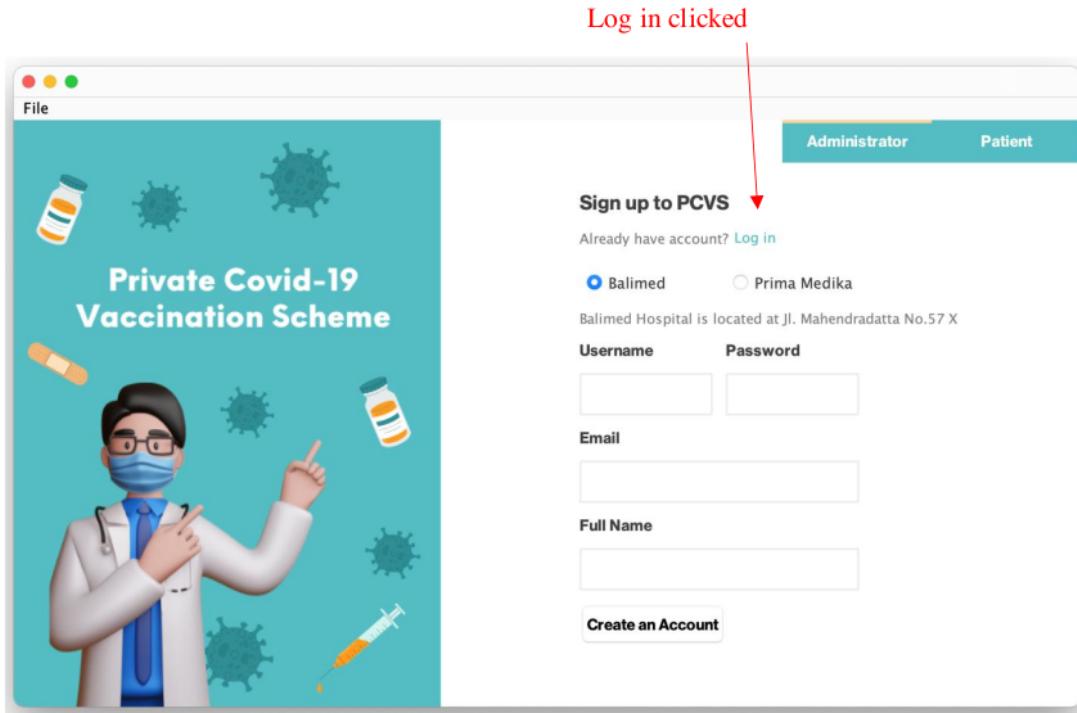
Log Out



Picture 12.2: Record New Vaccine Batch

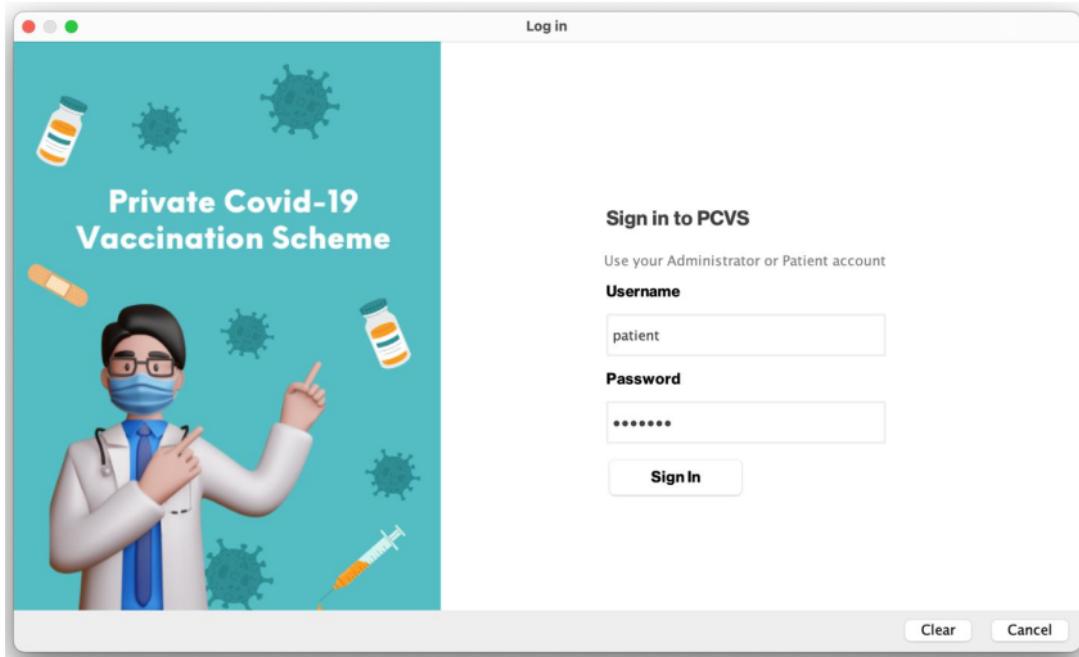
[Log Out](#)

Use Case 3 Request Vaccination Appointment



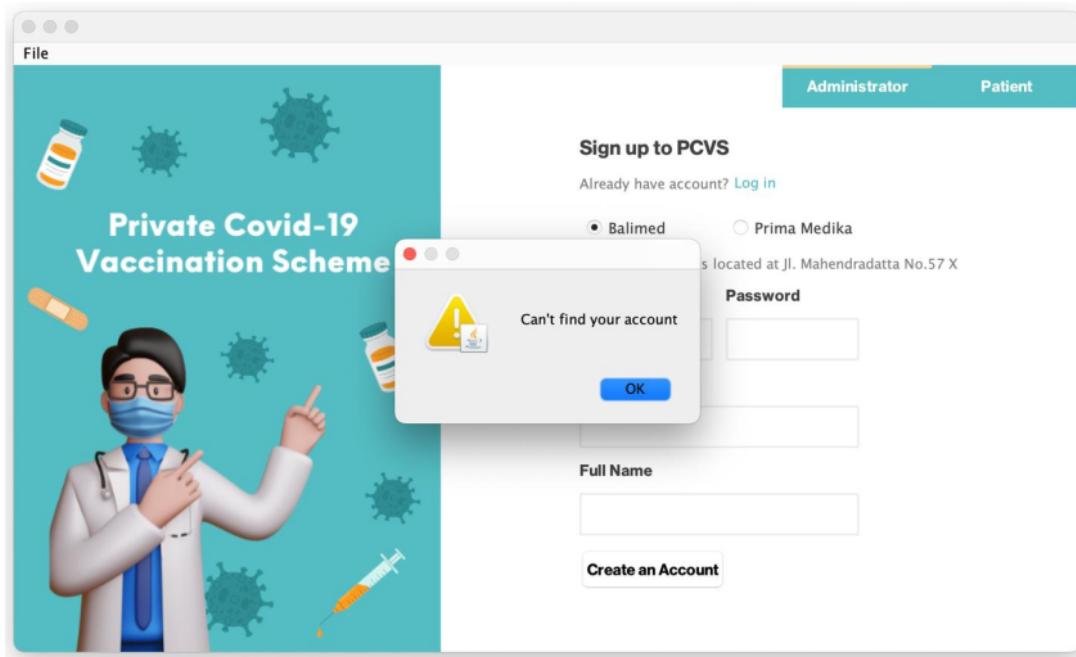
Picture 1: Request Vaccination Appointment

Log in



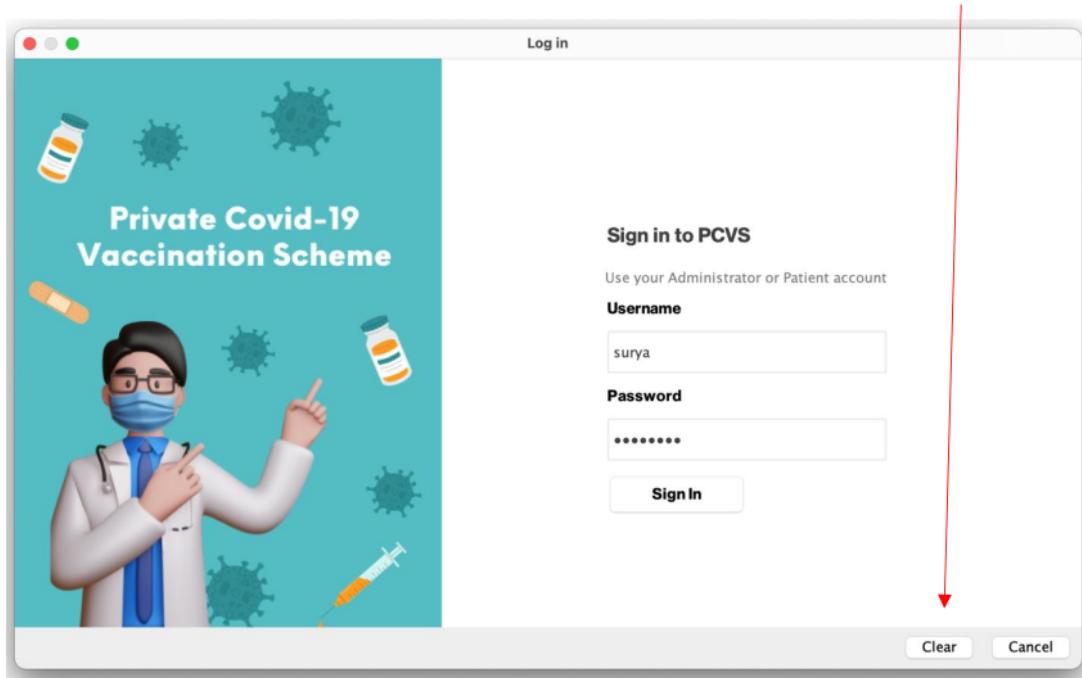
Picture 1.1: Request Vaccination Appointment

Log in Validation



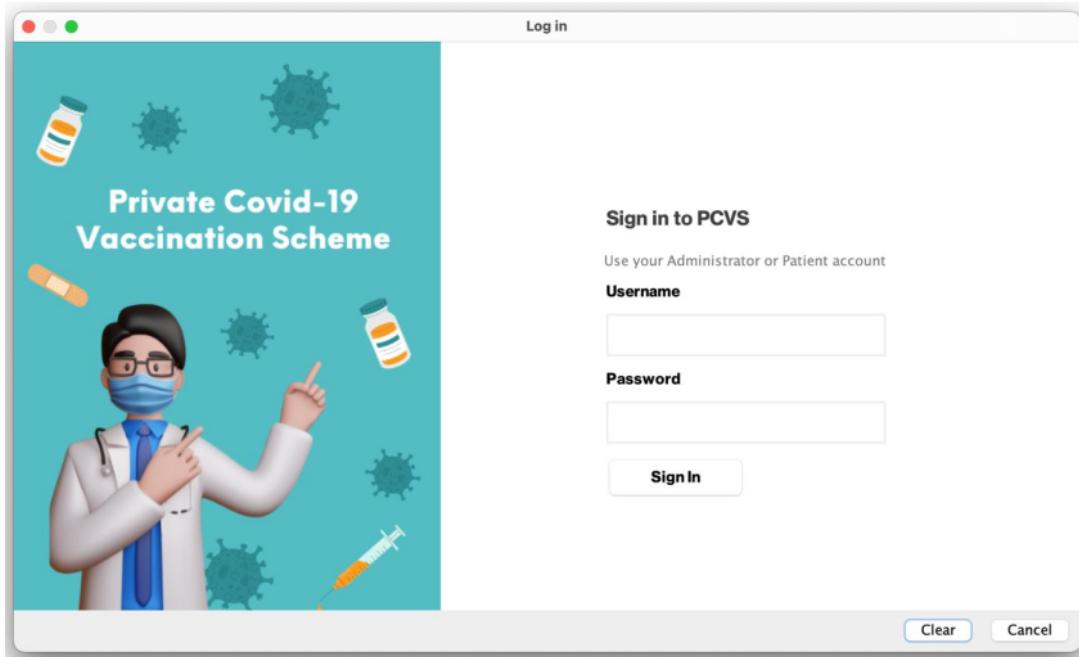
Picture 1.2: Request Vaccination Appointment

Log in Validation



Picture 1.3: Request Vaccination Appointment

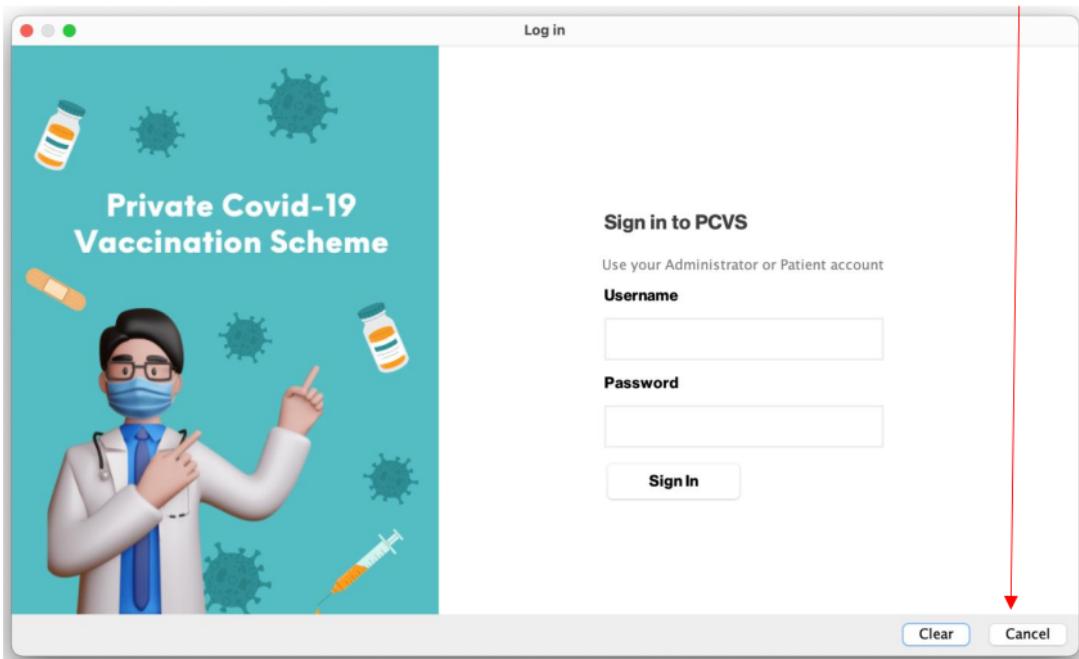
Log in Validation



Picture 1.4: Request Vaccination Appointment

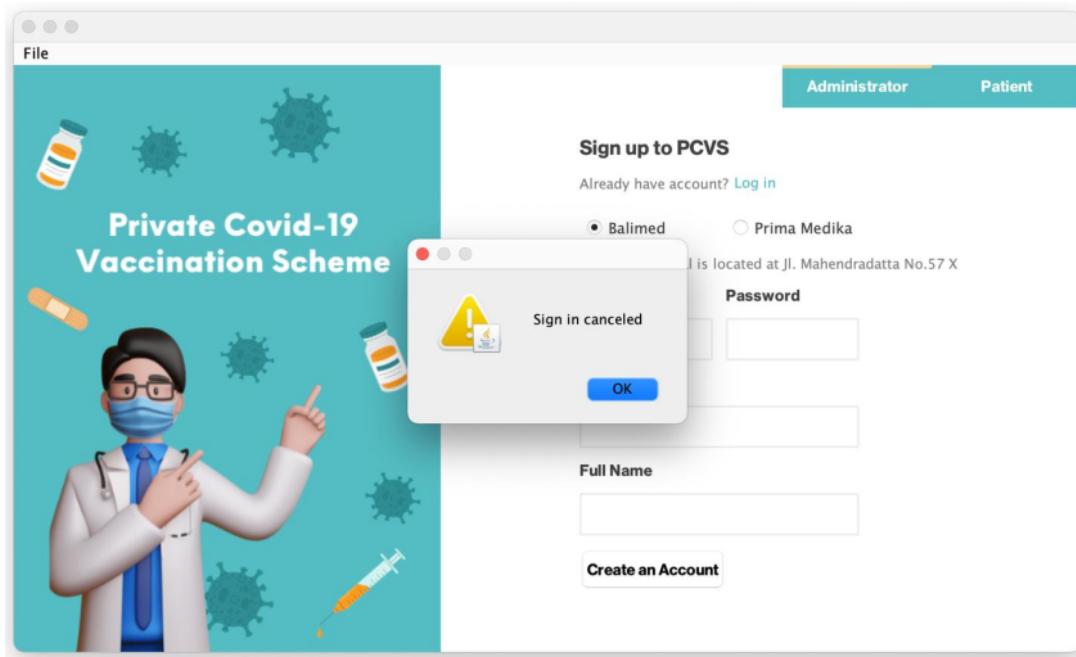
Log in Validation

Cancel clicked



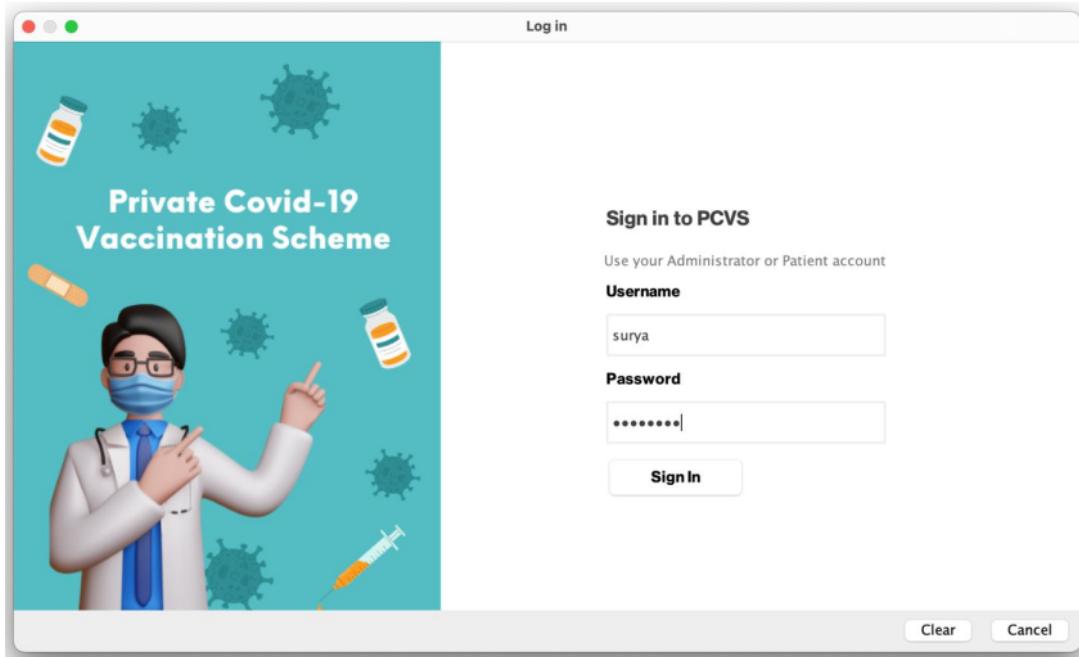
Picture 1.5: Request Vaccination Appointment

Log in Validation



Picture 1.6: Request Vaccination Appointment

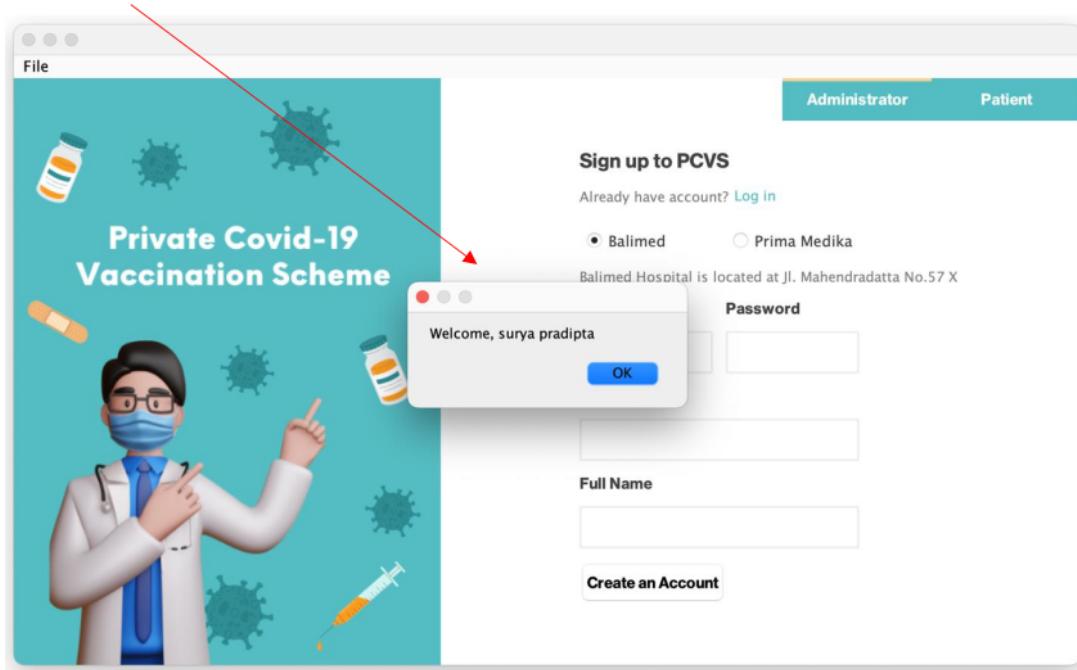
Log in Validation



Picture 2: Request Vaccination Appointment

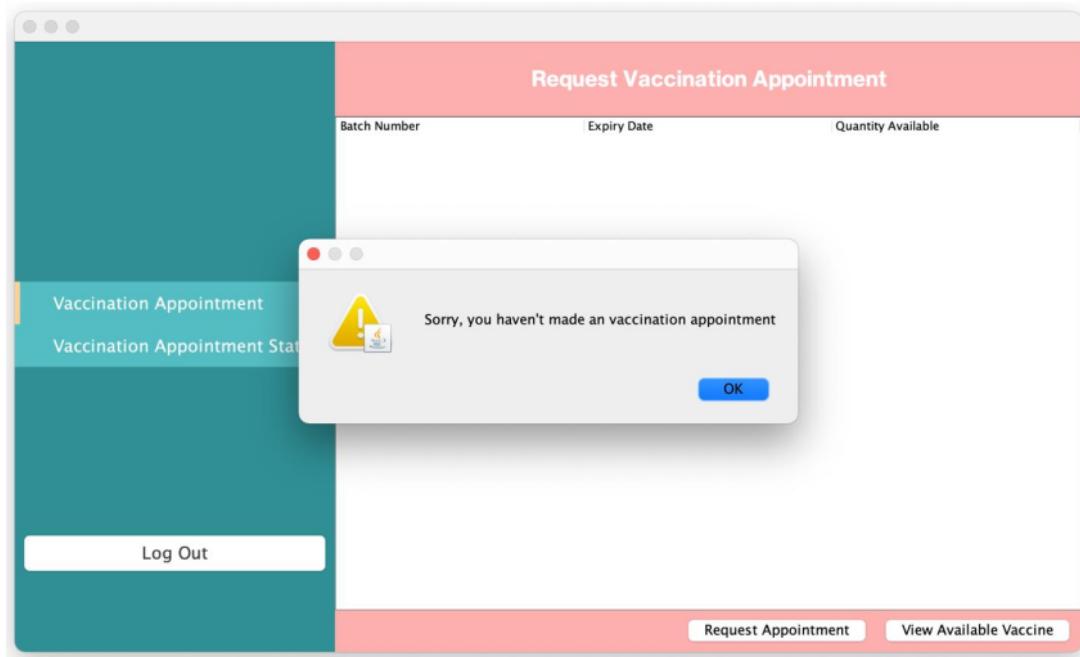
Log in

Patient's full name displayed



Picture 2.1: Request Vaccination Appointment

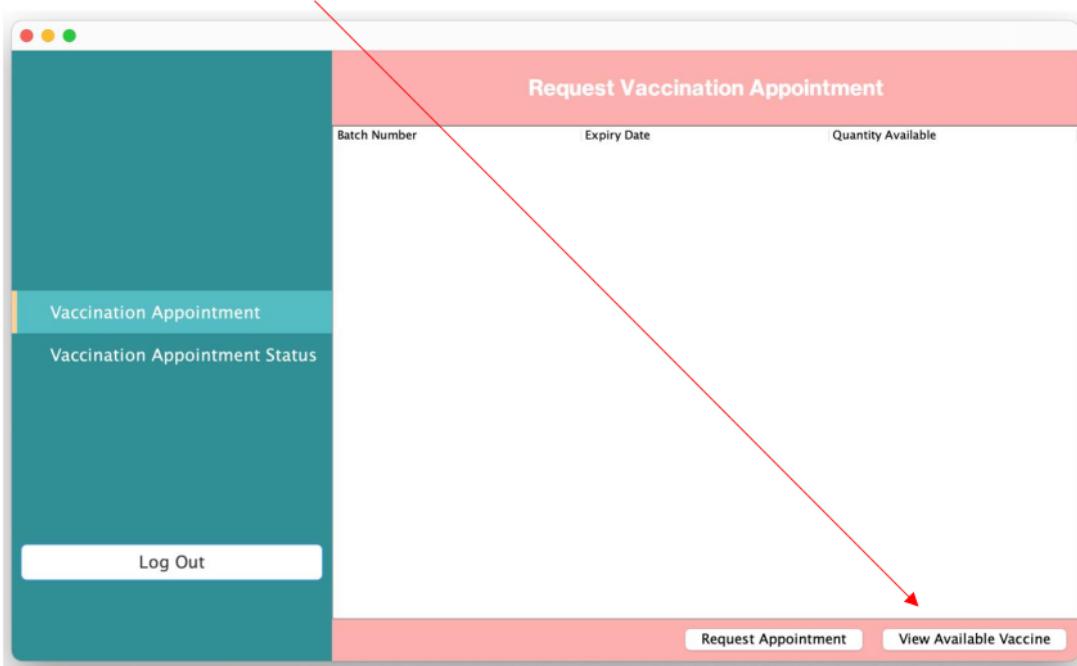
Log in



Picture 2.1: Request Vaccination Appointment

View Vaccination Appointment Status

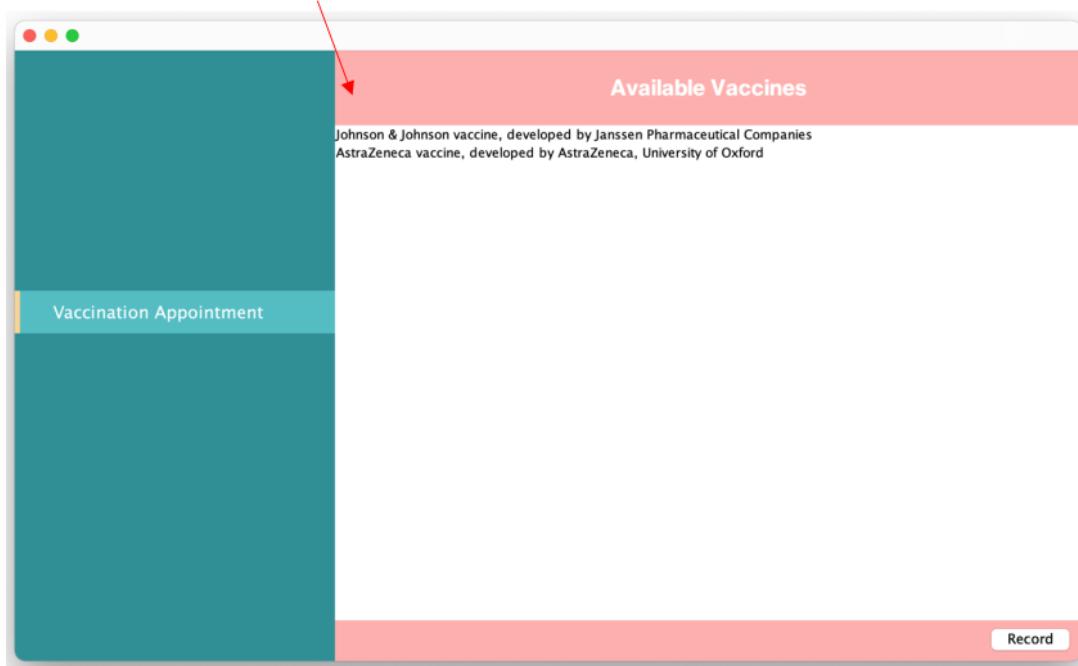
Patient select to view available vaccine



Picture 3: Request Vaccination Appointment

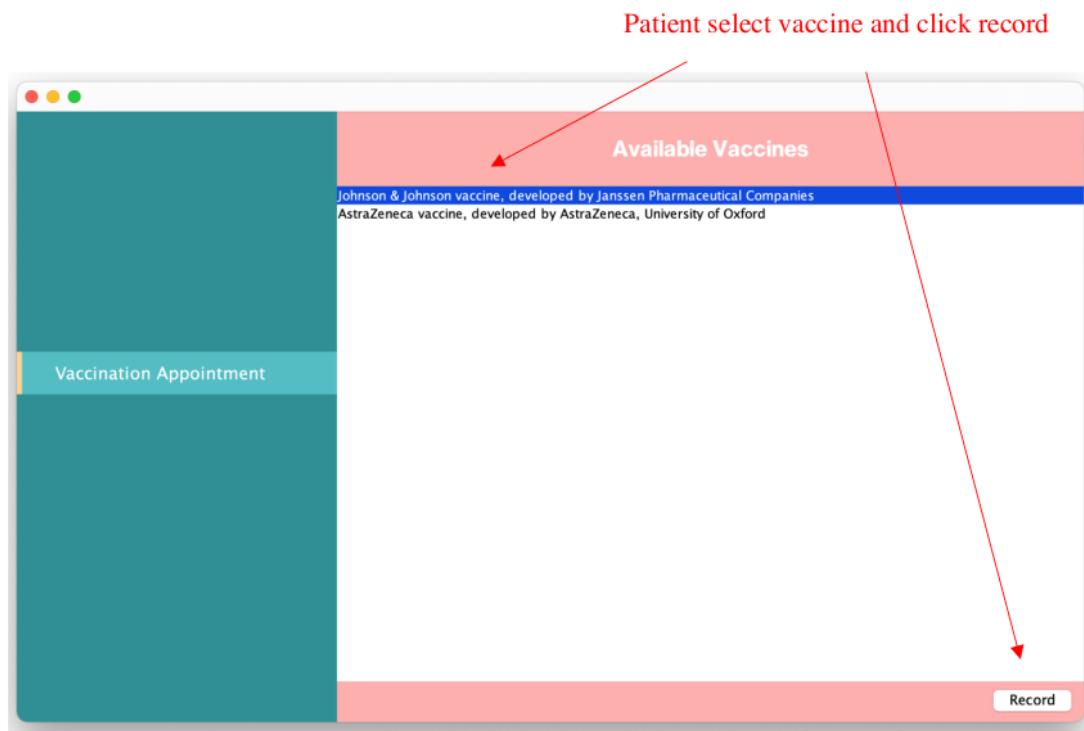
Patient Menu

List vaccine information displayed



Picture 4: Request Vaccination Appointment

[View Available Vaccine](#)

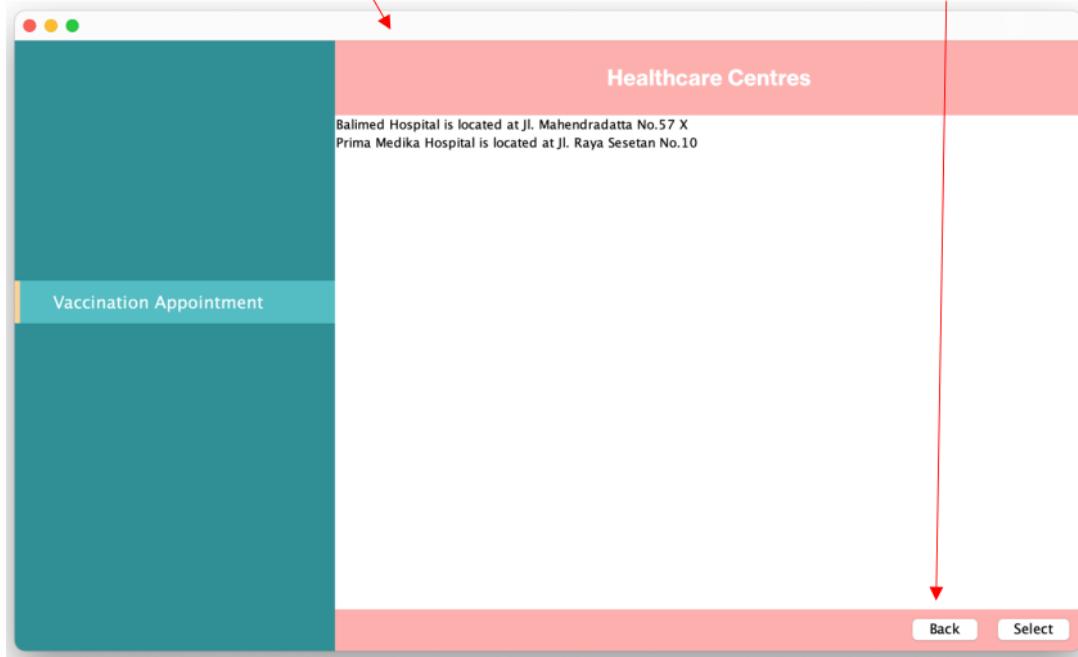


Picture 4.1: Request Vaccination Appointment

View Available Vaccine

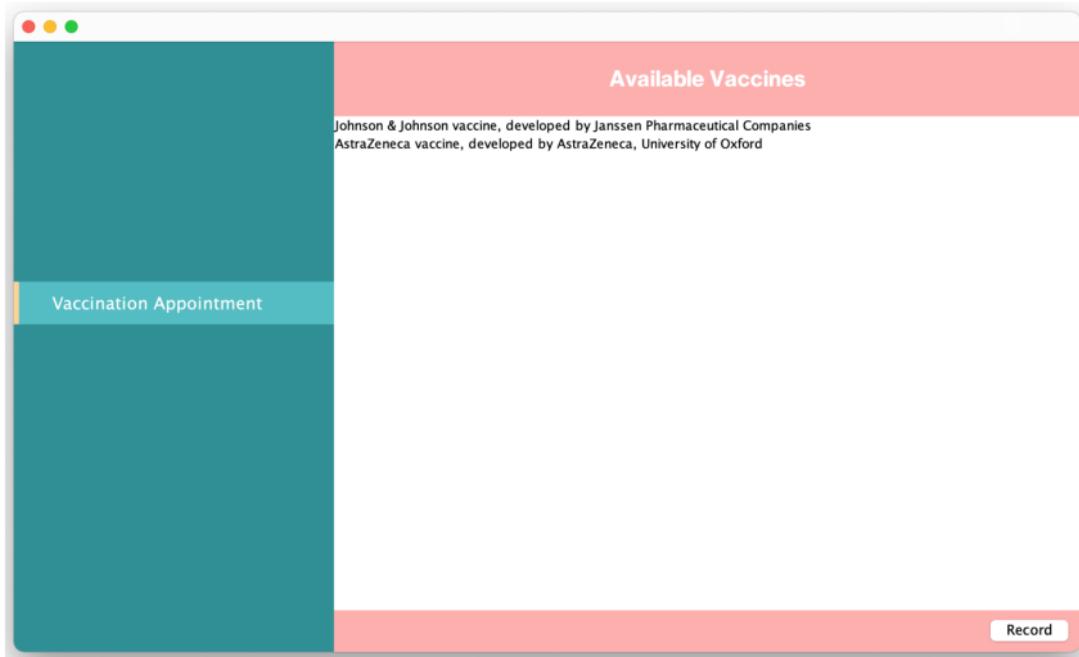
List healthcare centres offering vaccine displayed

Patient clicks back



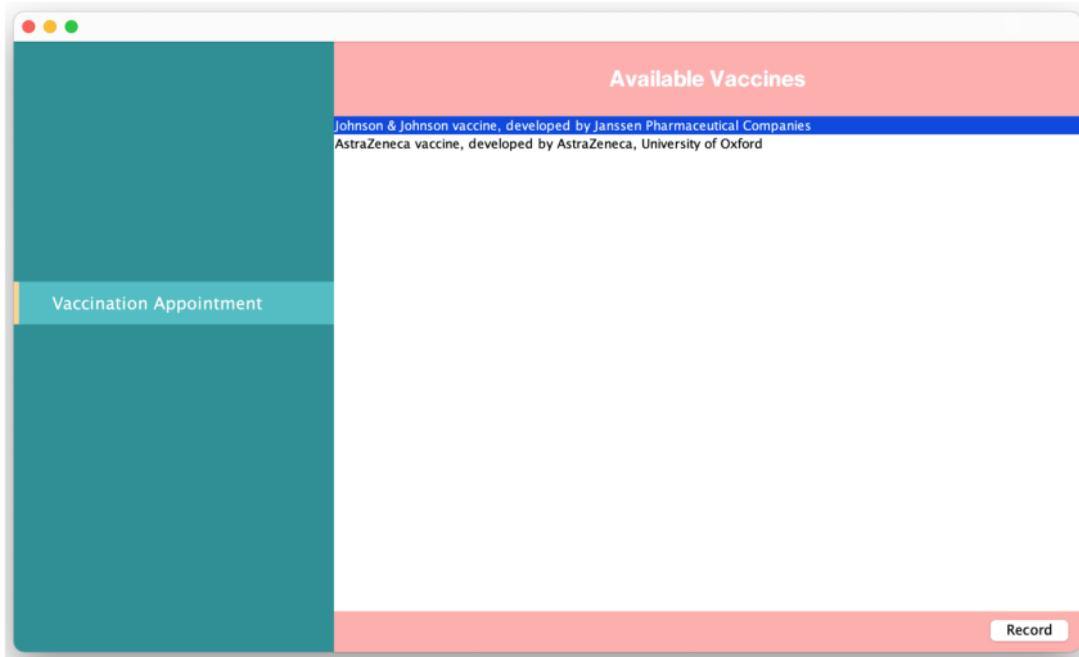
Picture 4.2: Request Vaccination Appointment

View Available Vaccine



Picture 4.3: Request Vaccination Appointment

View Available Vaccine

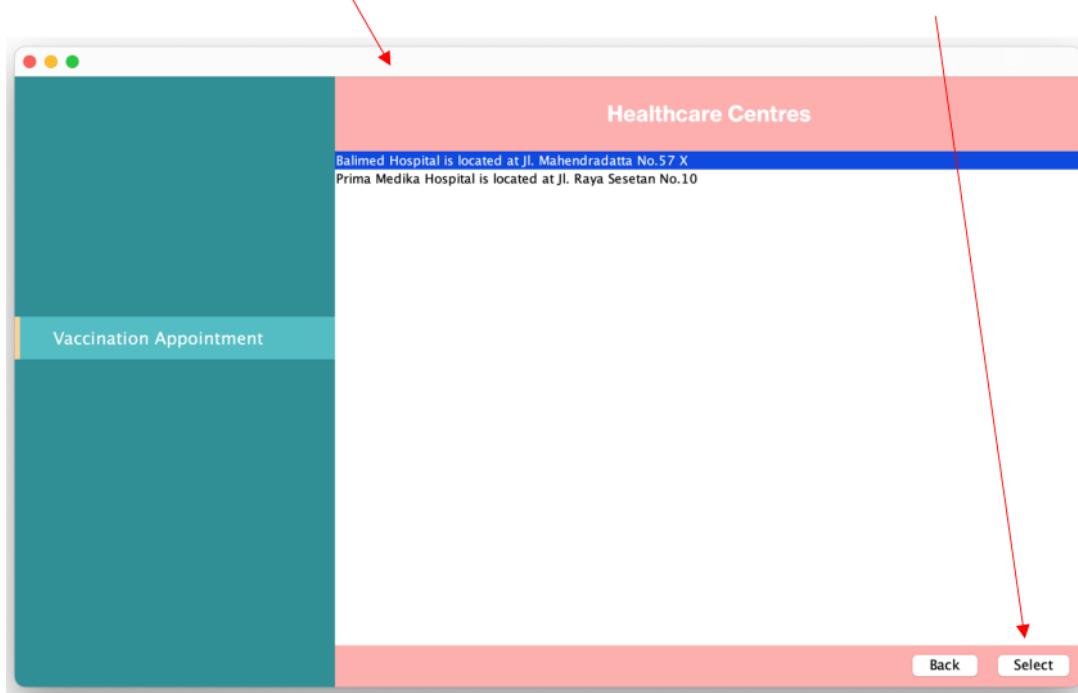


Picture 4.4: Request Vaccination Appointment

View Available Vaccine

Patient select healthcare to view vaccine batch information

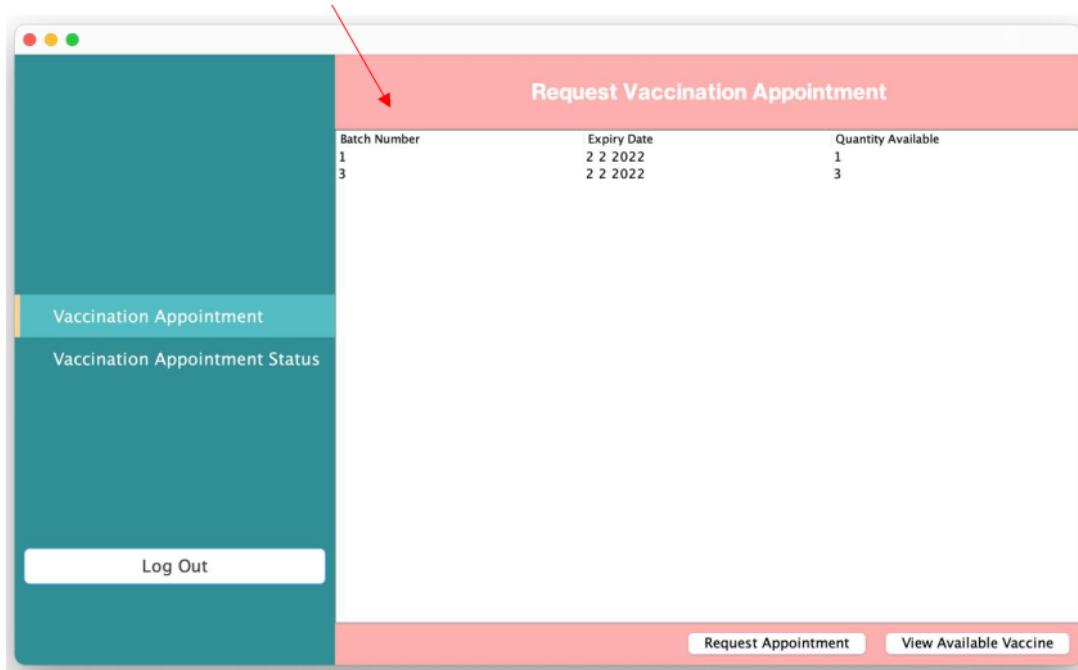
Patient clicks select



Picture 4.5: Request Vaccination Appointment

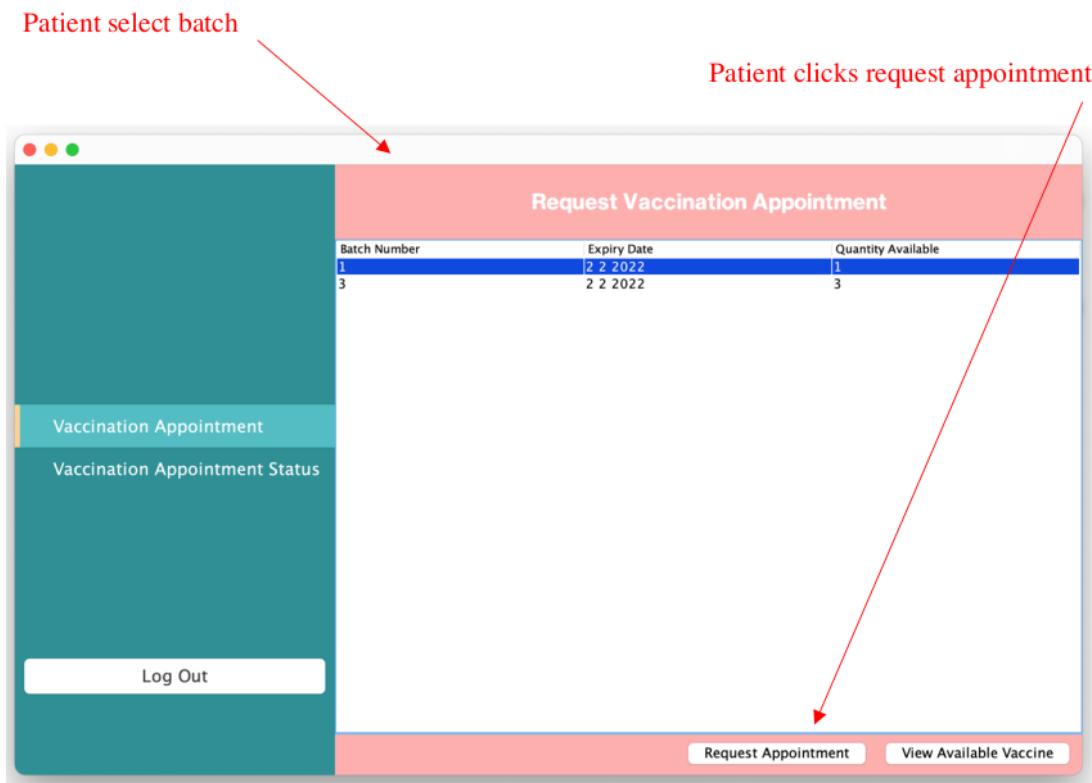
View Batch of Vaccine Information

Batch of vaccine that have quantity available and not expired displayed



Picture 4.6: Request Vaccination Appointment

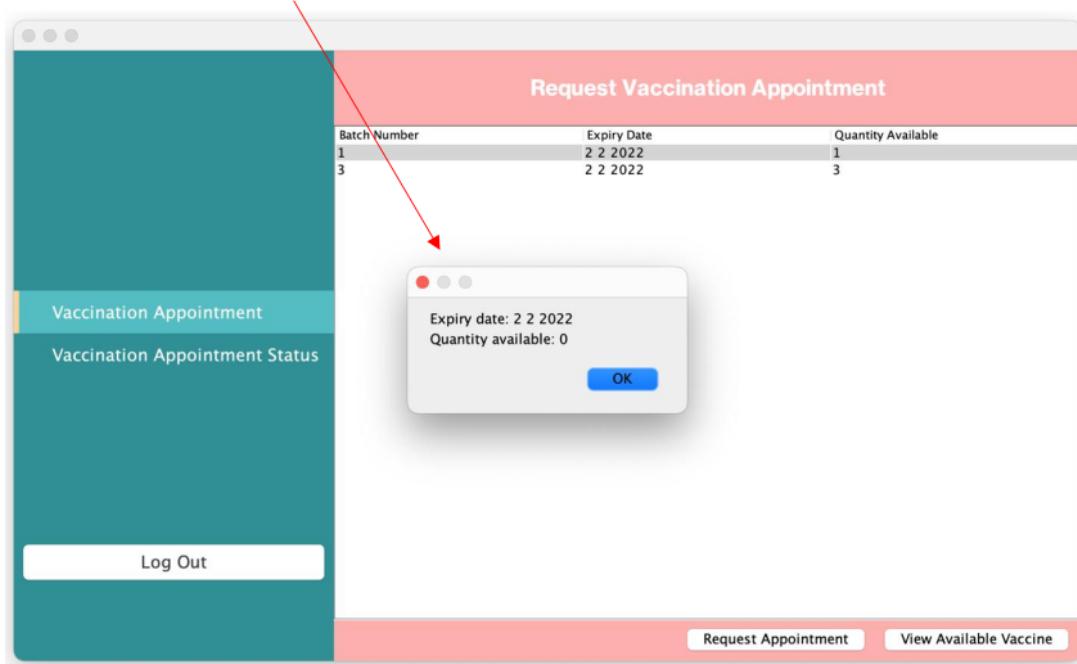
View Batch of Vaccine Information



Picture 4.7: Request Vaccination Appointment

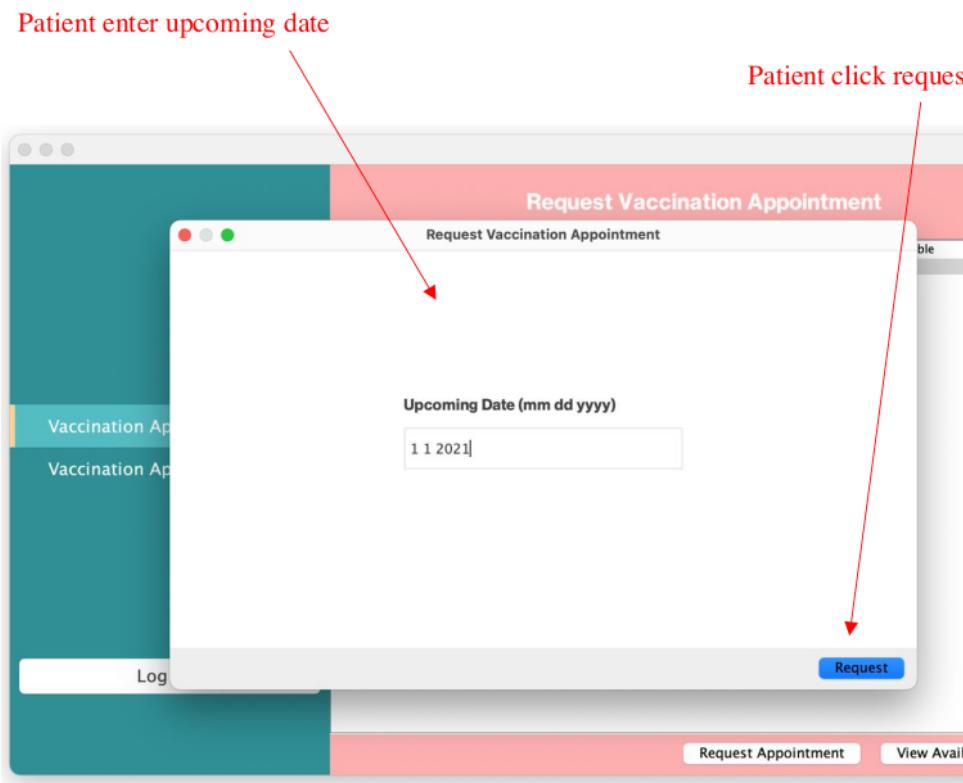
Request Appointment

Expiry date and quantity available is shown



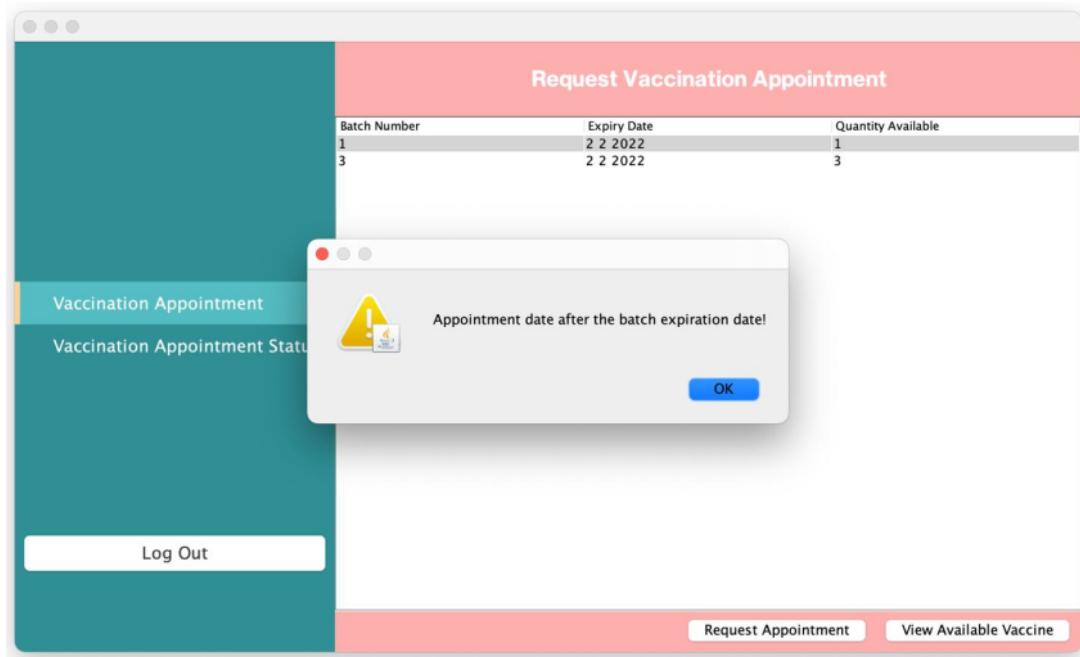
Picture 4.8: Request Vaccination Appointment

Request Appointment



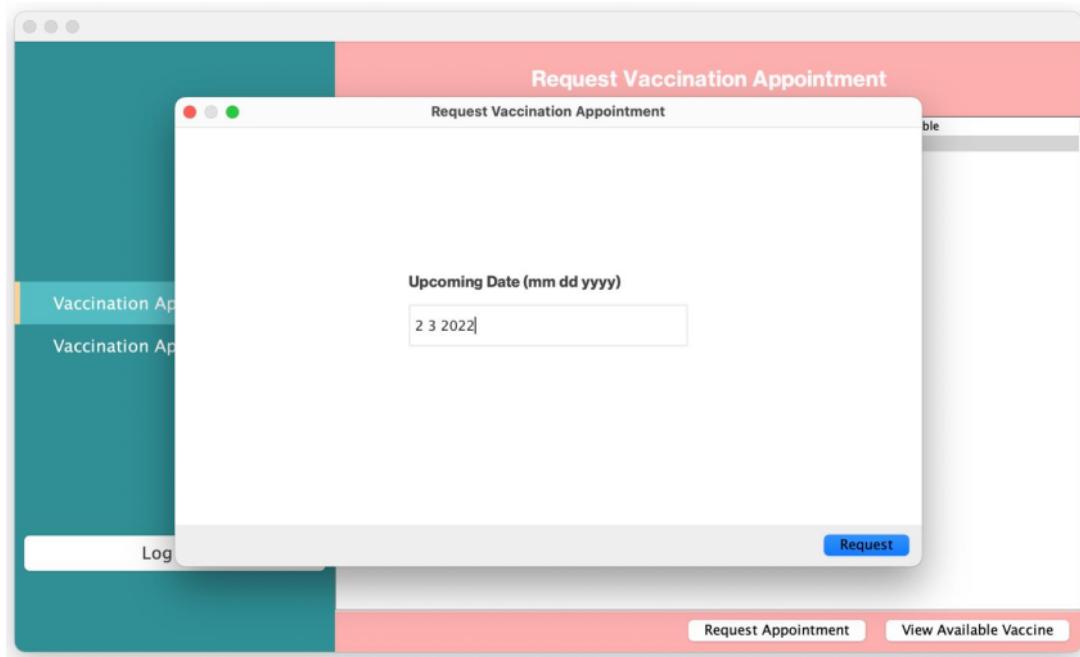
Picture 4.9: Request Vaccination Appointment

Date Validation



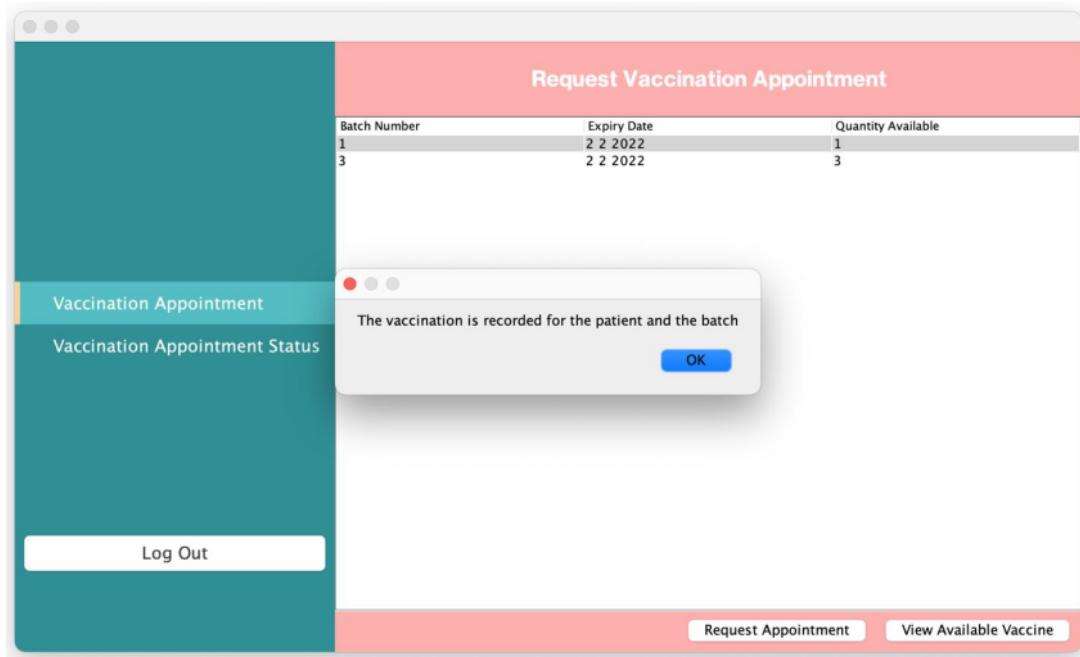
Picture 4.9.1: Request Vaccination Appointment

Date Validation



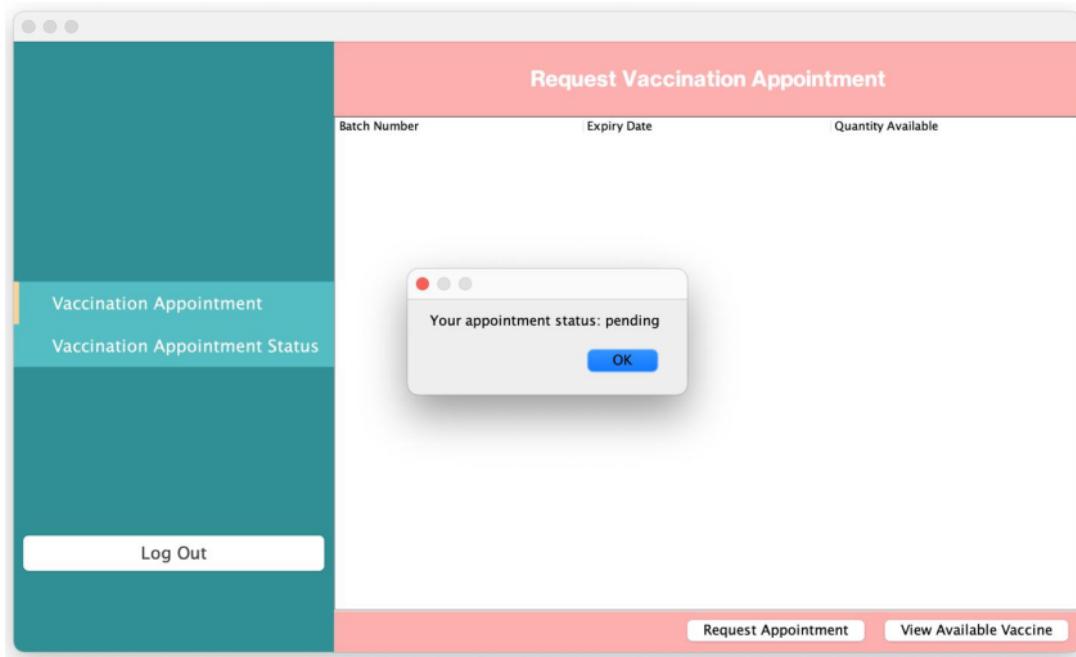
Picture 4.9.2: Request Vaccination Appointment

Date Validation



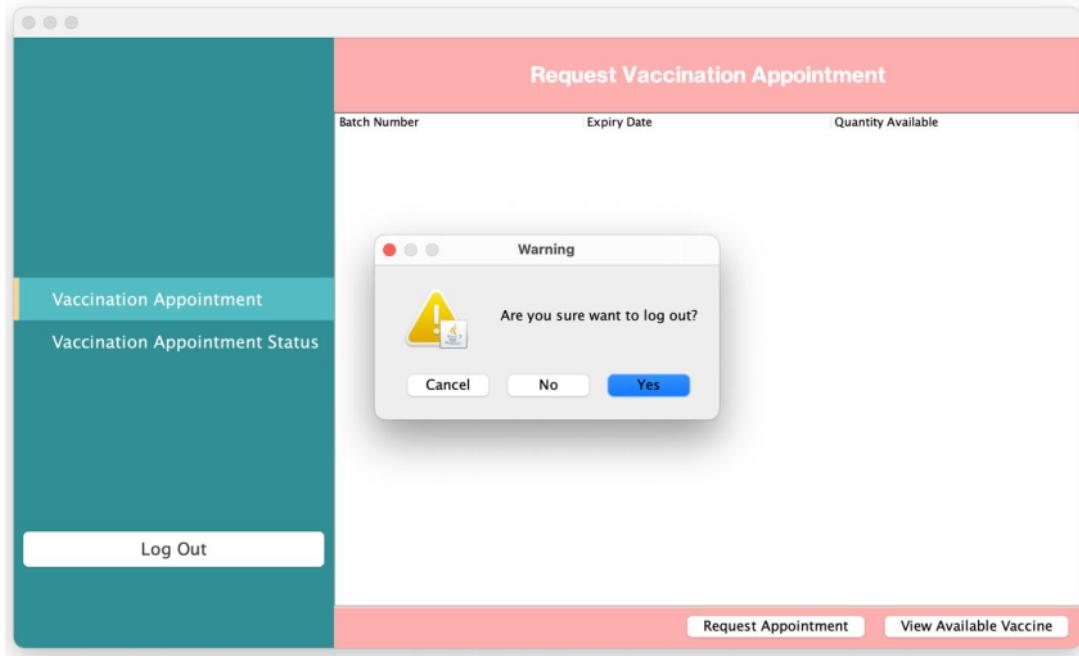
Picture 4.9.3: Request Vaccination Appointment

Request Appointment Success



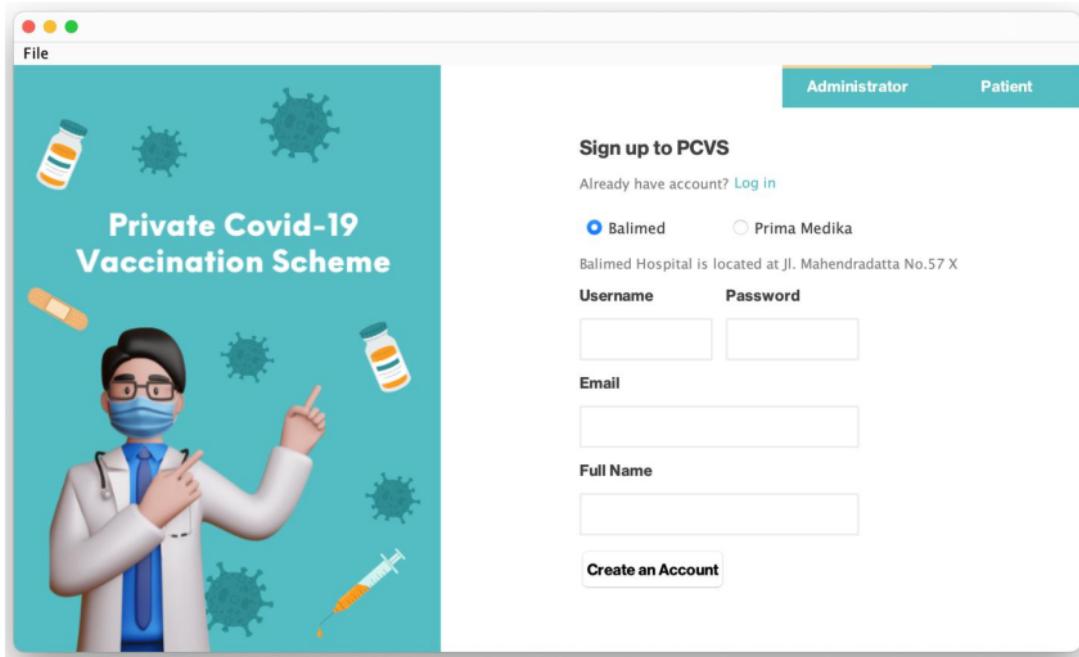
Picture 5: Request Vaccination Appointment

View Appointment Status



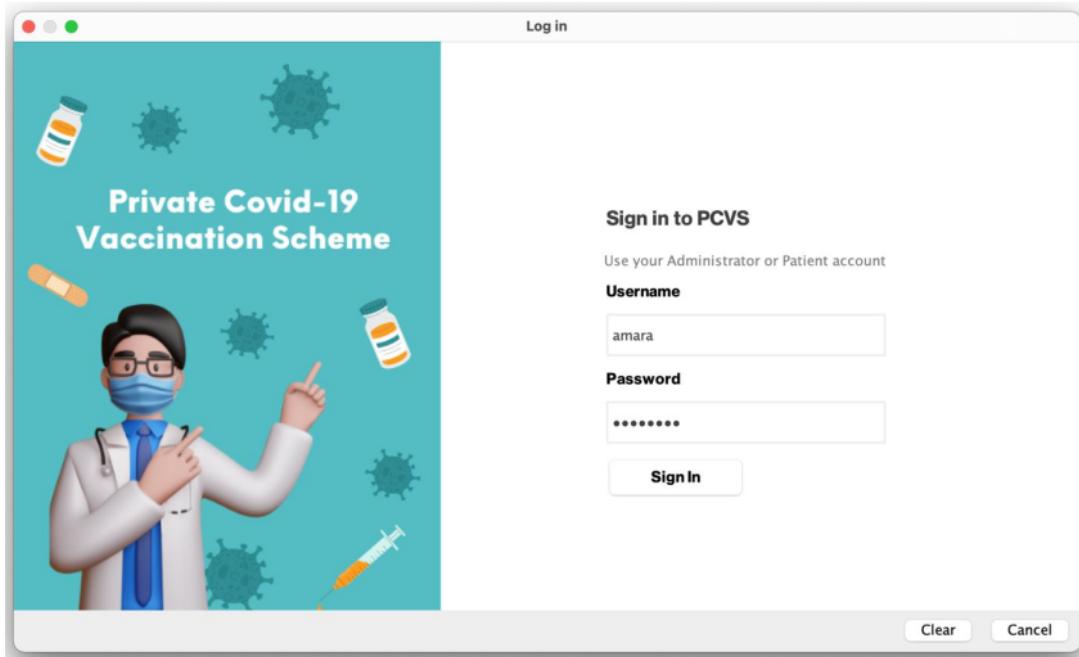
Picture 5: Request Vaccination Appointment

Log Out

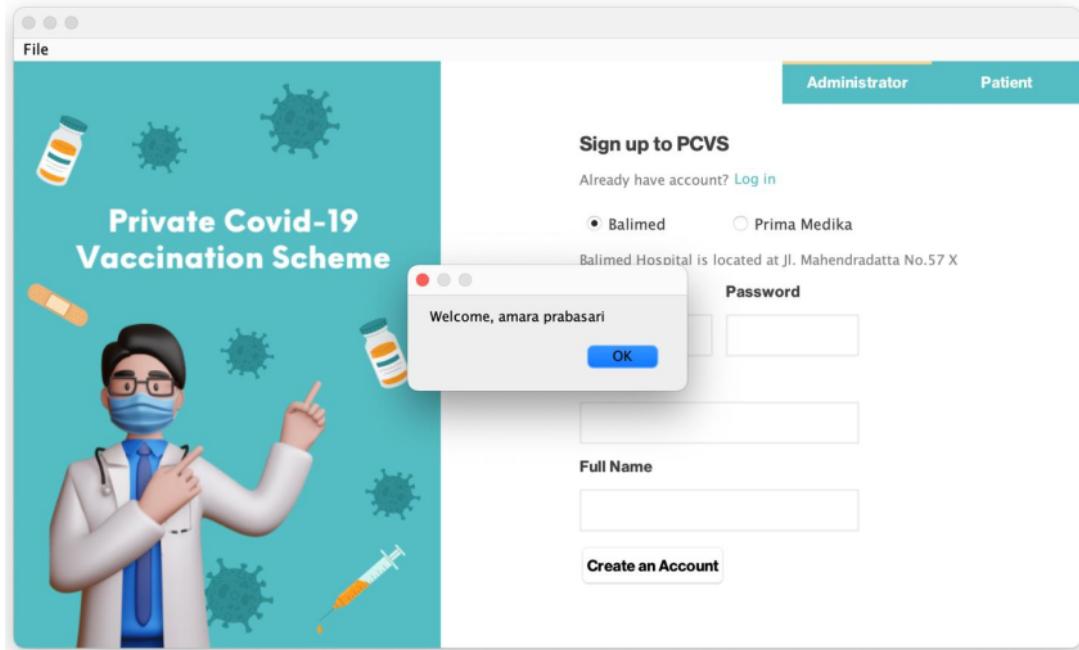


Picture 5.1: Request Vaccination Appointment

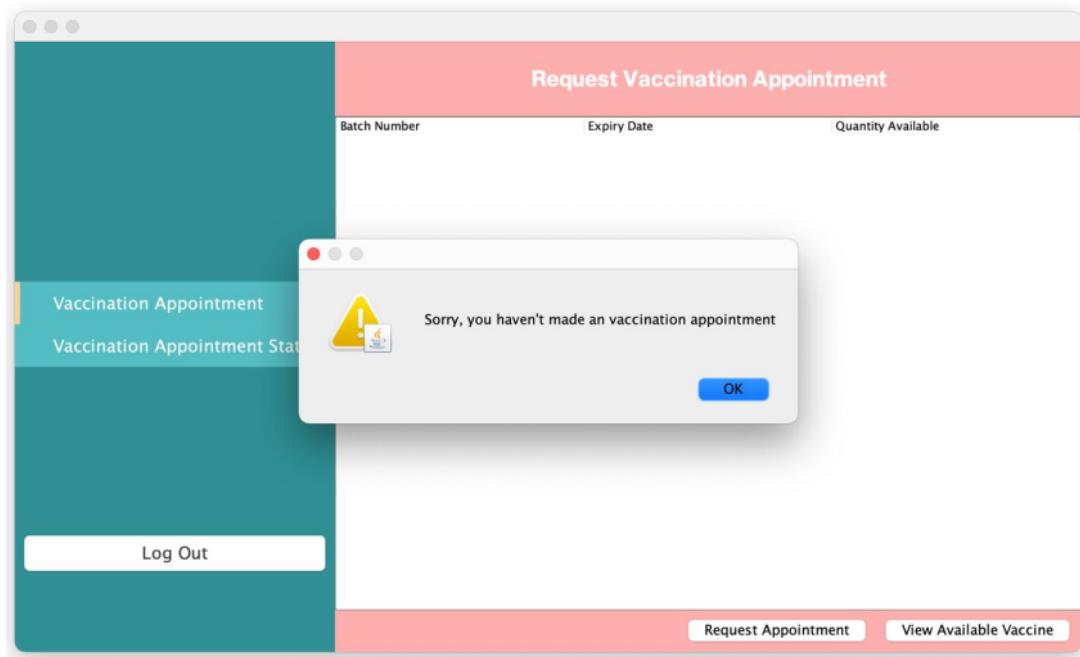
Log Out



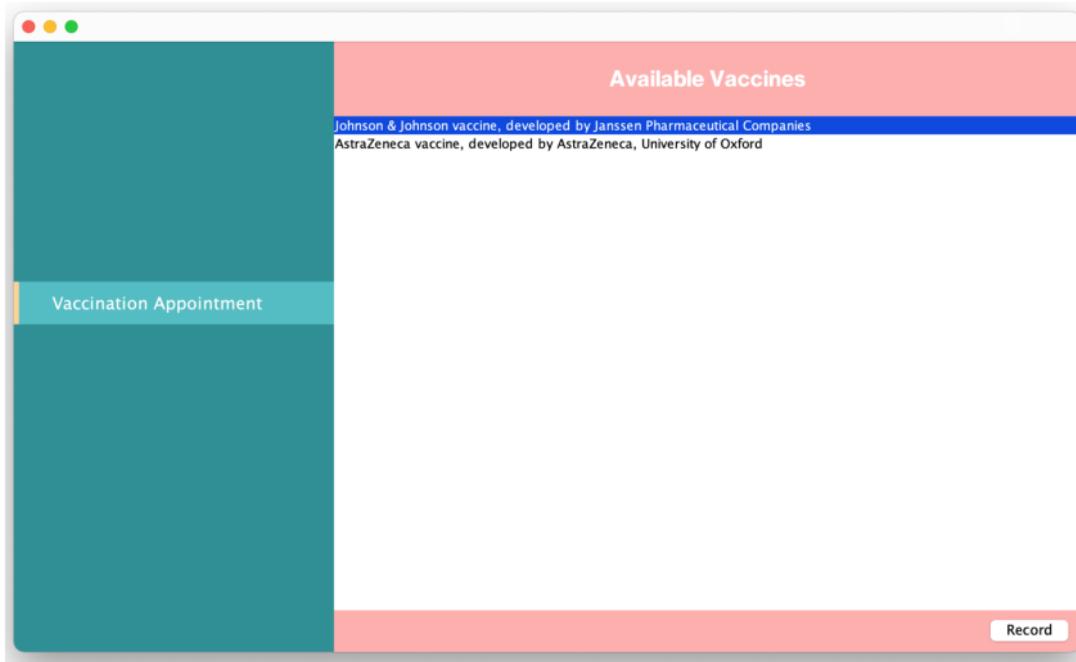
Picture 6: Request Vaccination Appointment



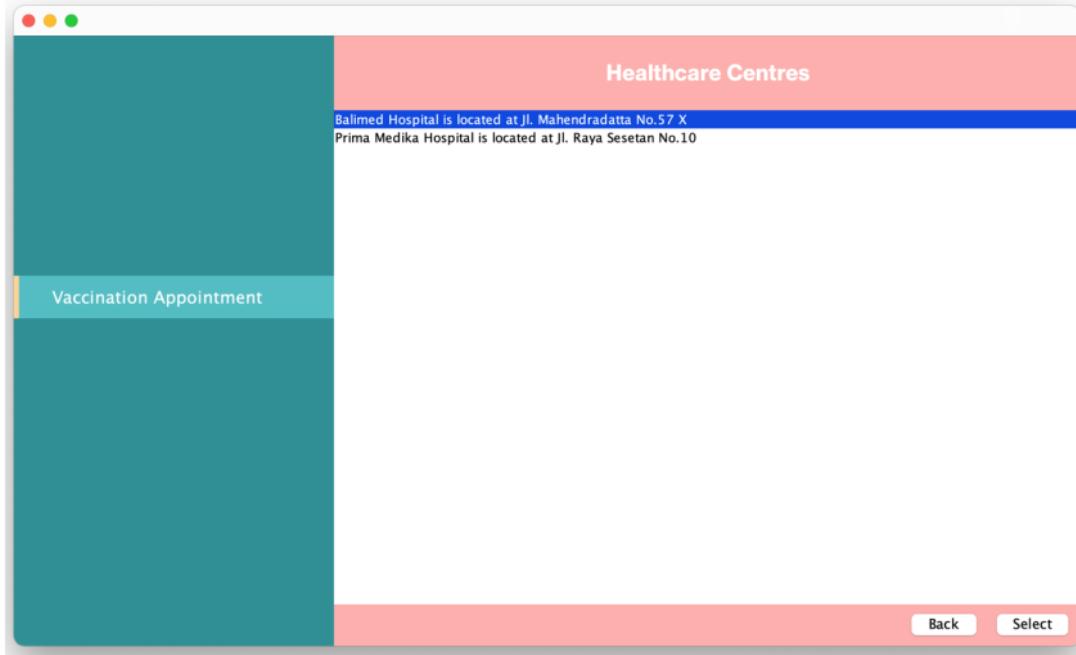
Picture 6.1: Request Vaccination Appointment



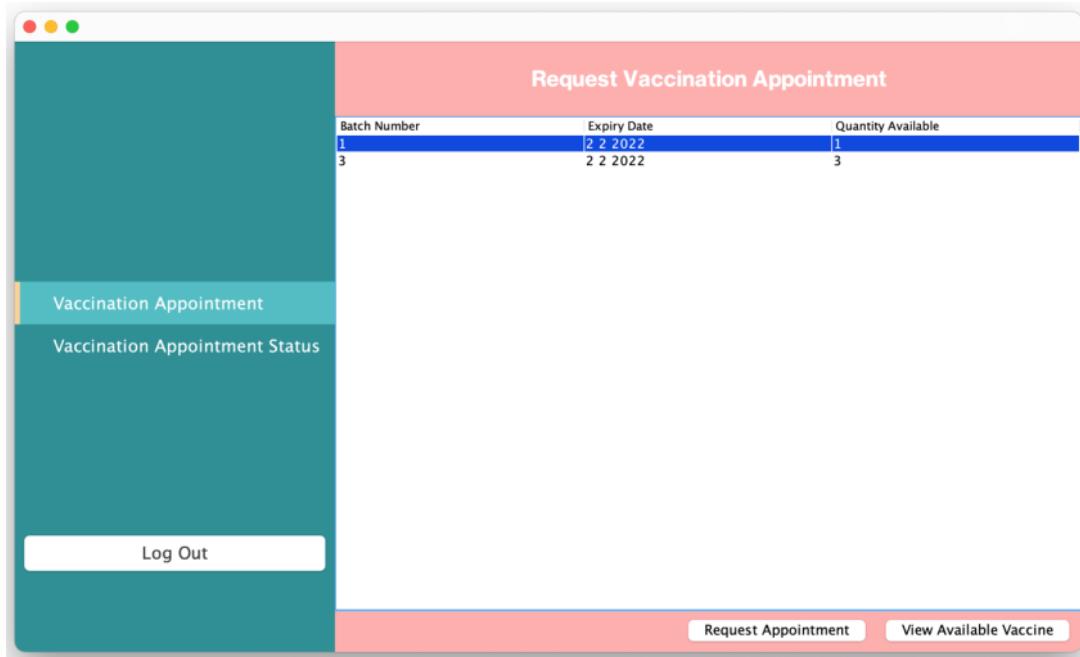
Picture 6.1: Request Vaccination Appointment



Picture 6.2: Request Vaccination Appointment

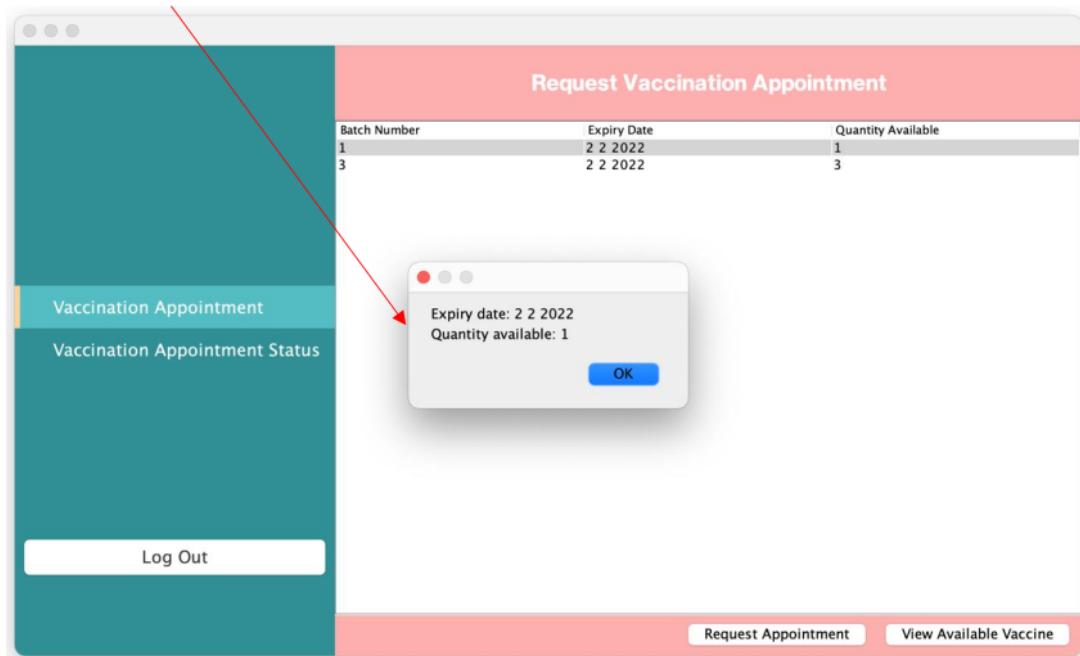


Picture 6.3: Request Vaccination Appointment

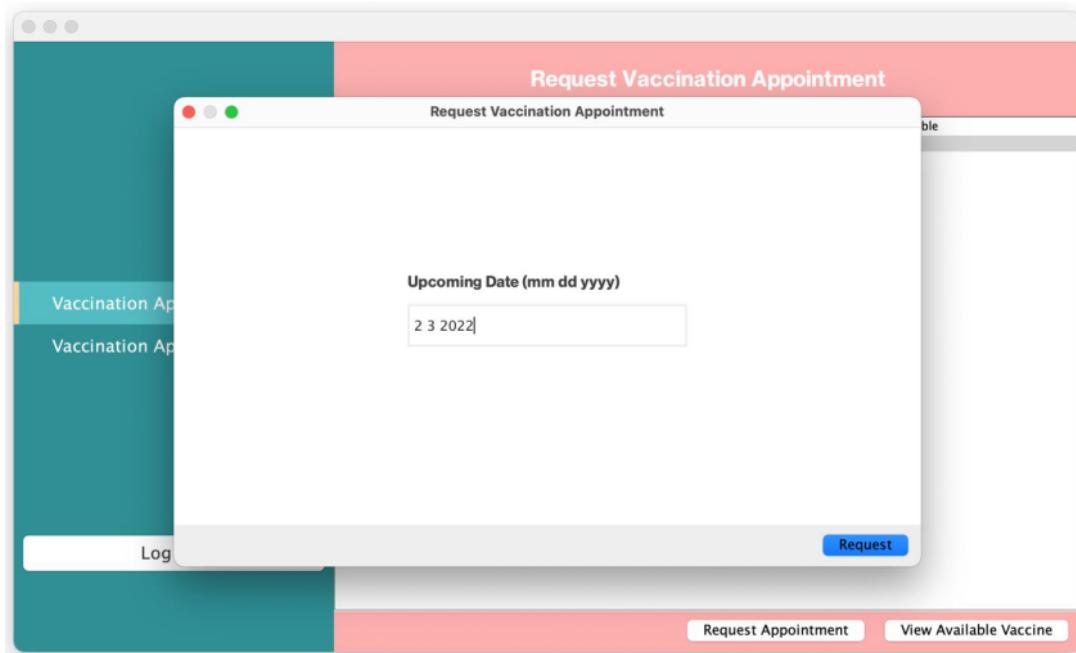


Picture 6.4: Request Vaccination Appointment

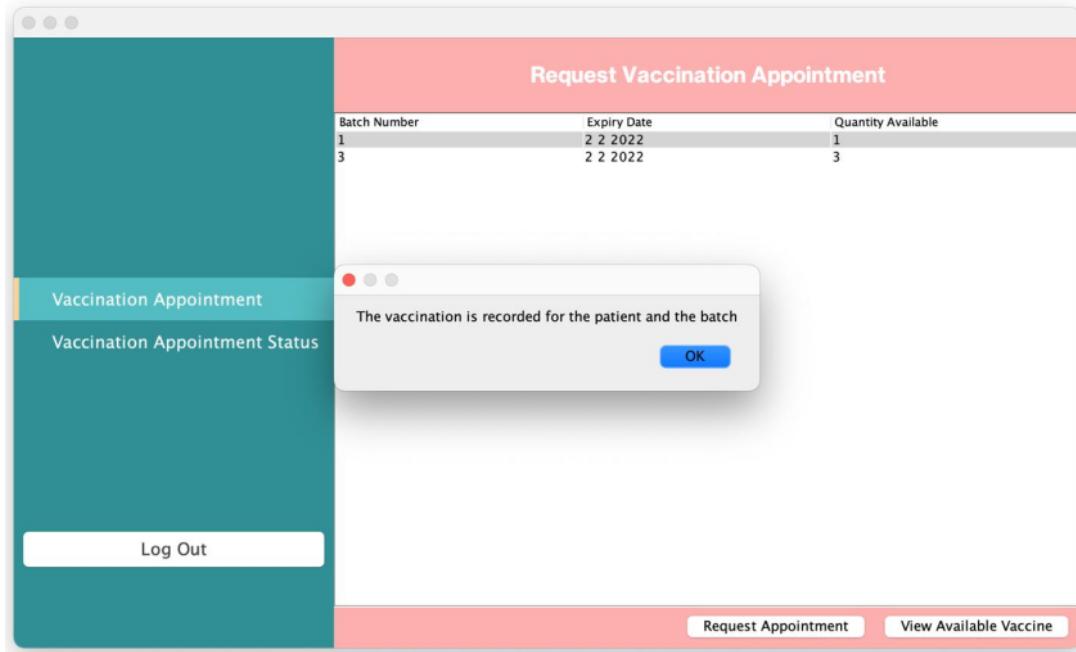
Quantity available increase



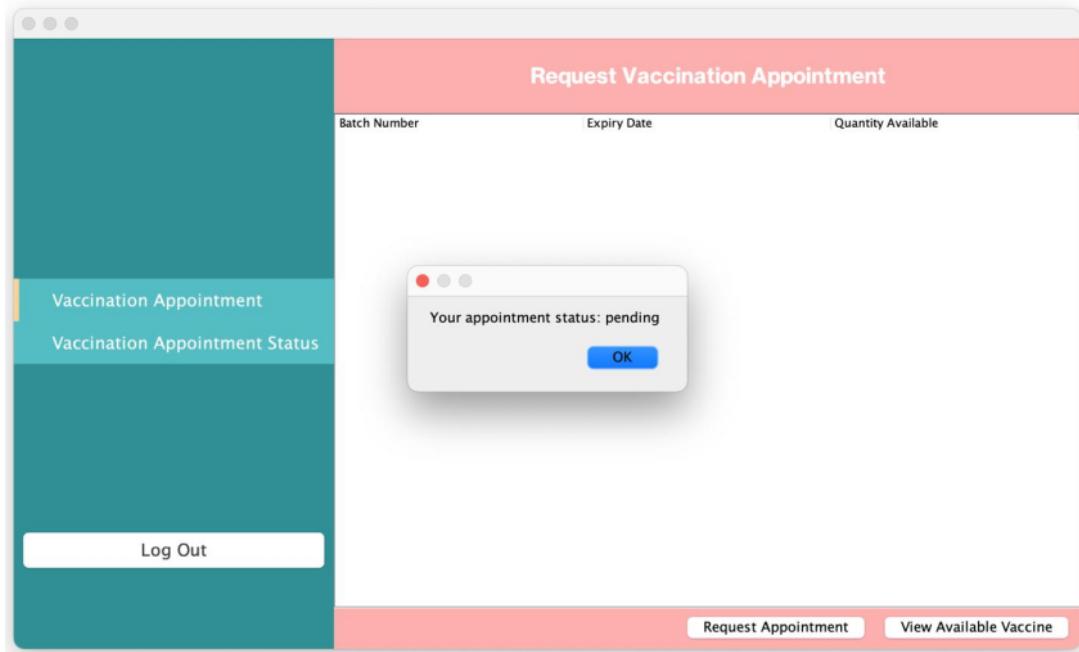
Picture 6.5: Request Vaccination Appointment



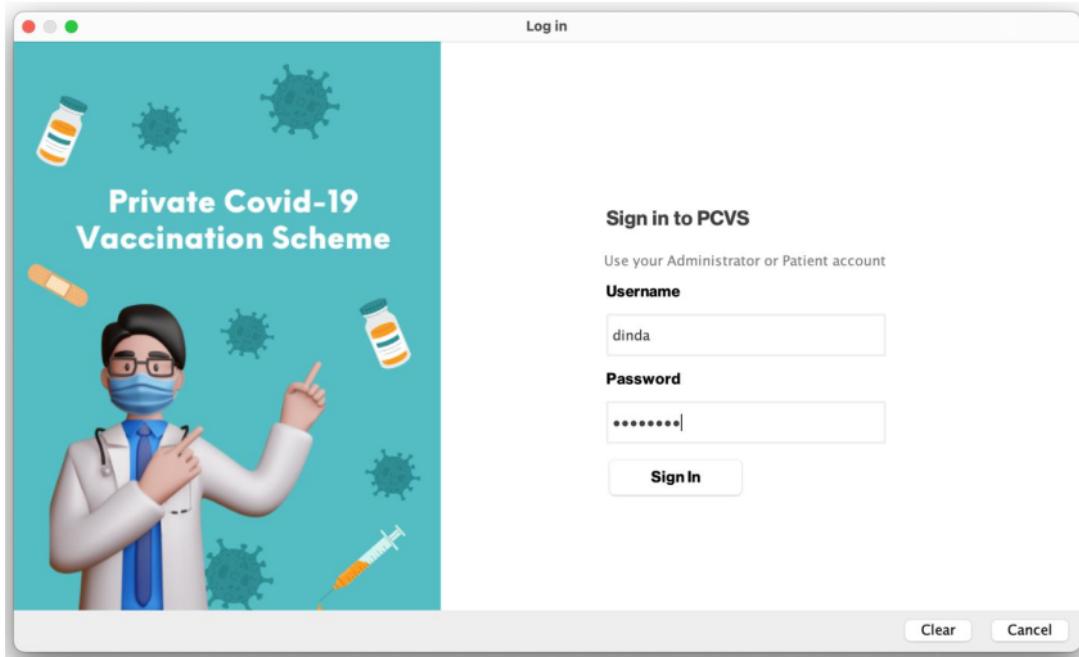
Picture 6.6: Request Vaccination Appointment



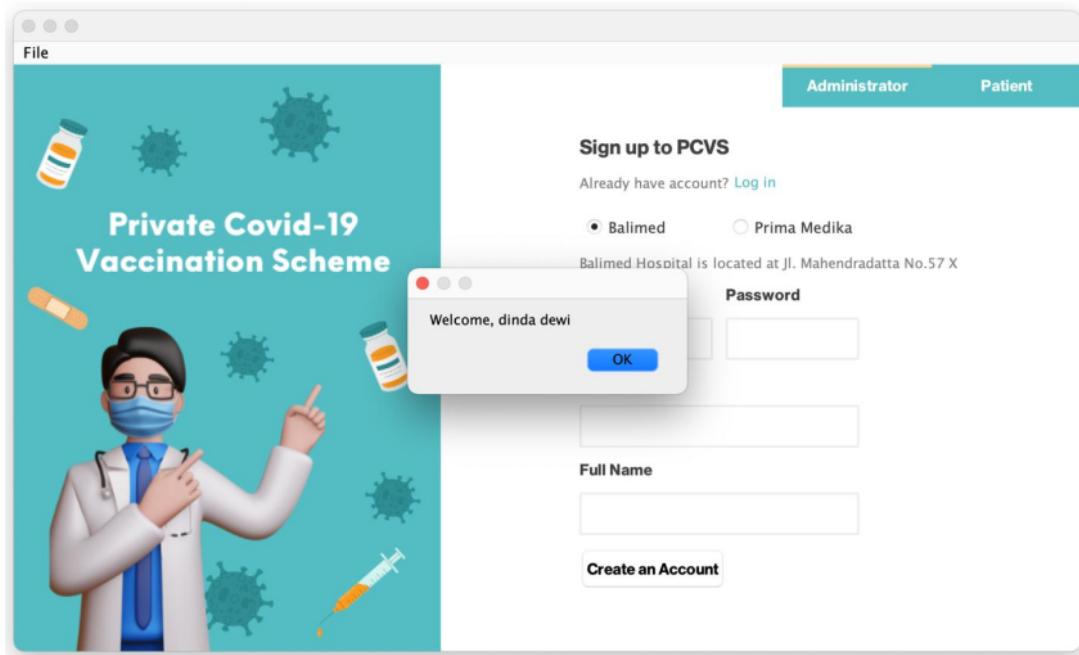
Picture 6.6: Request Vaccination Appointment



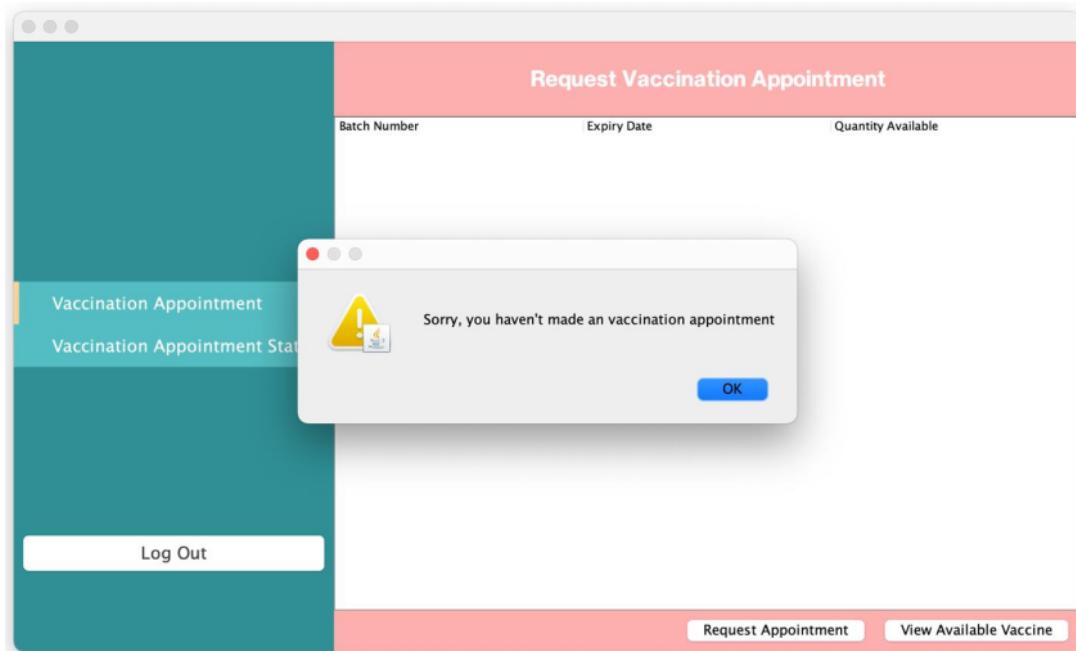
Picture 6.6: Request Vaccination Appointment



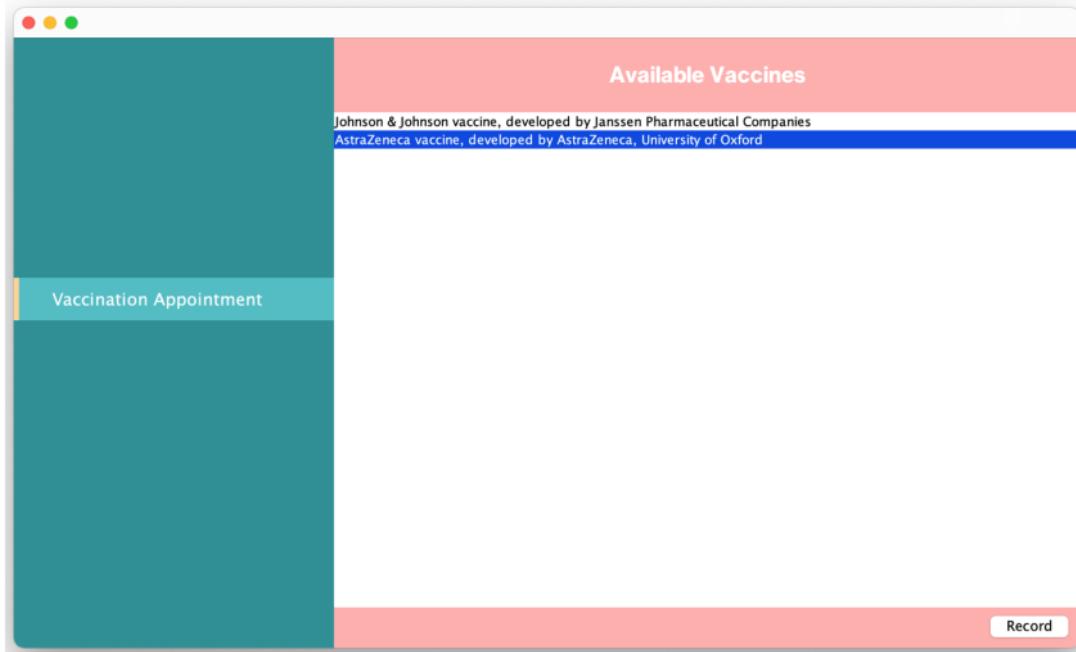
Picture 7: Request Vaccination Appointment



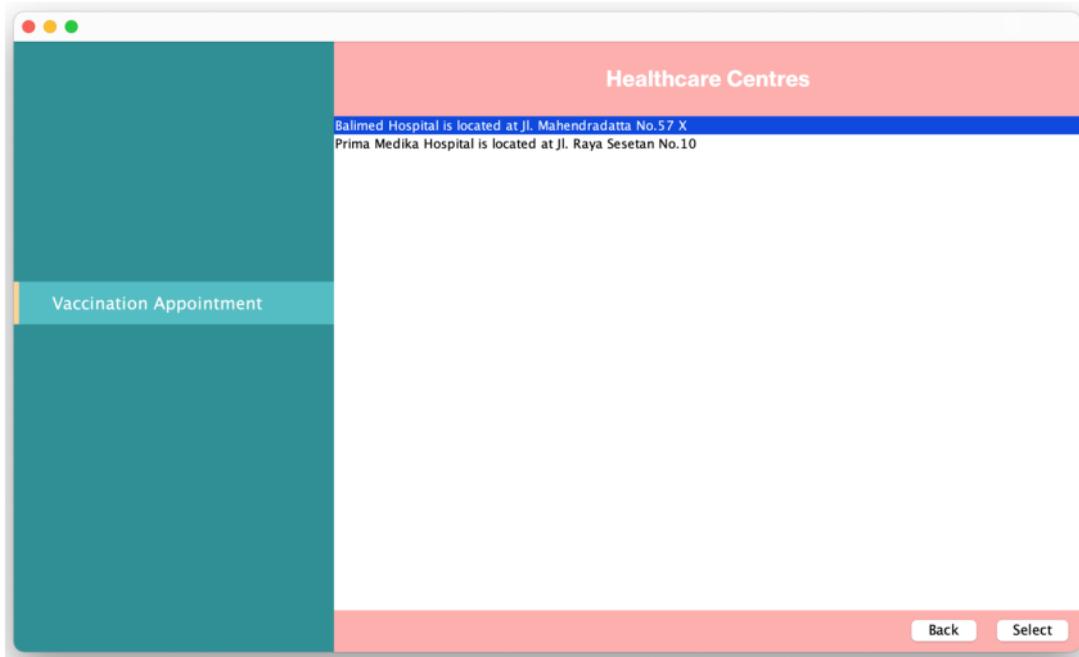
Picture 7.1: Request Vaccination Appointment



Picture 7.2: Request Vaccination Appointment

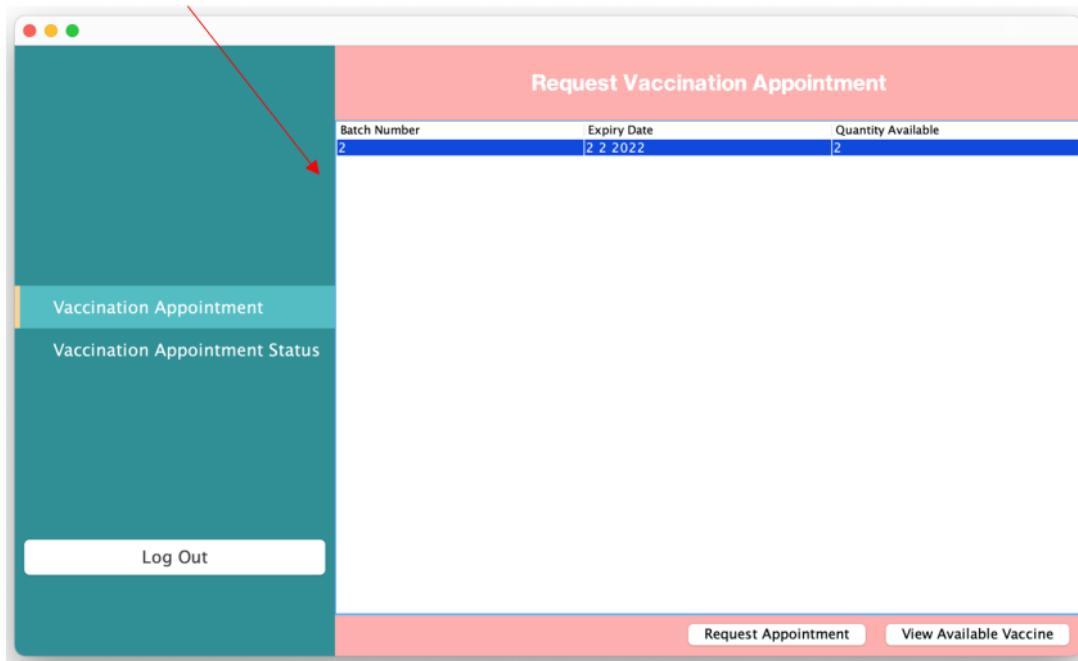


Picture 7.3: Request Vaccination Appointment

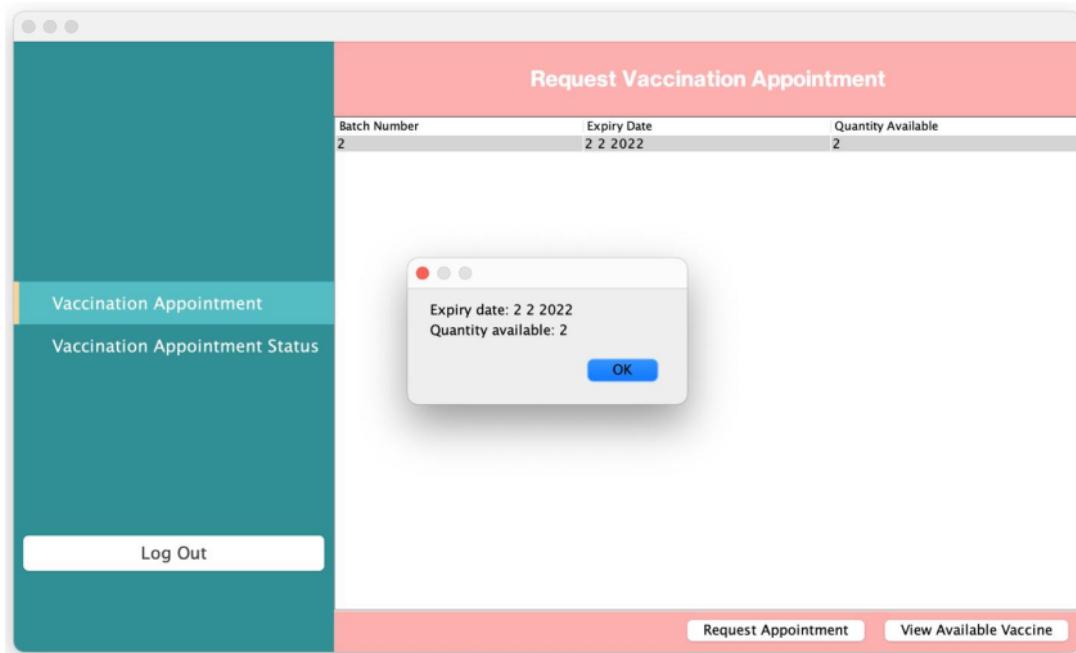


Picture 7.4: Request Vaccination Appointment

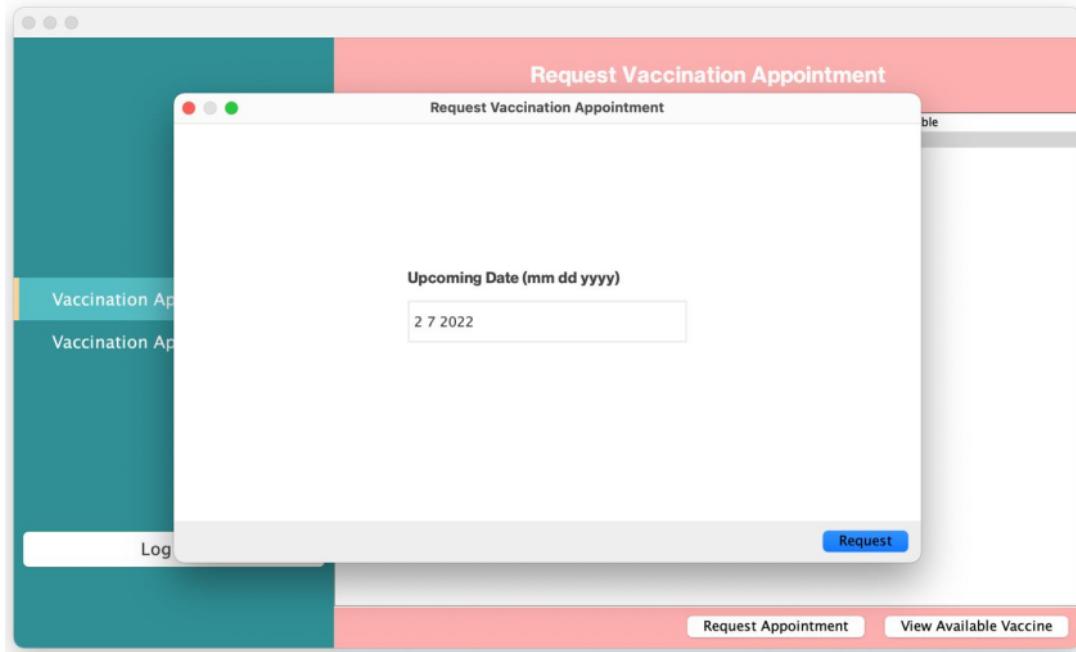
Batch number 4 is not shown because expired



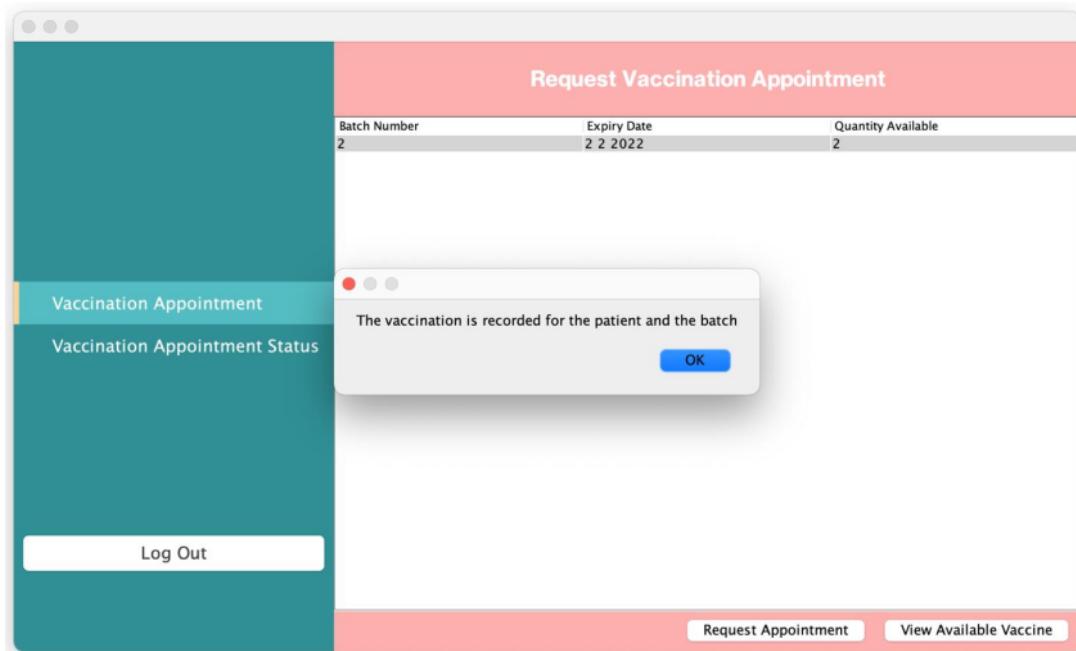
Picture 7.5: Request Vaccination Appointment



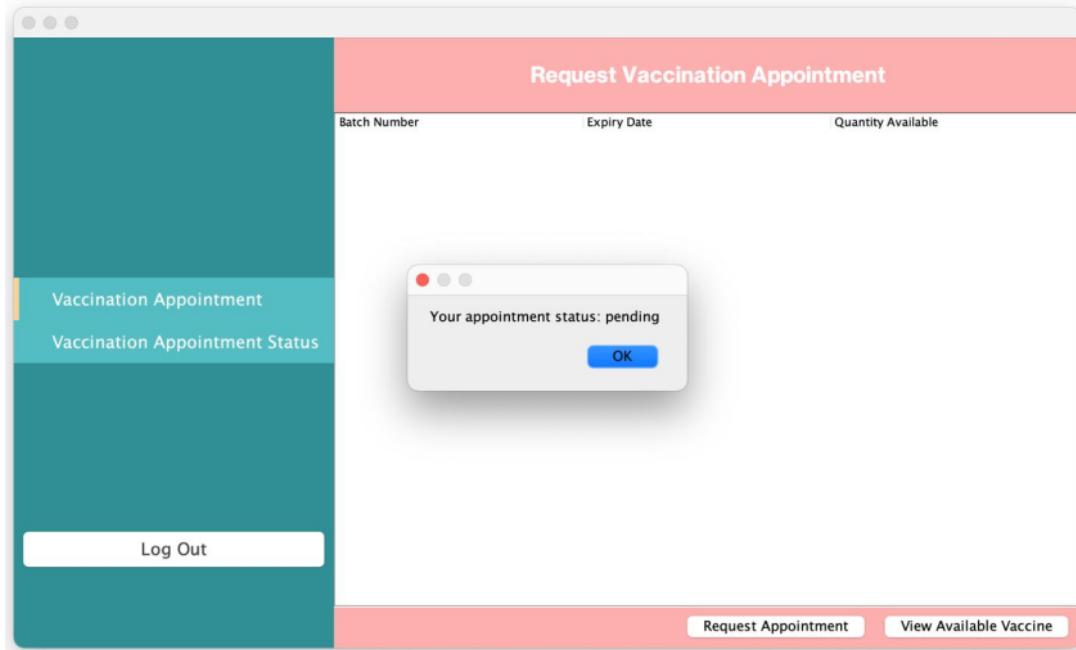
Picture 7.6: Request Vaccination Appointment



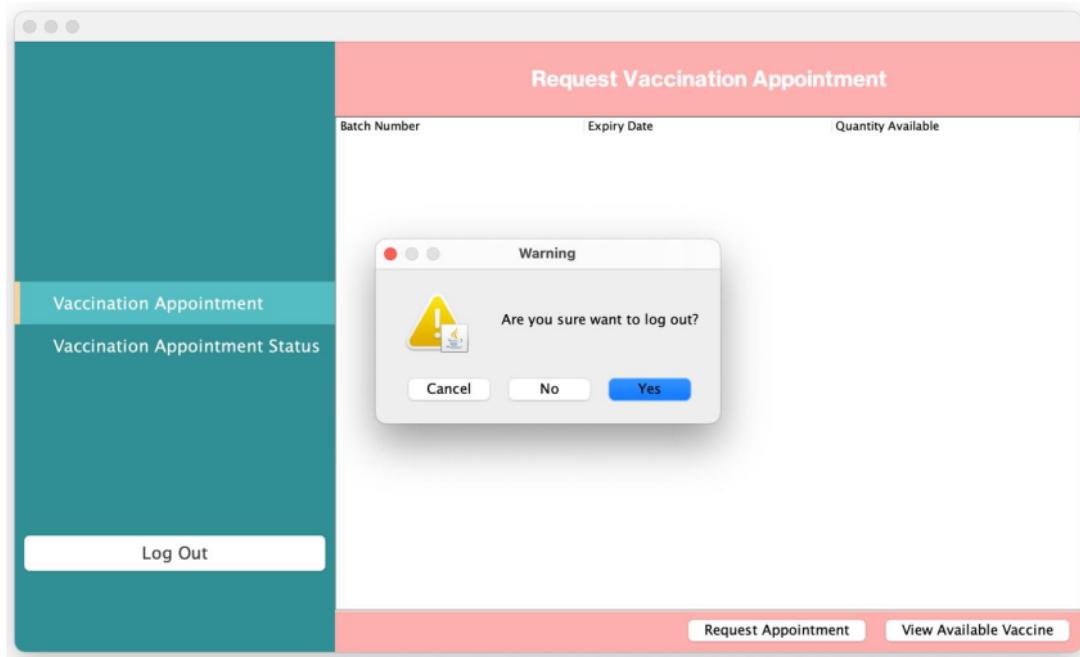
Picture 7.7: Request Vaccination Appointment



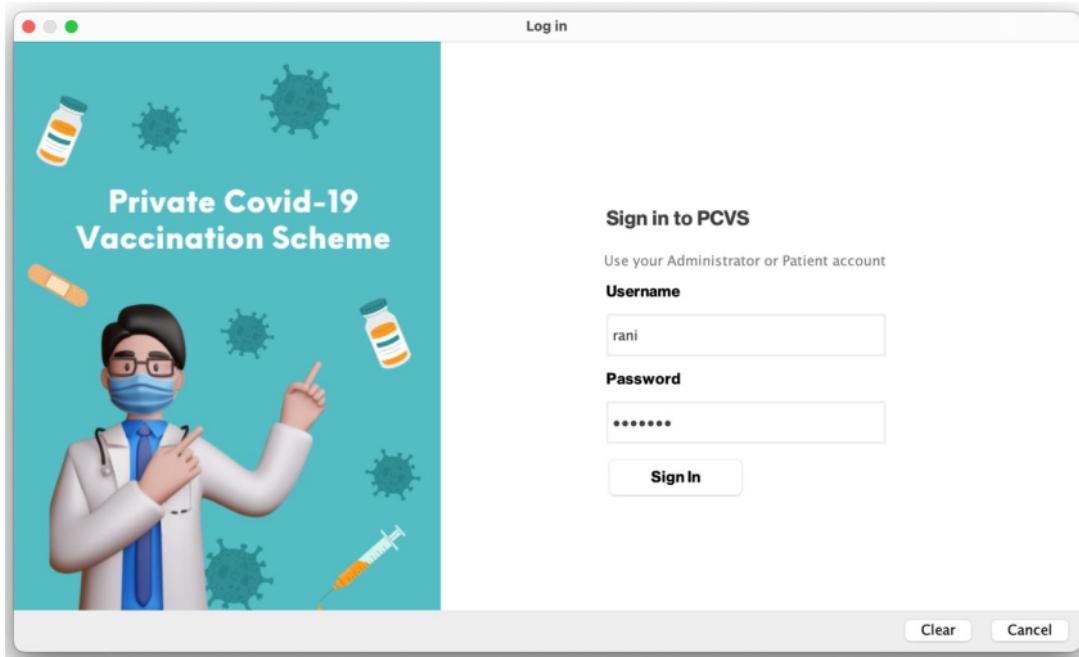
Picture 7.8: Request Vaccination Appointment



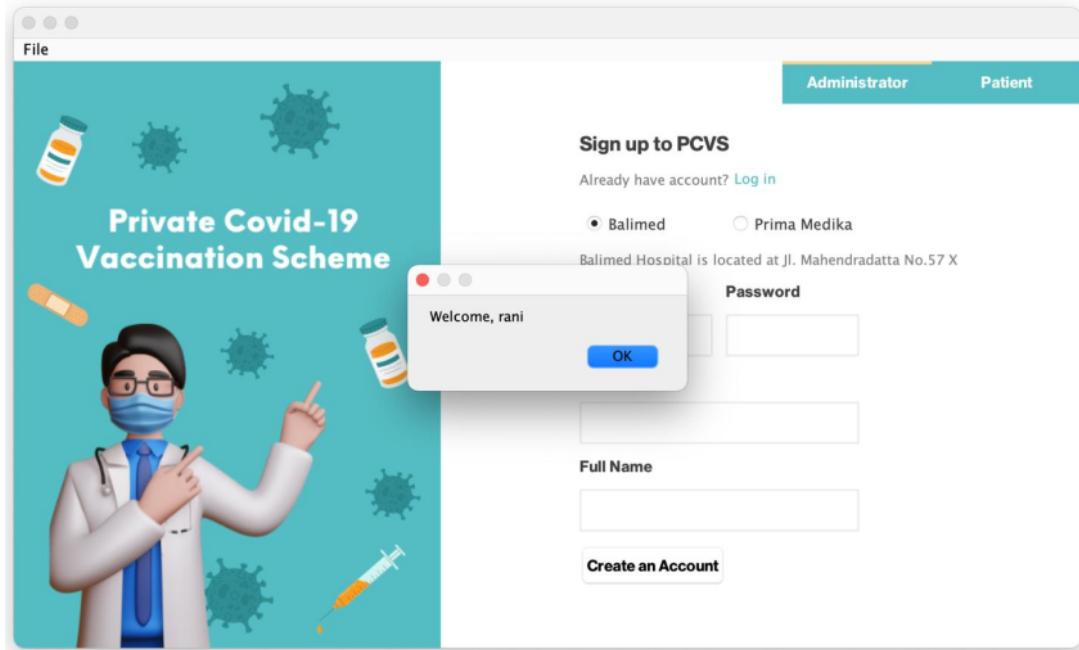
Picture 7.9: Request Vaccination Appointment



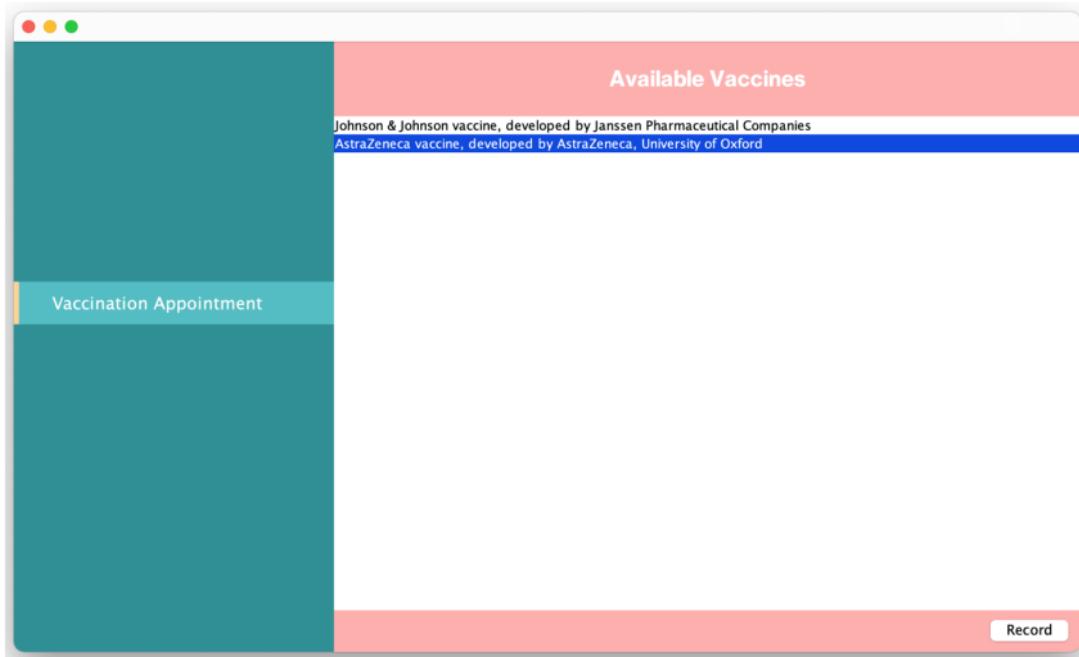
Picture 7.9.1: Request Vaccination Appointment



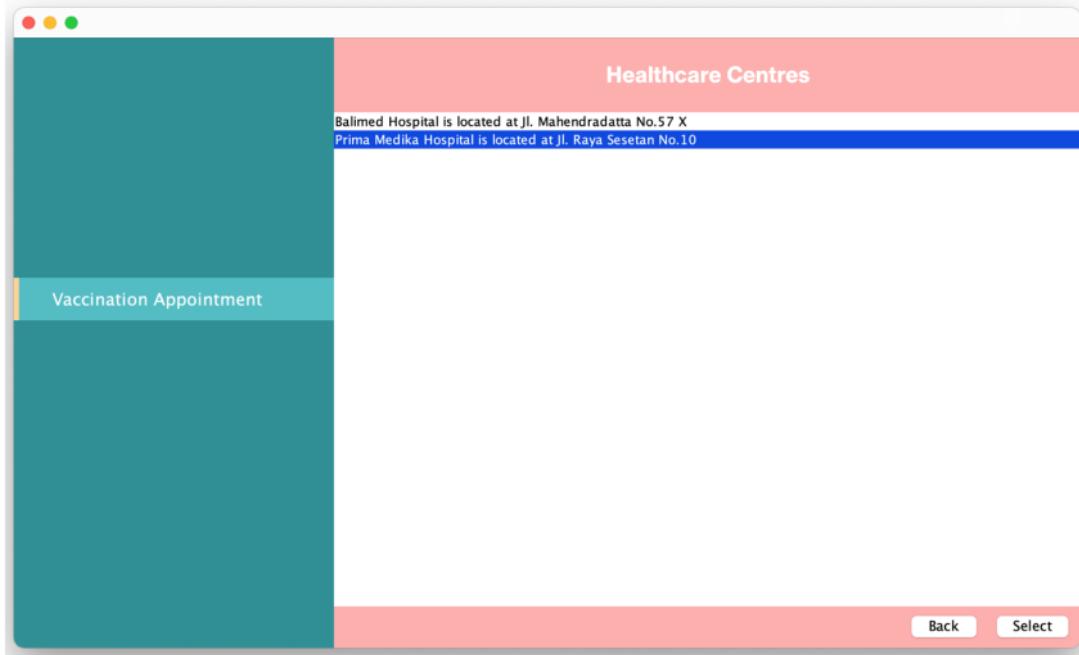
Picture 8: Request Vaccination Appointment



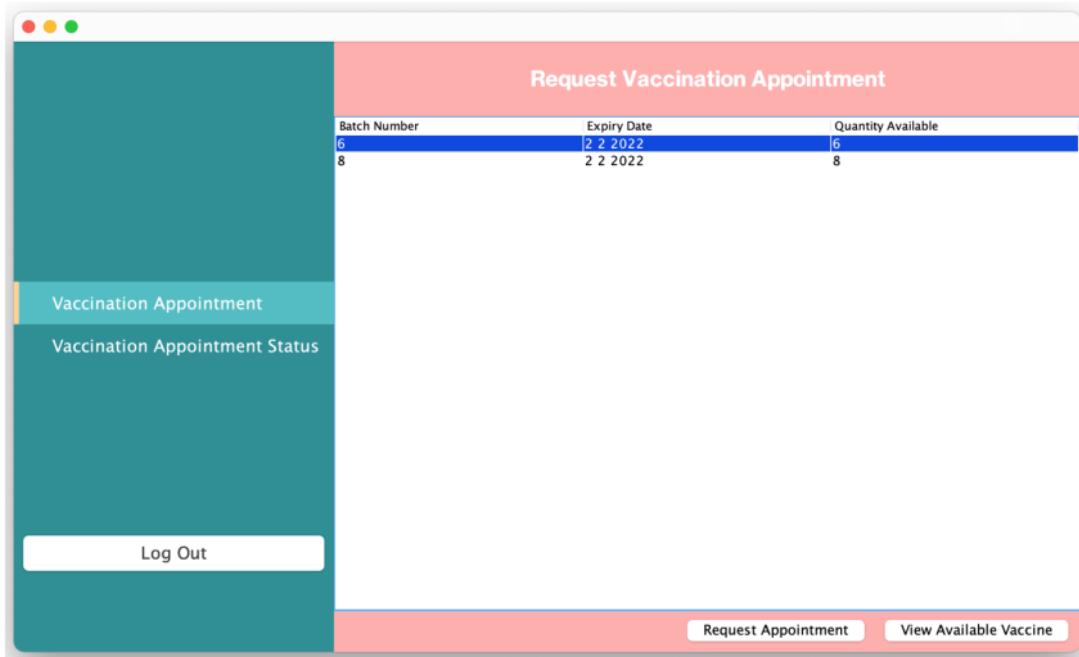
Picture 8.1: Request Vaccination Appointment



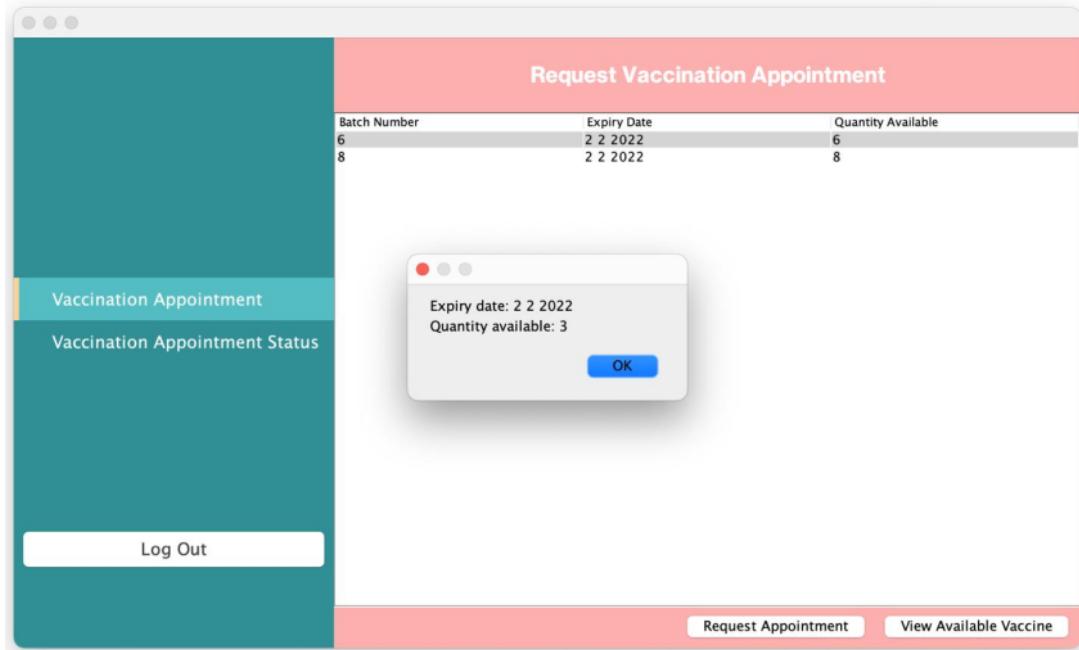
Picture 8.2: Request Vaccination Appointment



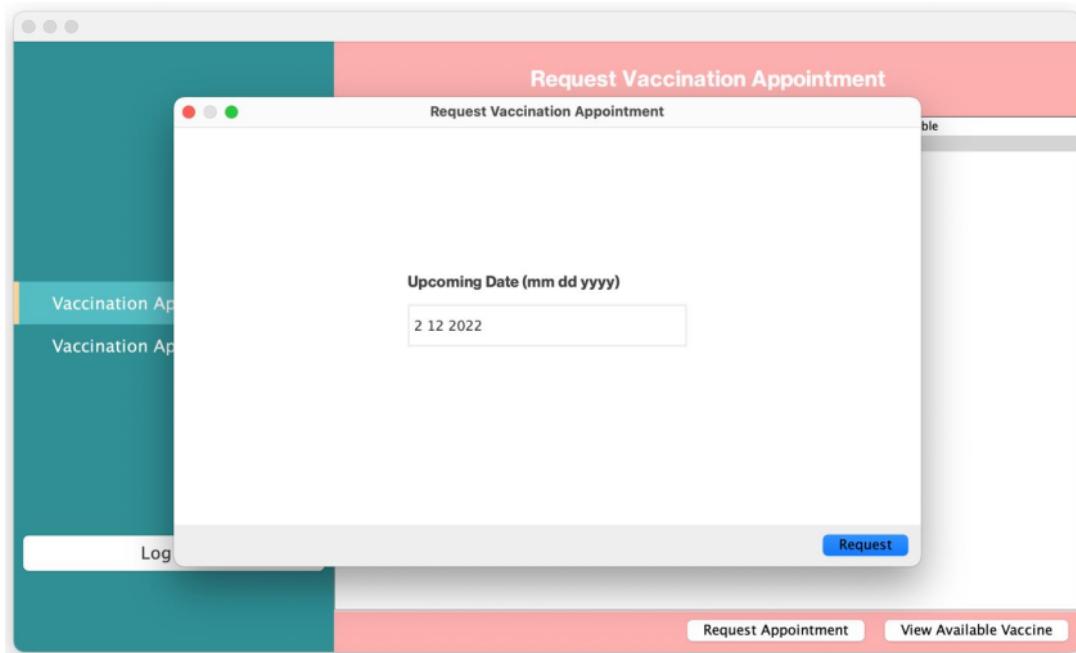
Picture 8.3: Request Vaccination Appointment



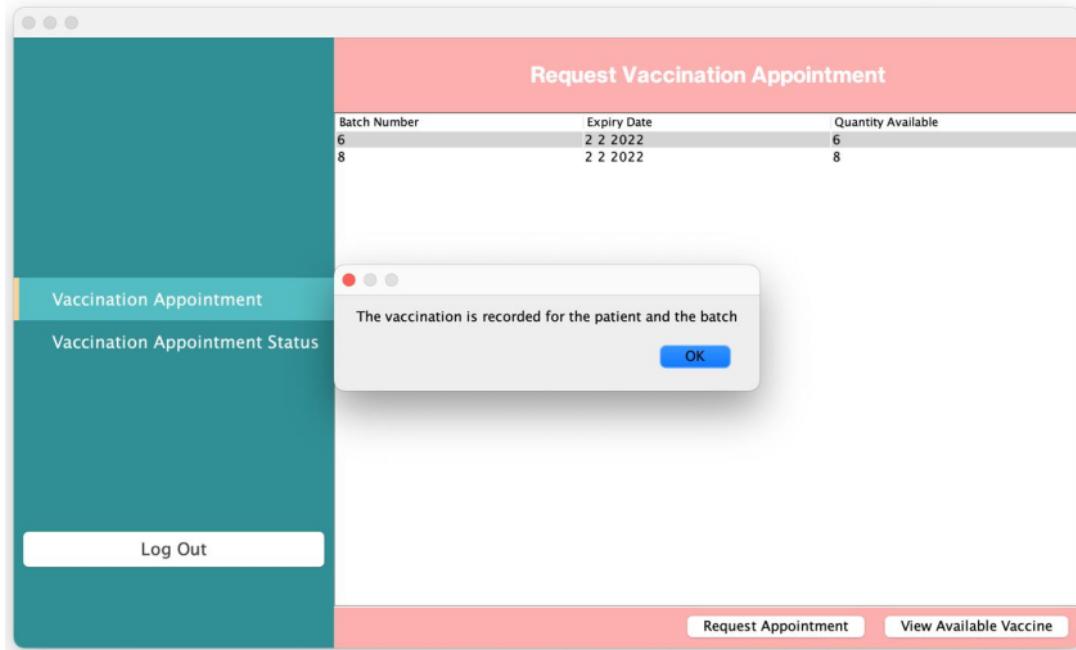
Picture 8.4: Request Vaccination Appointment



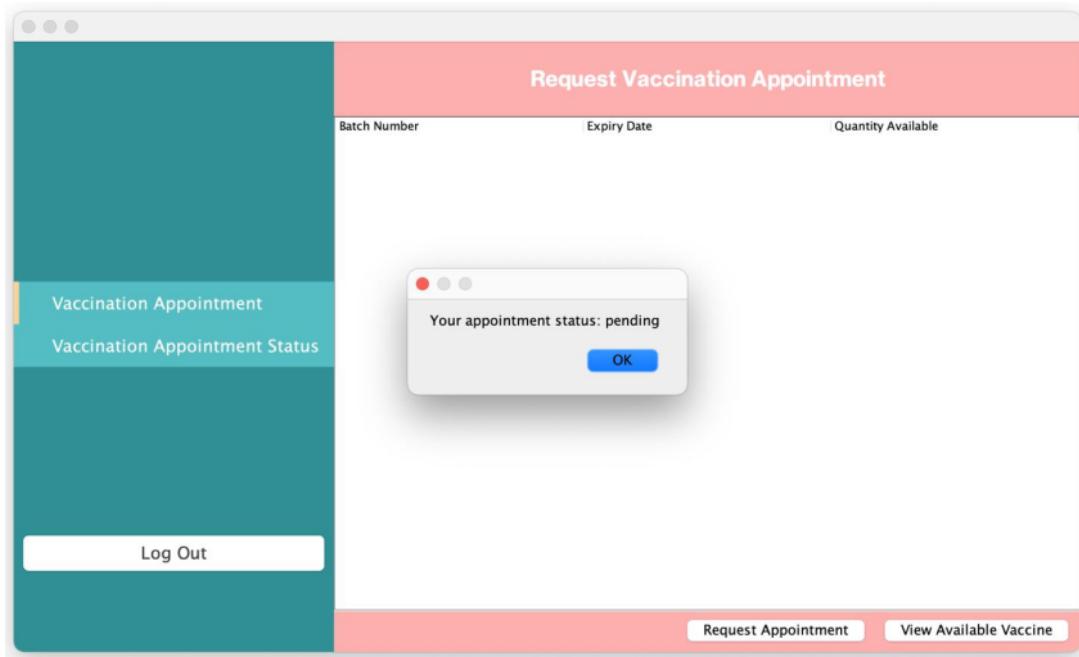
Picture 8.5: Request Vaccination Appointment



Picture 8.6: Request Vaccination Appointment



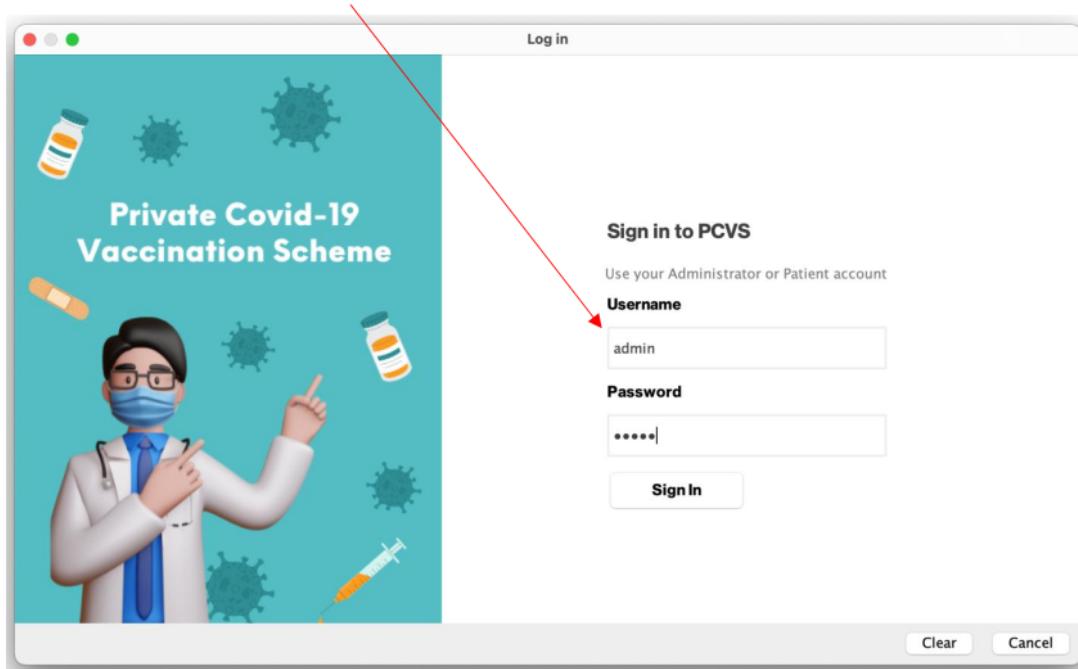
Picture 8.7: Request Vaccination Appointment



Picture 8.8: Request Vaccination Appointment

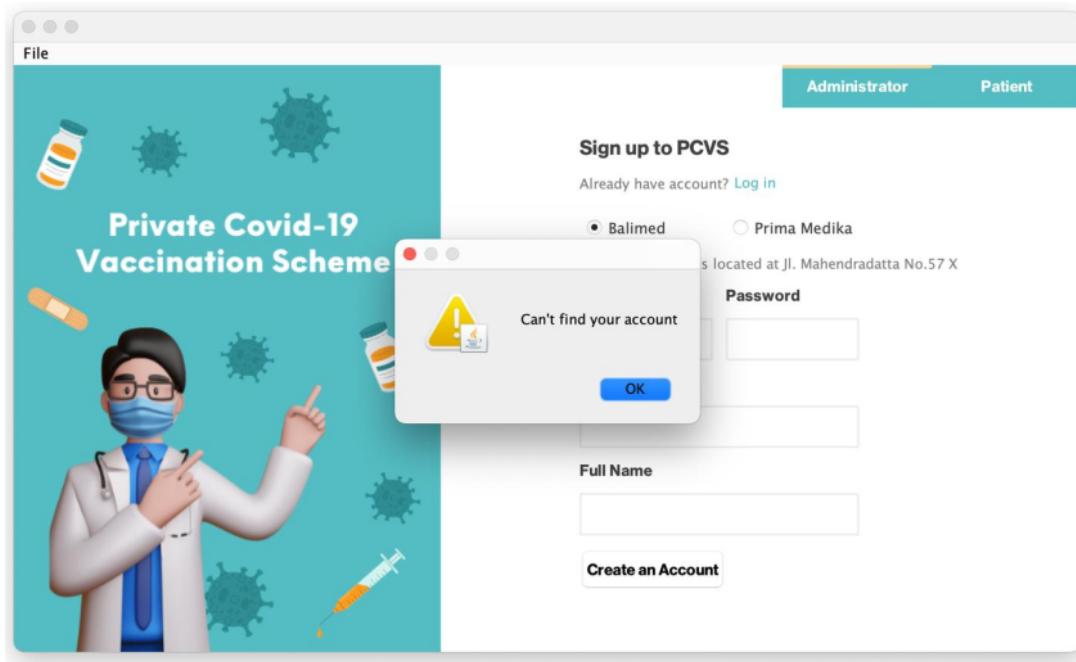
Use Case 4 View Vaccine Batch Information

Administrator login with valid account



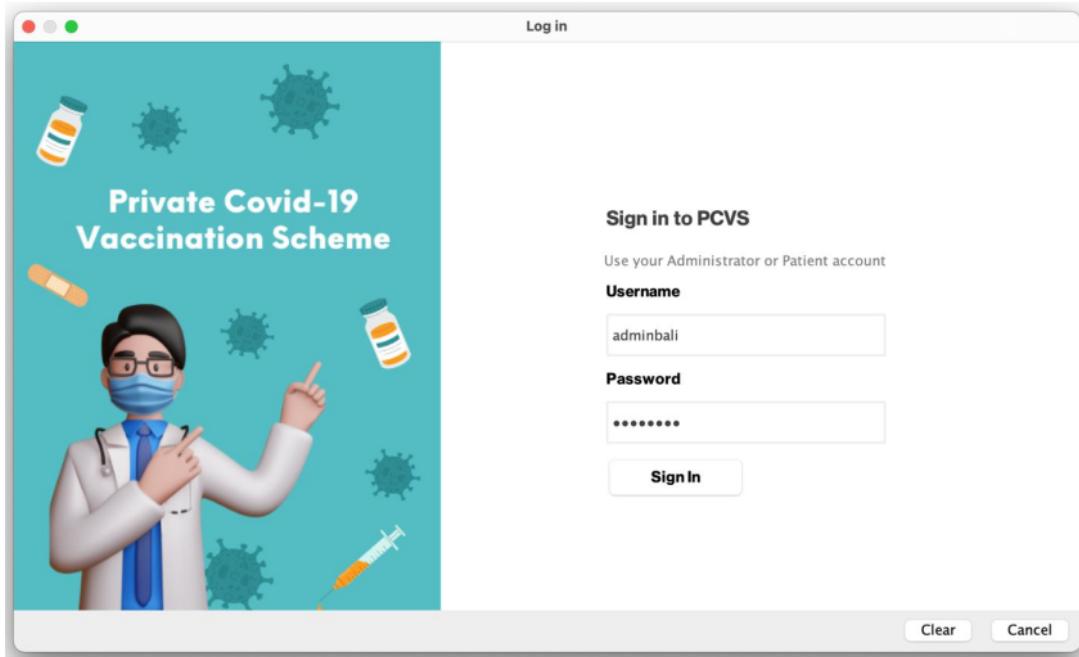
Picture 1: View Vaccine Batch Information

Login Validation



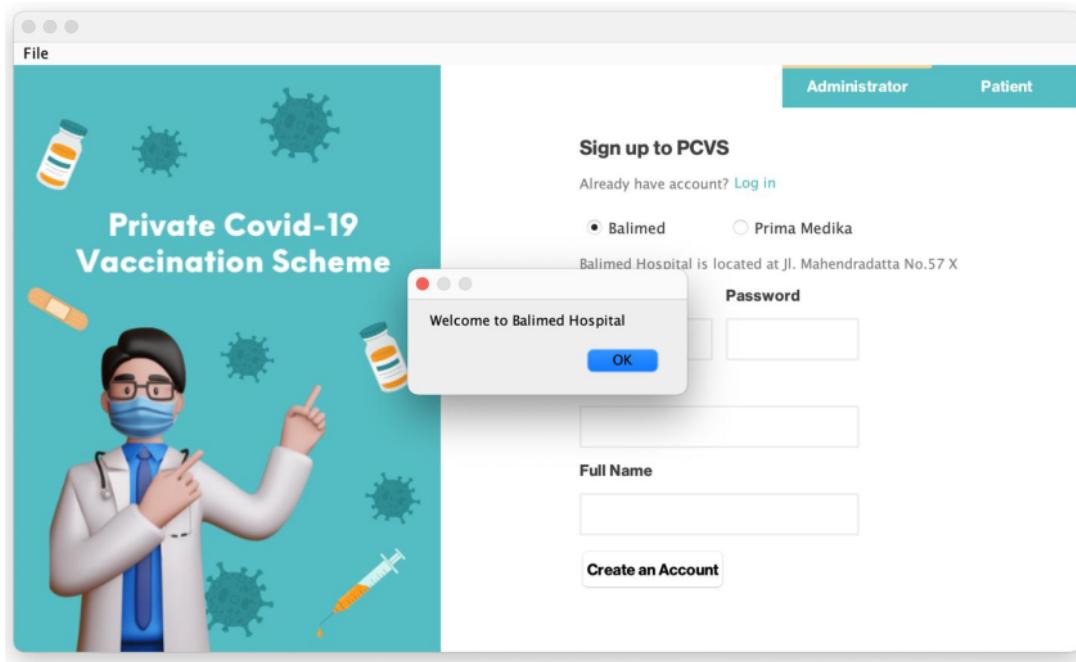
Picture 1.1: View Vaccine Batch Information

Login Validation



Picture 2: View Vaccine Batch Information

Login Validation



Picture 2.1: View Vaccine Batch Information

Login Validation

The screenshot shows a computer application window titled "Record Vaccine Batch". The window has a teal sidebar on the left containing a vertical menu with options: "Record Vaccine Batch" (which is selected and highlighted in orange), "View Vaccine Batch", "Administrator Information", "Patient Information", and "Vaccination Information". At the bottom of the sidebar is a "Log Out" button. The main content area has a pink header bar with the title "Record Vaccine Batch". Below the header, there is a section titled "Select Vaccine ID" with two radio buttons: "JNJ" (selected) and "ASZ". A note below the radio buttons states "Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies". There are three input fields: "Batch Number", "Expiry Date (mm dd yyyy)", and "Quantity Available". At the bottom right of the main content area is a "Record" button.

Picture 3: 1 View Vaccine Batch Information

Administrator Menu

Administrator select view vaccine batch menu

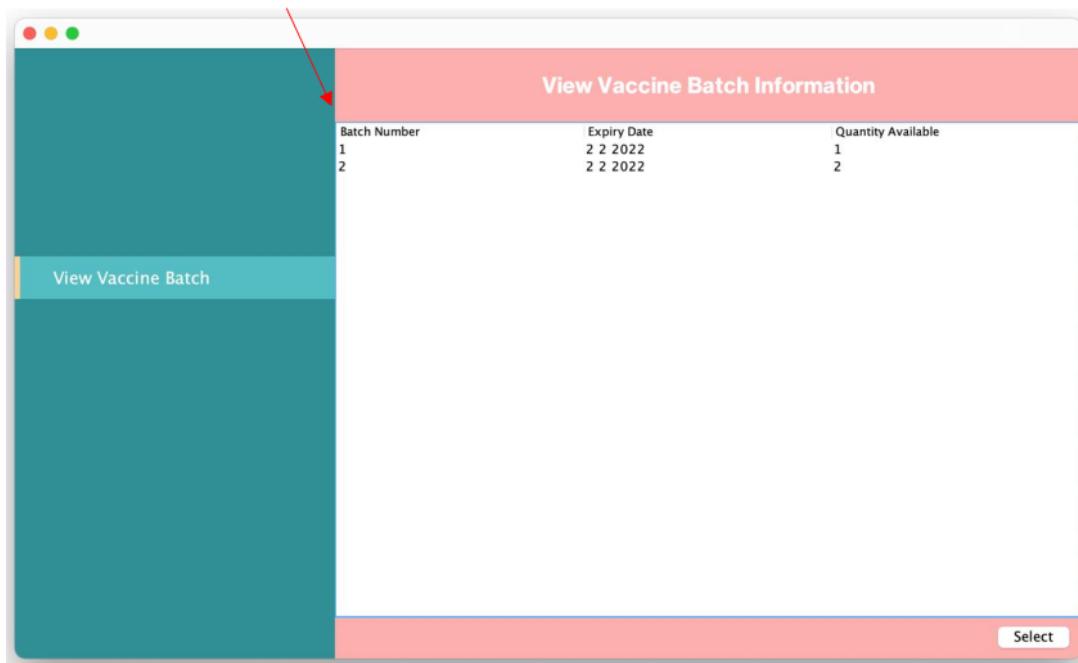
The screenshot shows a software application window titled "Record Vaccine Batch". On the left, there is a vertical sidebar menu with the following options: "Record Vaccine Batch" (selected), "View Vaccine Batch", "Administrator Information", "Patient Information", and "Vaccination Information". At the bottom of the sidebar is a "Log Out" button. The main content area has a pink header bar. Below it, the title "Record Vaccine Batch" is displayed. Underneath, there is a section titled "Select Vaccine ID" with two radio buttons: "JNJ" (selected) and "ASZ". A note below states: "Johnson & Johnson vaccine, developed by Janssen Pharmaceutical Companies". The main form contains fields for "Batch Number" (an input field), "Expiry Date (mm dd yyyy)" (an input field), and "Quantity Available" (an input field). At the bottom right of the form is a "Record" button.

1
Picture 4: View Vaccine Batch Information

Administrator Menu

List of available vaccine batches by vaccine name and number pending

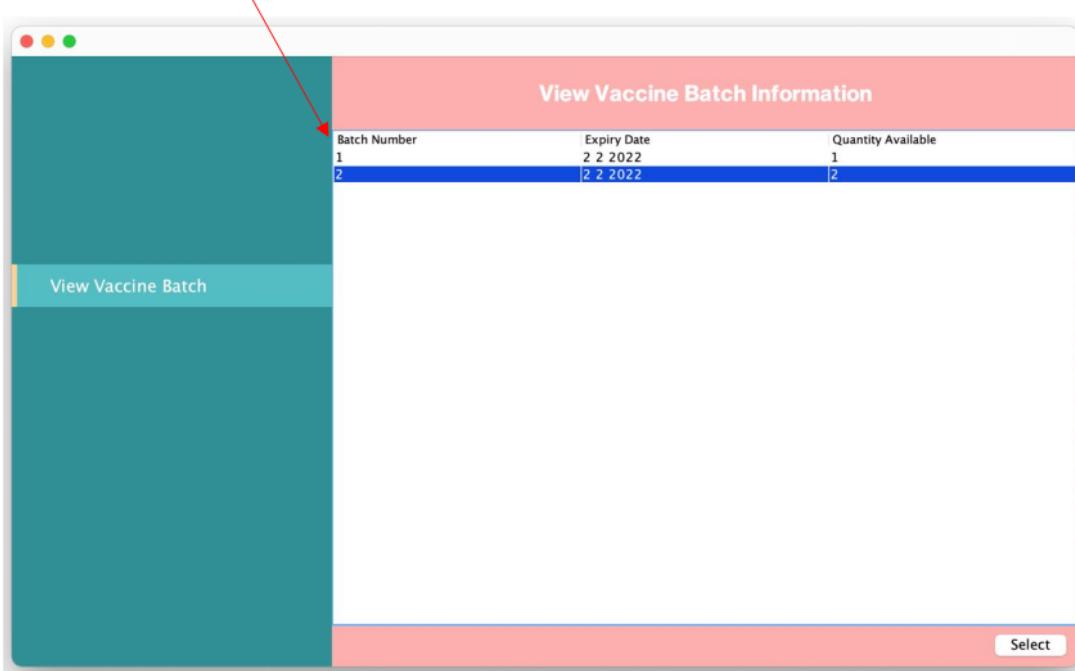
appointment is shown



Picture 5: ¹ View Vaccine Batch Information

Available Vaccine Batch Information is Shown

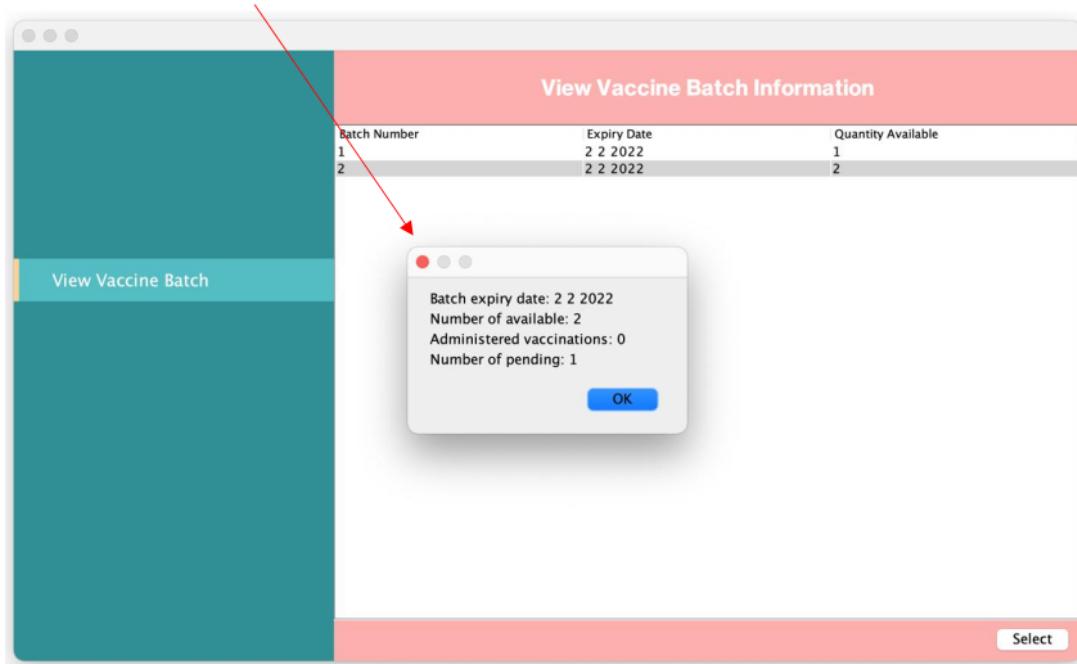
Patient select batch



Picture 5.1: View Vaccine Batch Information

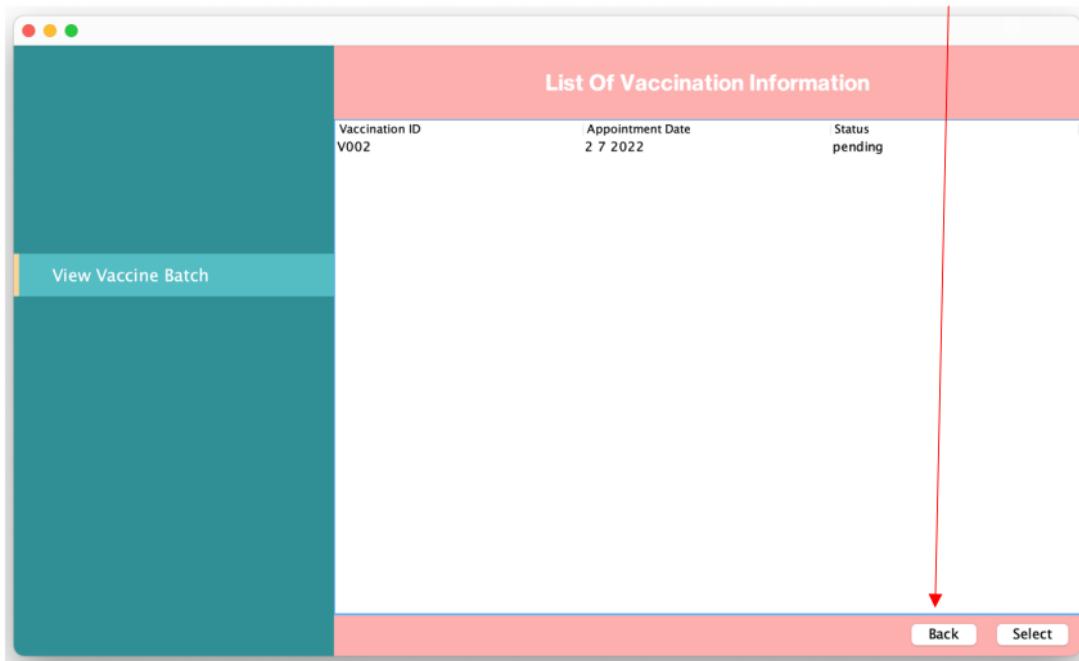
Available Vaccine Batch Information is Shown

Batch and Vaccination information is shown

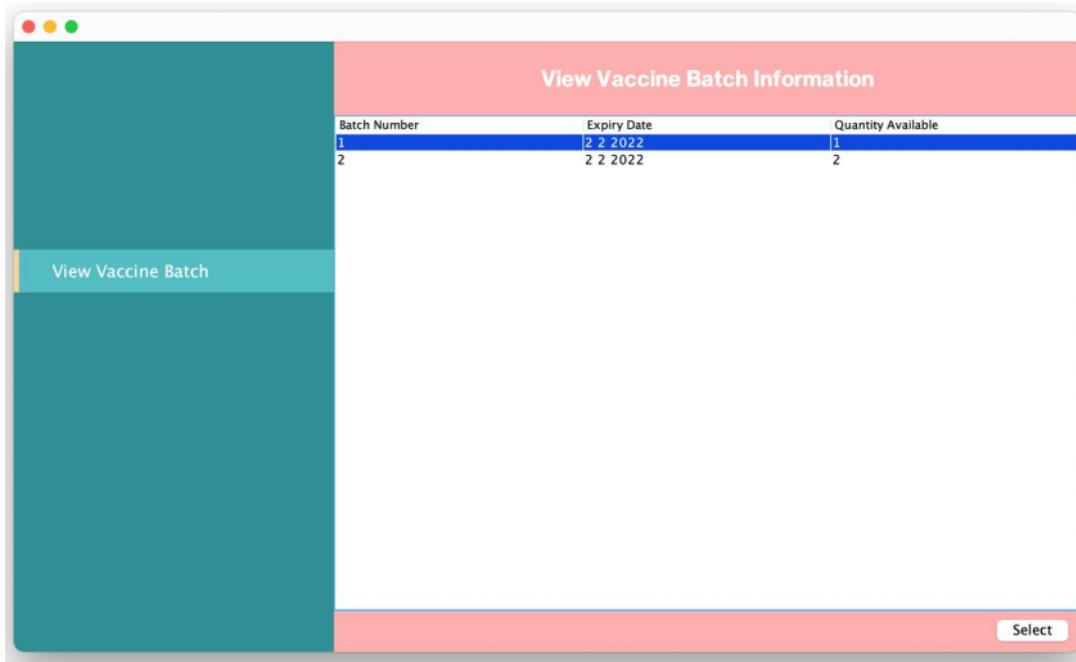


Picture 6: View Vaccine Batch Information

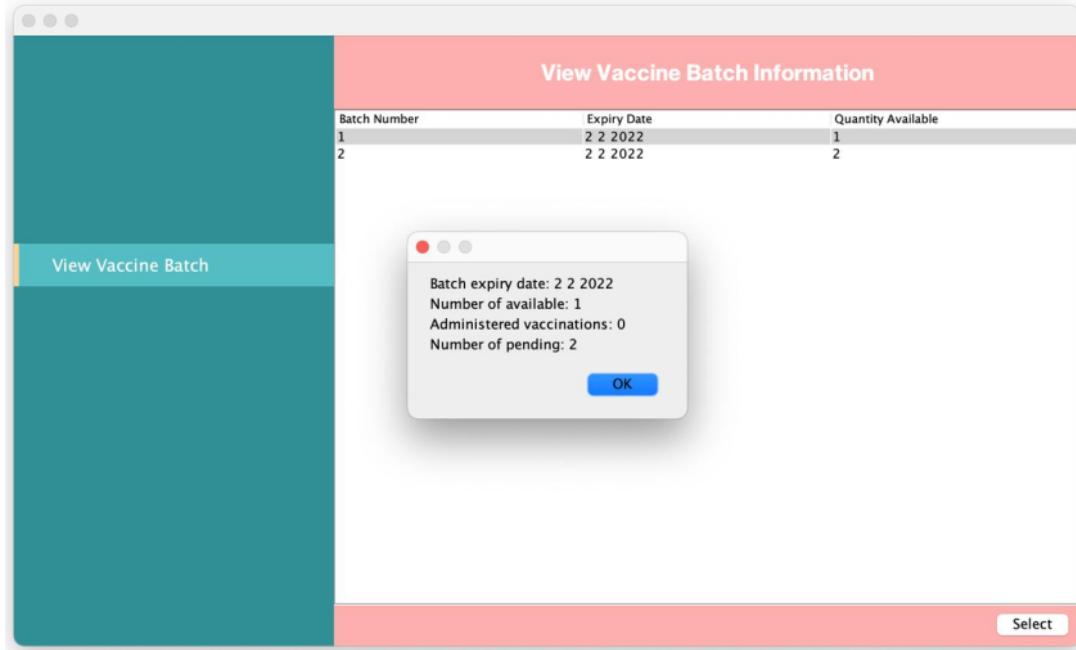
Choose different batch



Picture 7: ¹ **View Vaccine Batch Information**

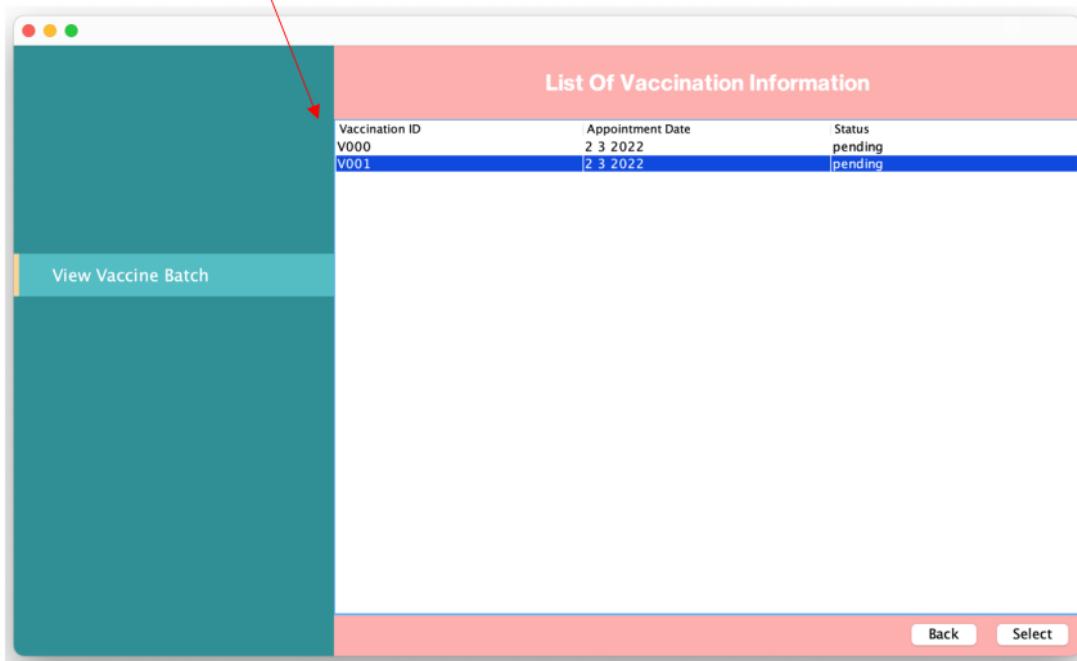


Picture 8: View Vaccine Batch Information



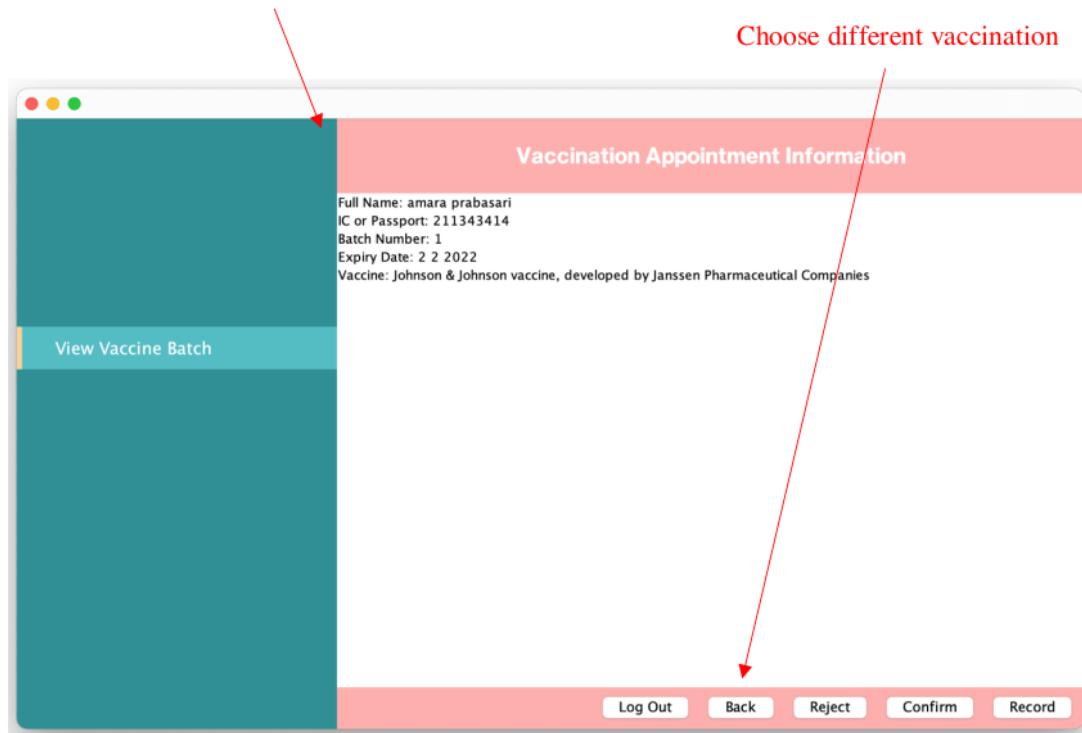
Picture 9: View Vaccine Batch Information

Select vaccination



Picture 10: View Vaccine Batch Information

The information about the patient, batch, and vaccine is shown



Picture 11: View Vaccine Batch Information

A screenshot of a mobile application interface titled "List Of Vaccination Information". The screen is divided into three main sections: a teal sidebar on the left, a white central content area, and a light red footer bar at the bottom.

The central content area displays a table with two rows of vaccination information:

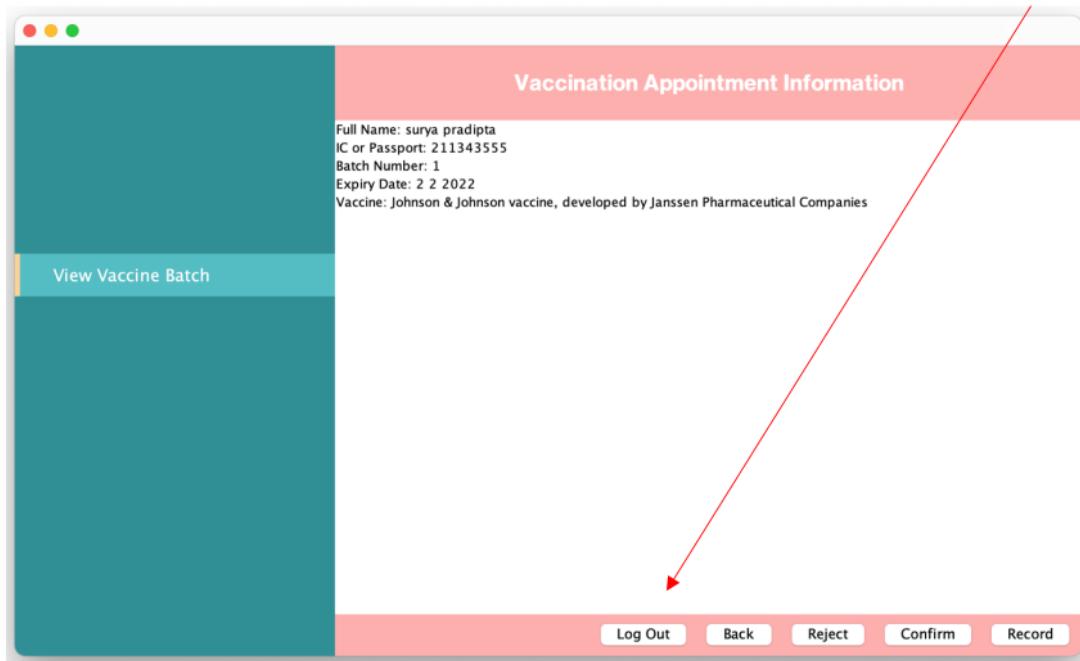
Vaccination ID	Appointment Date	Status
V000	23 2022	pending
V001	23 2022	pending

At the bottom of the central content area, there is a teal button labeled "View Vaccine Batch".

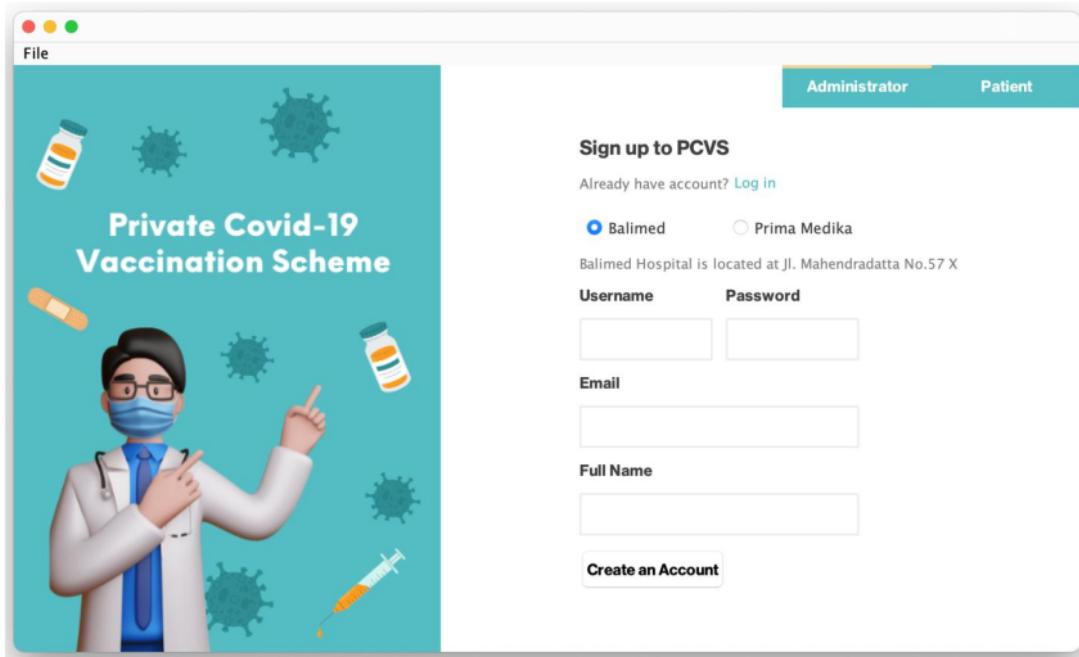
The footer bar is light red and contains two buttons: "Back" and "Select".

Picture 12: View Vaccine Batch Information

Select log out



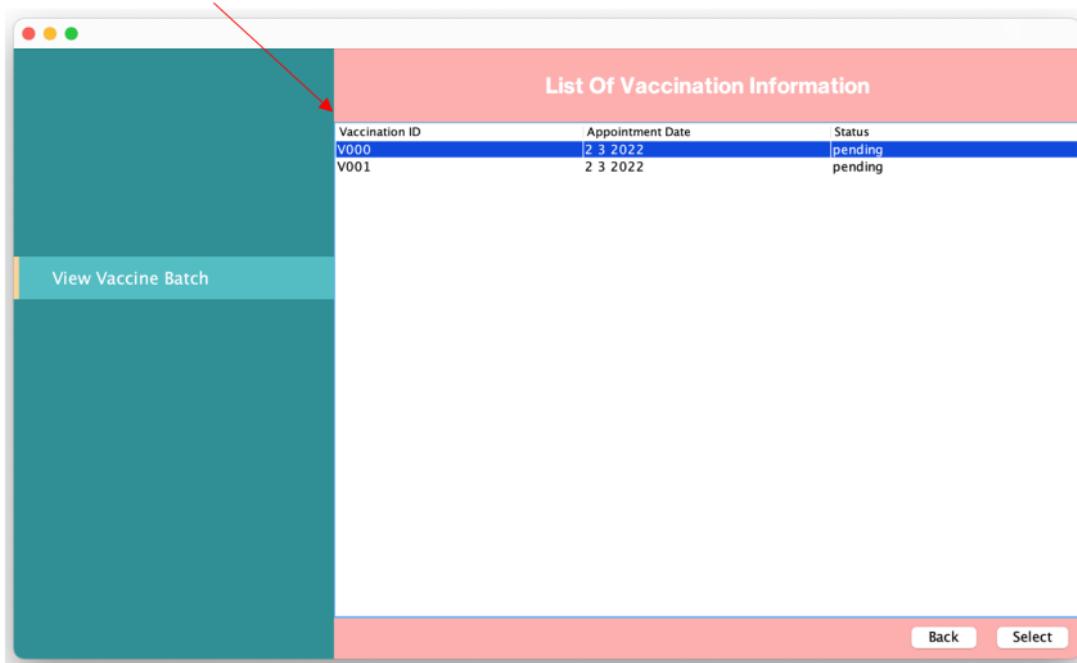
Picture 13: View Vaccine Batch Information



Picture 14: View Vaccine Batch Information

Use Case 5 Confirm Vaccination Appointment

Select vaccination ID

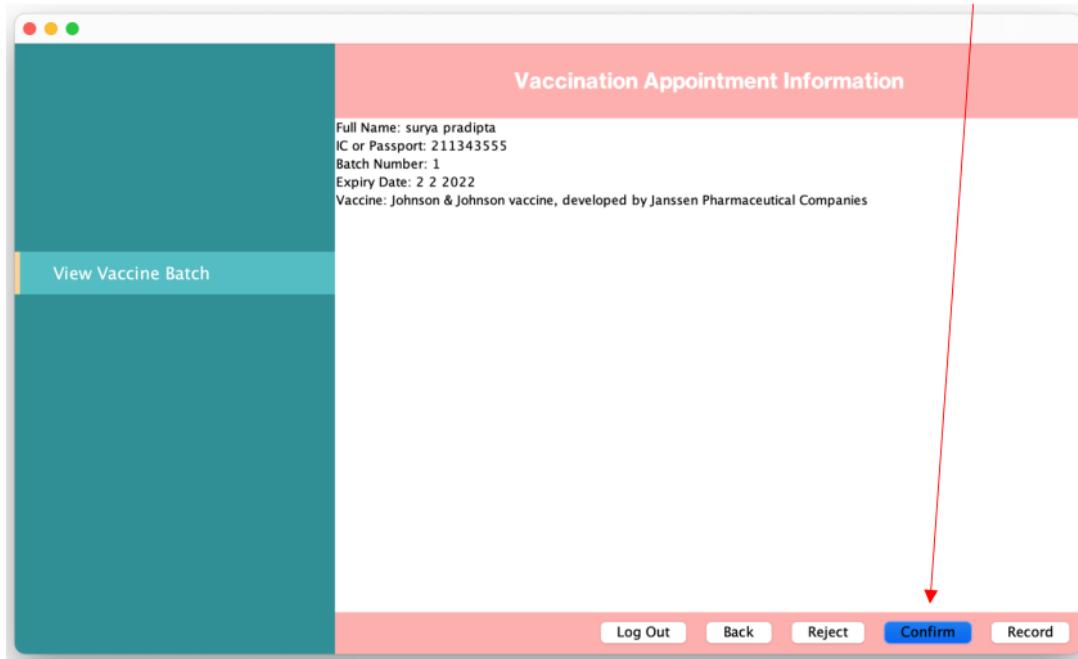


Picture 1: Confirm Vaccination Appointment

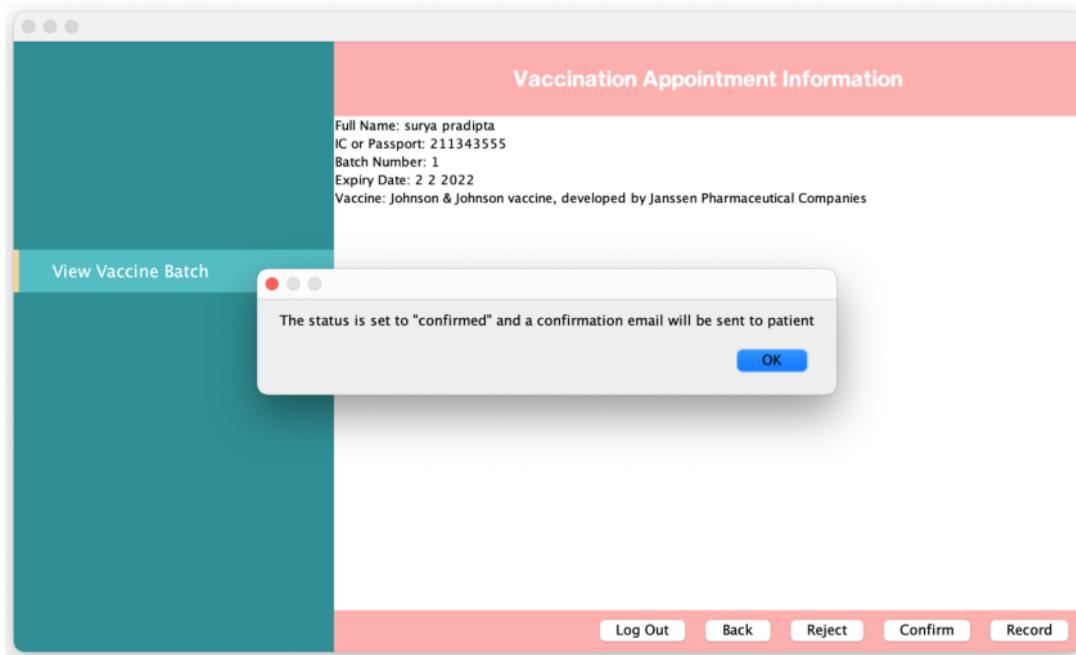


Picture 2: Confirm Vaccination Appointment

Select confirm

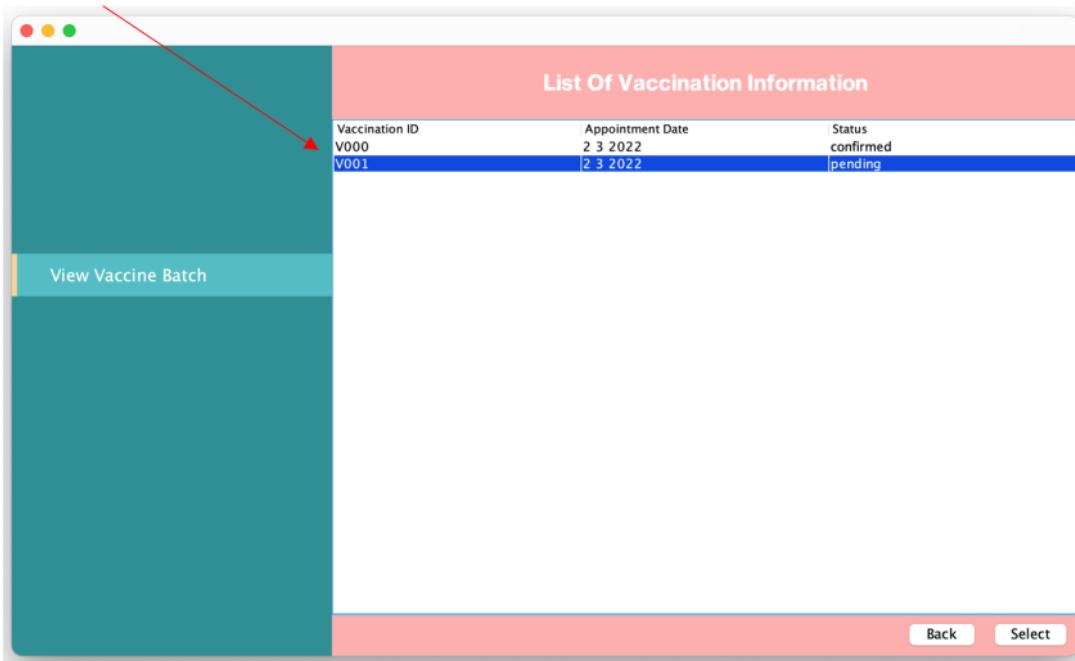


Picture 3: Confirm Vaccination Appointment



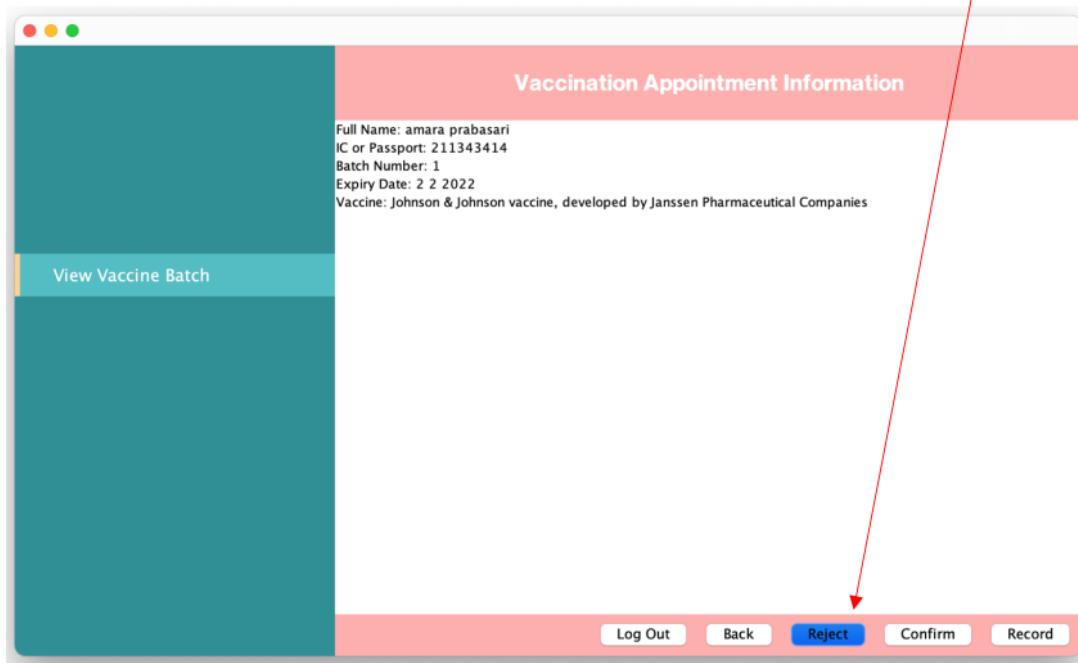
Picture 4: Confirm Vaccination Appointment

Select another vaccination

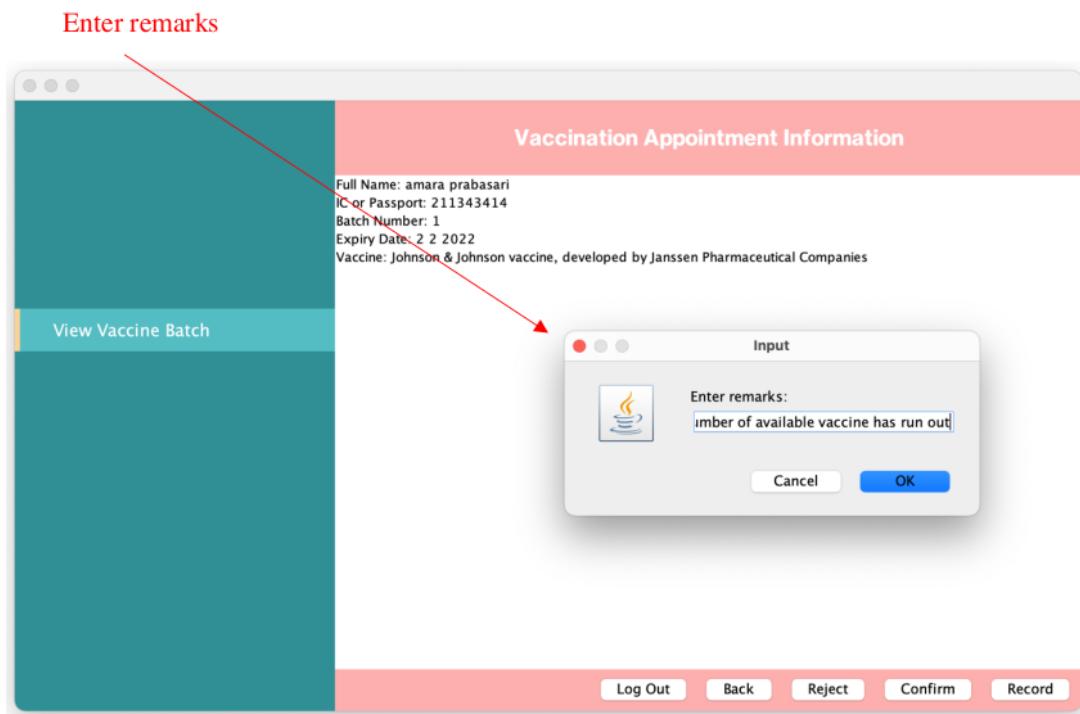


Picture 5: Confirm Vaccination Appointment

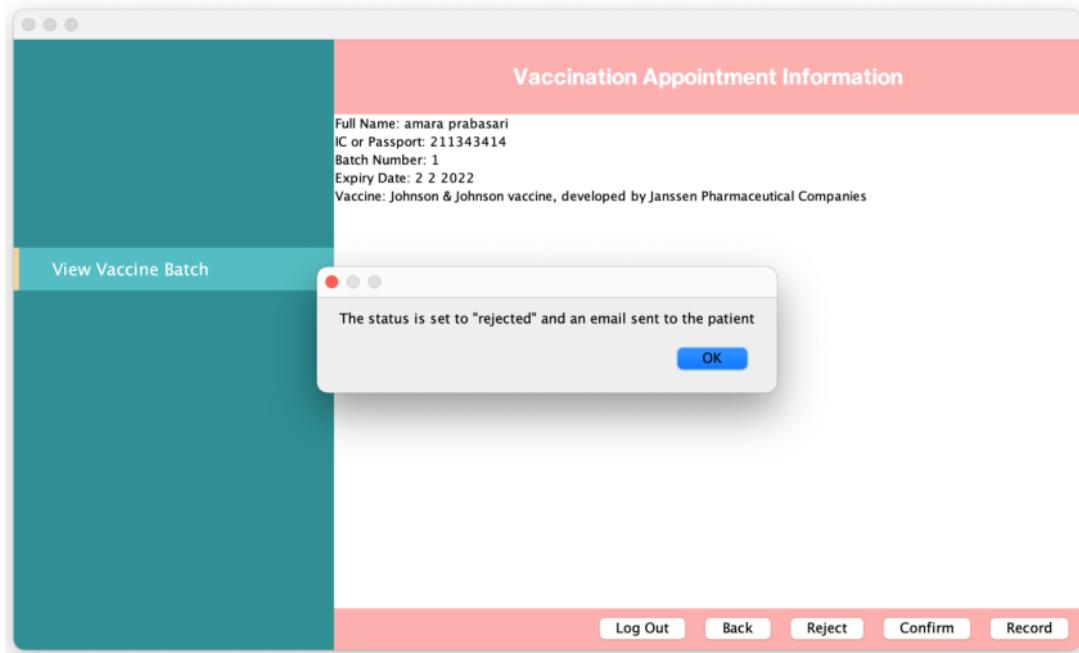
Select reject



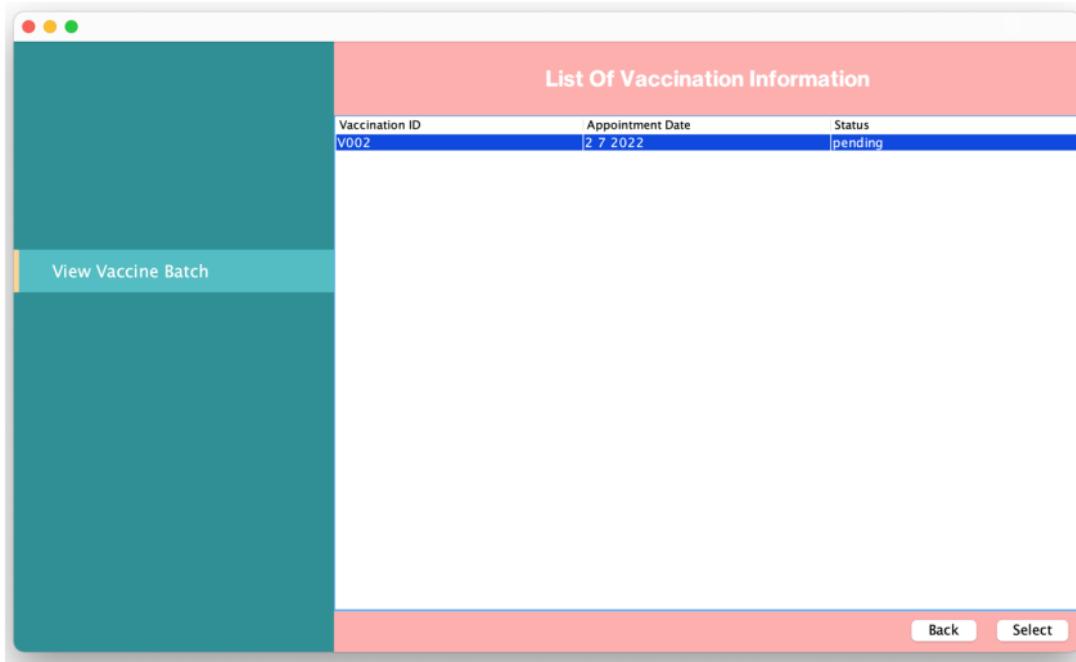
Picture 6: Confirm Vaccination Appointment



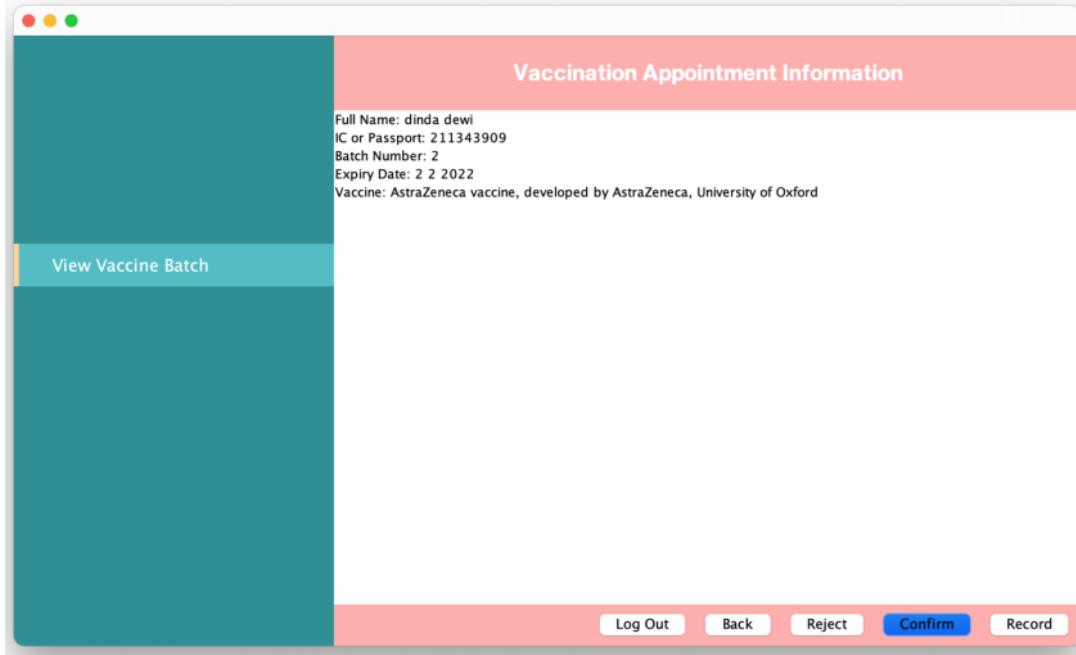
Picture 7: Confirm Vaccination Appointment



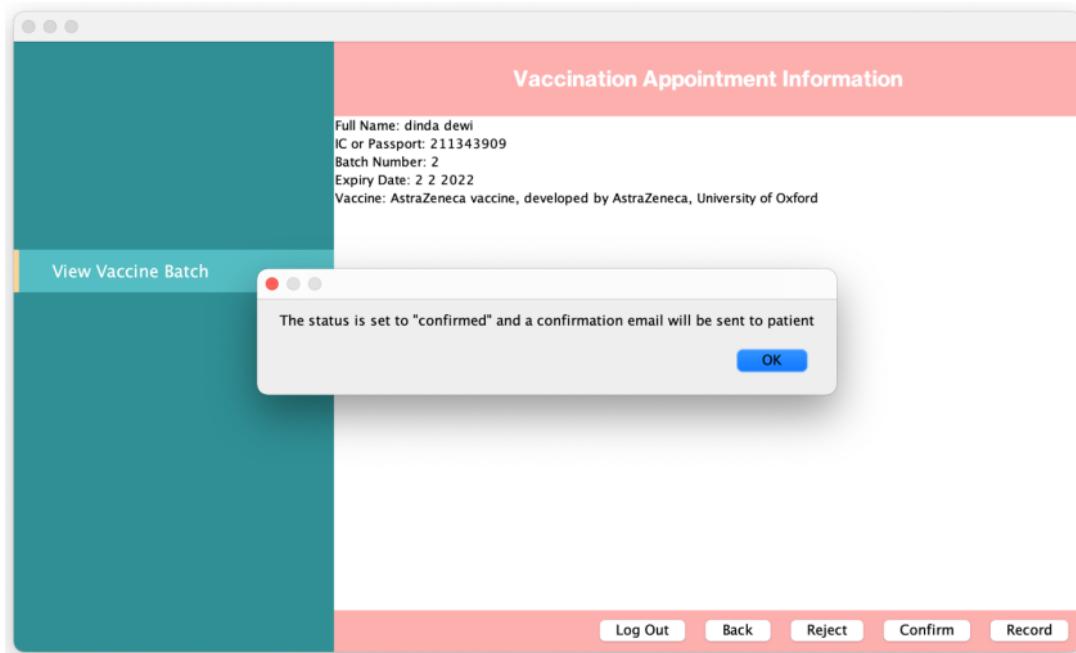
Picture 8: Confirm Vaccination Appointment



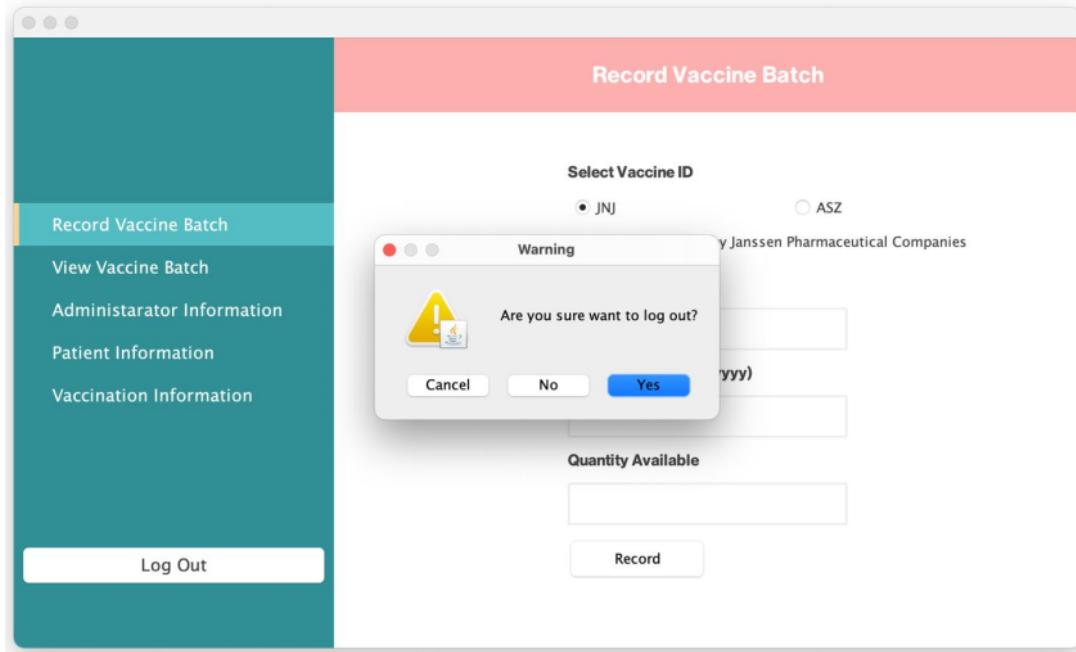
Picture 8: Confirm Vaccination Appointment



Picture 9: Confirm Vaccination Appointment

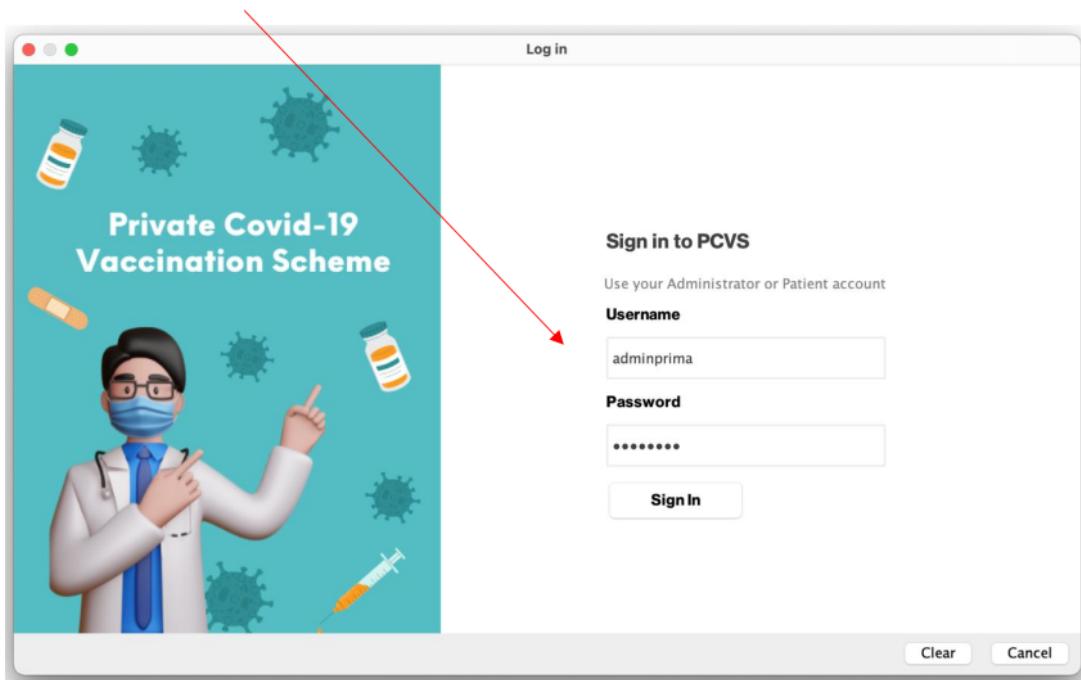


Picture 10: Confirm Vaccination Appointment

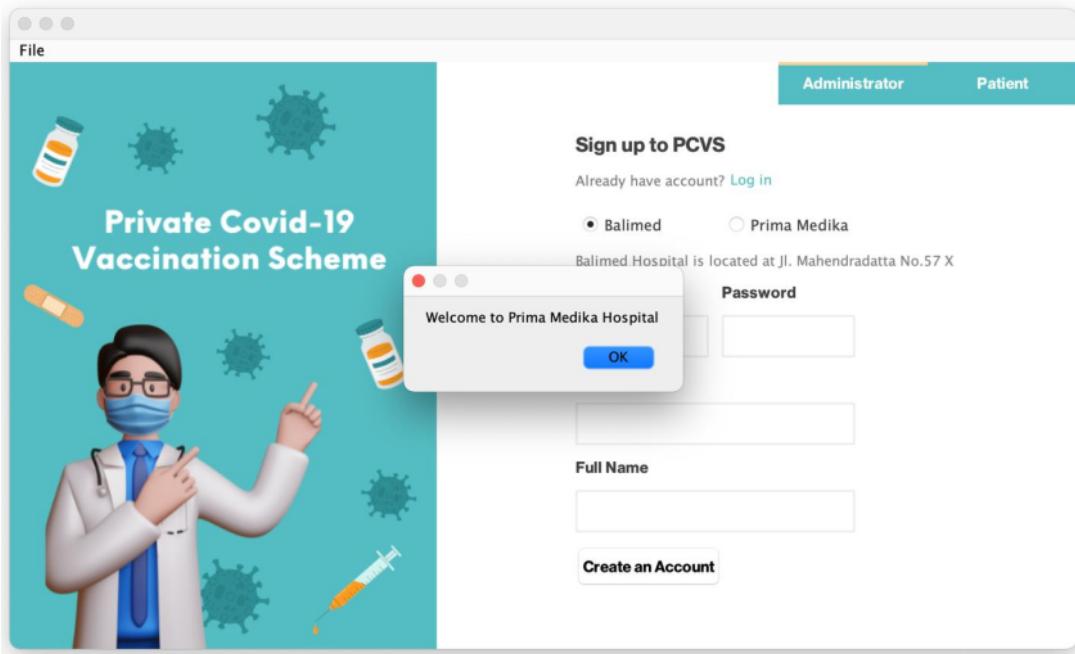


Picture 11: Confirm Vaccination Appointment

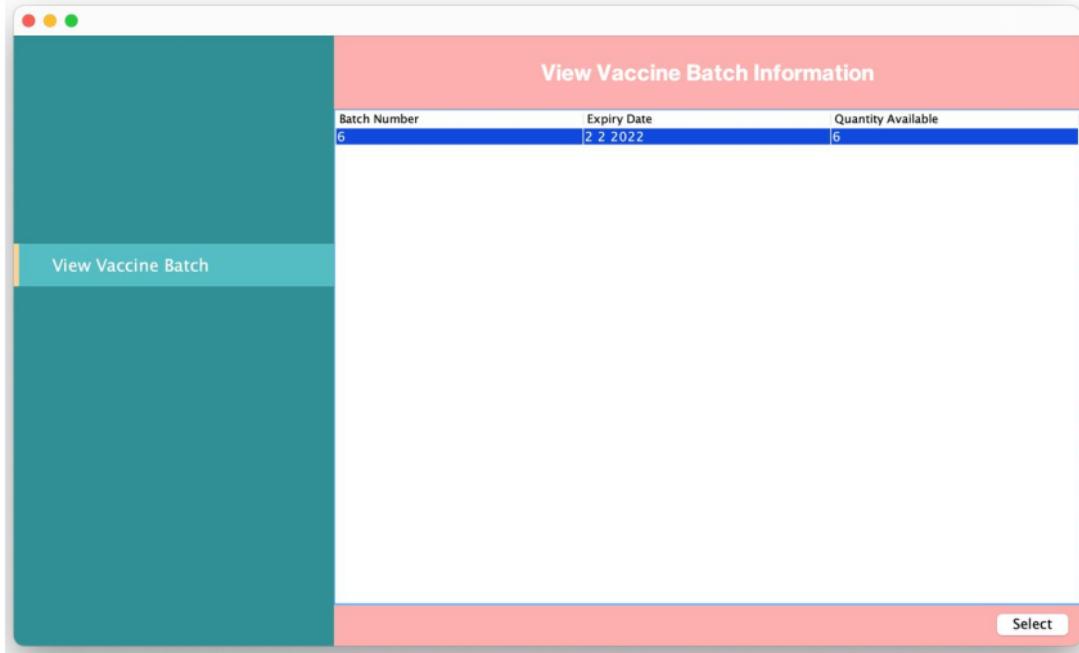
Confirm vaccination at another healthcare centre



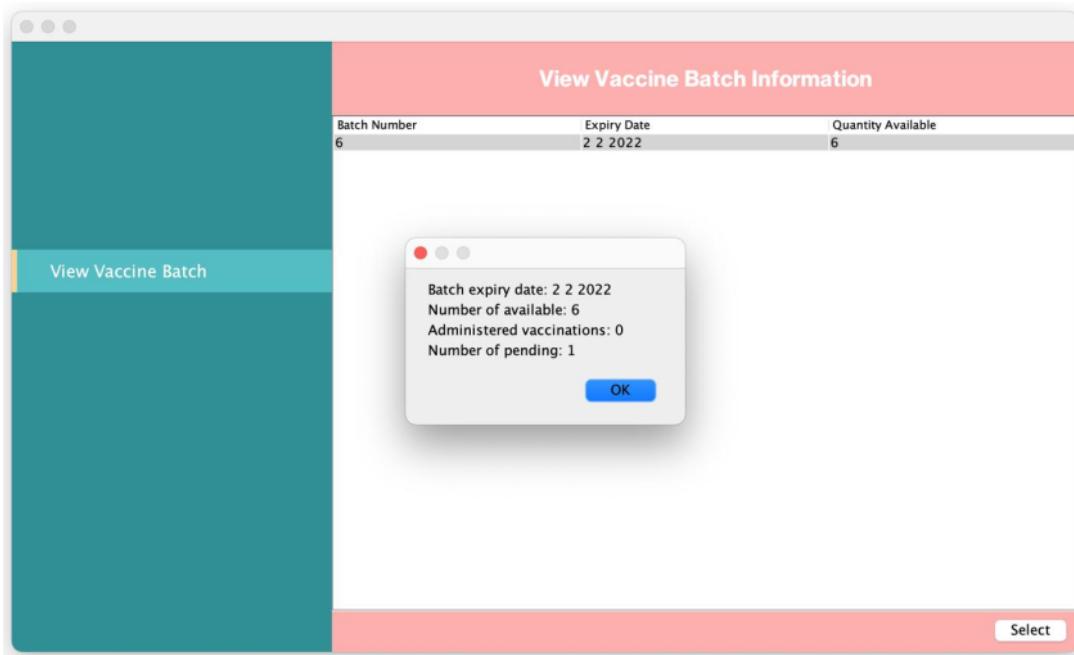
Picture 12: Confirm Vaccination Appointment



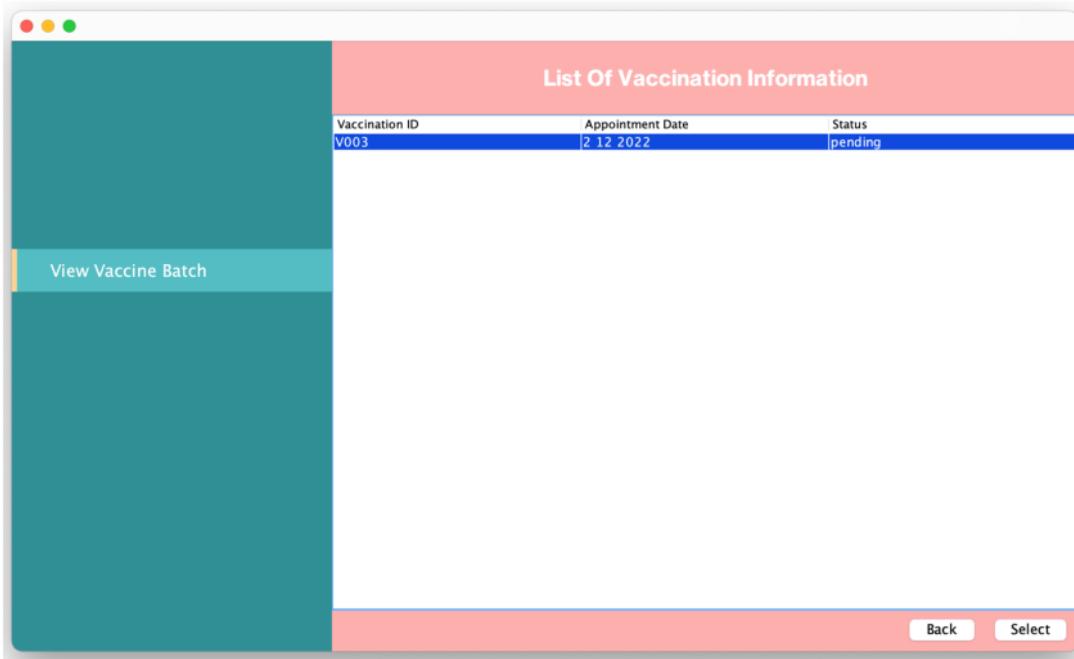
Picture 13: Confirm Vaccination Appointment



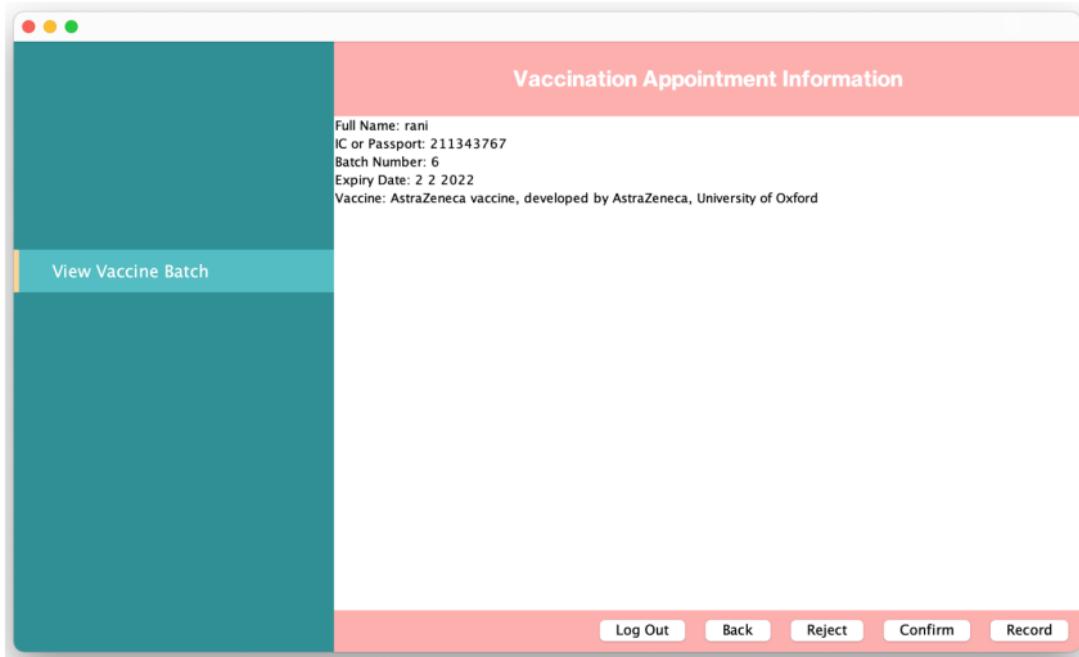
Picture 14: Confirm Vaccination Appointment



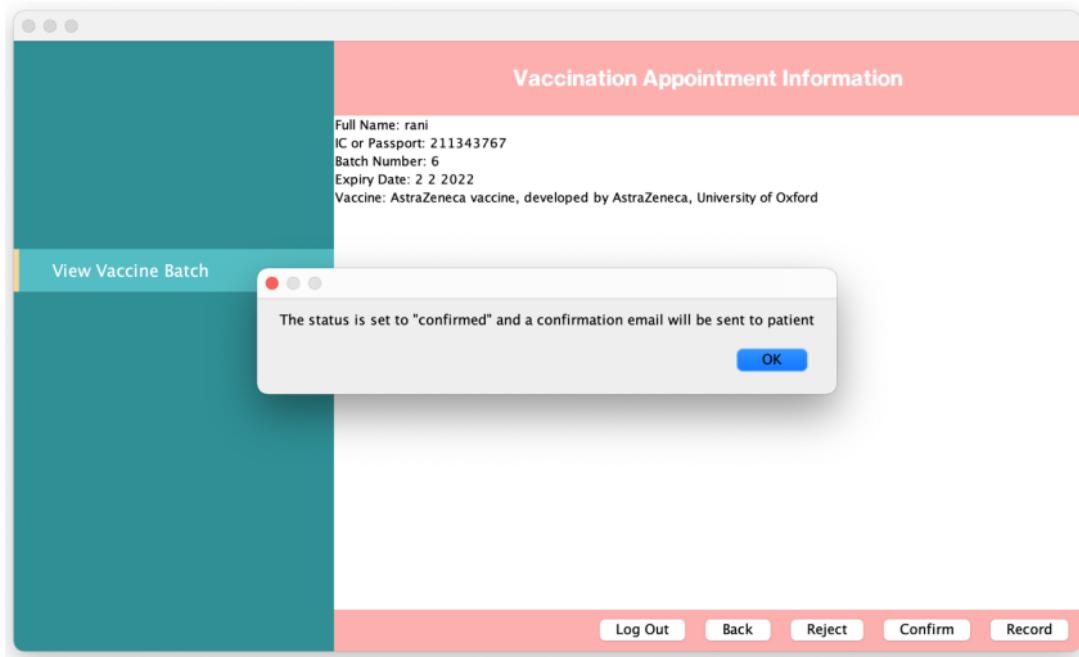
Picture 15: Confirm Vaccination Appointment



Picture 16: Confirm Vaccination Appointment



Picture 17: Confirm Vaccination Appointment



Picture 18: Confirm Vaccination Appointment

A screenshot of a mobile application interface titled "Vaccination Information". The screen is divided into three main sections: a teal sidebar on the left, a pink header bar at the top, and a white content area on the right.

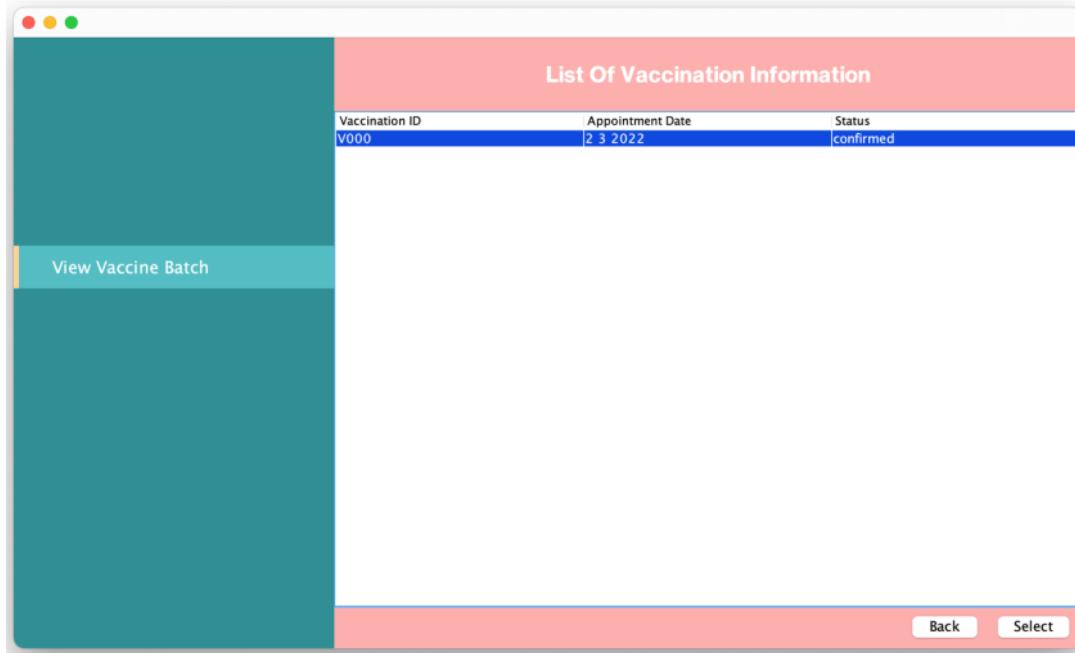
The header bar contains the title "Vaccination Information". The content area displays a table with four columns: "Vaccination ID", "Appointment Date", "Status", and "Remarks". The data in the table is as follows:

Vaccination ID	Appointment Date	Status	Remarks
V000	2 3 2022	confirmed	unknown
V001	2 3 2022	rejected	The number of available va...
V002	2 7 2022	confirmed	unknown
V003	2 12 2022	confirmed	unknown

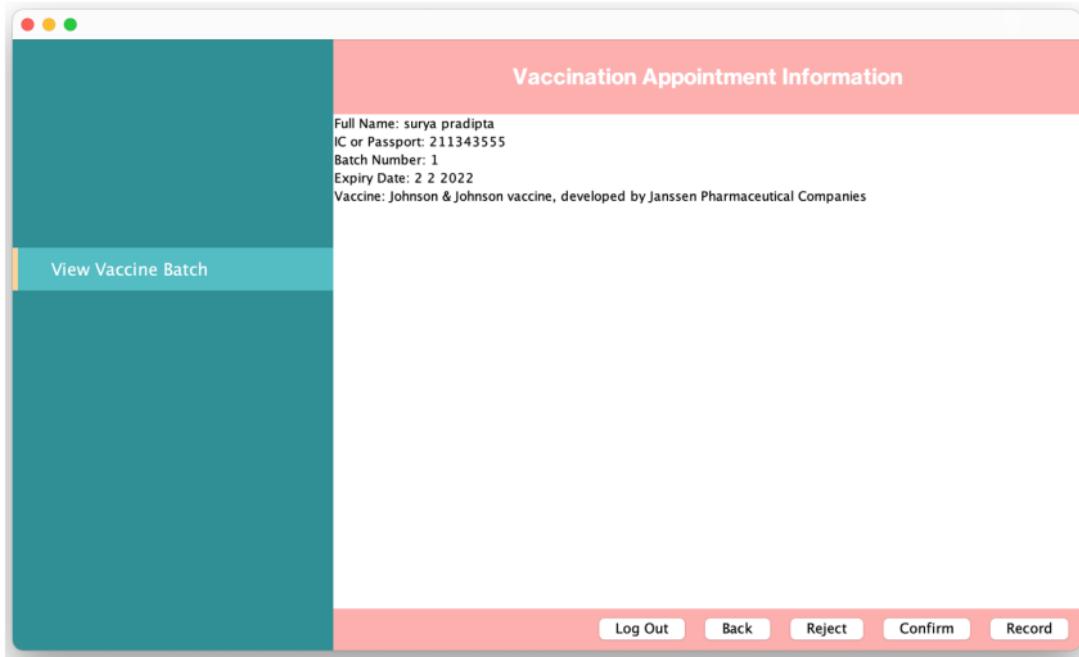
At the bottom of the content area, there is a small teal bar with the text "Vaccination Information". In the bottom right corner of the main content area, there is a "Back" button.

Picture 19: Confirm Vaccination Appointment

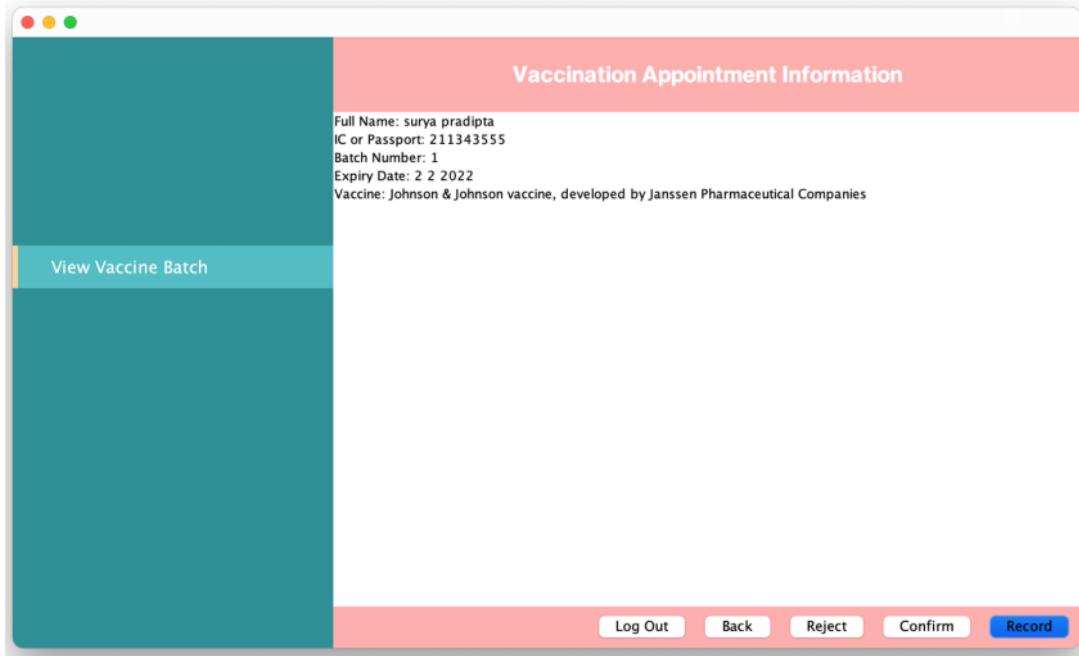
Use Case 6 Record Vaccination Administered



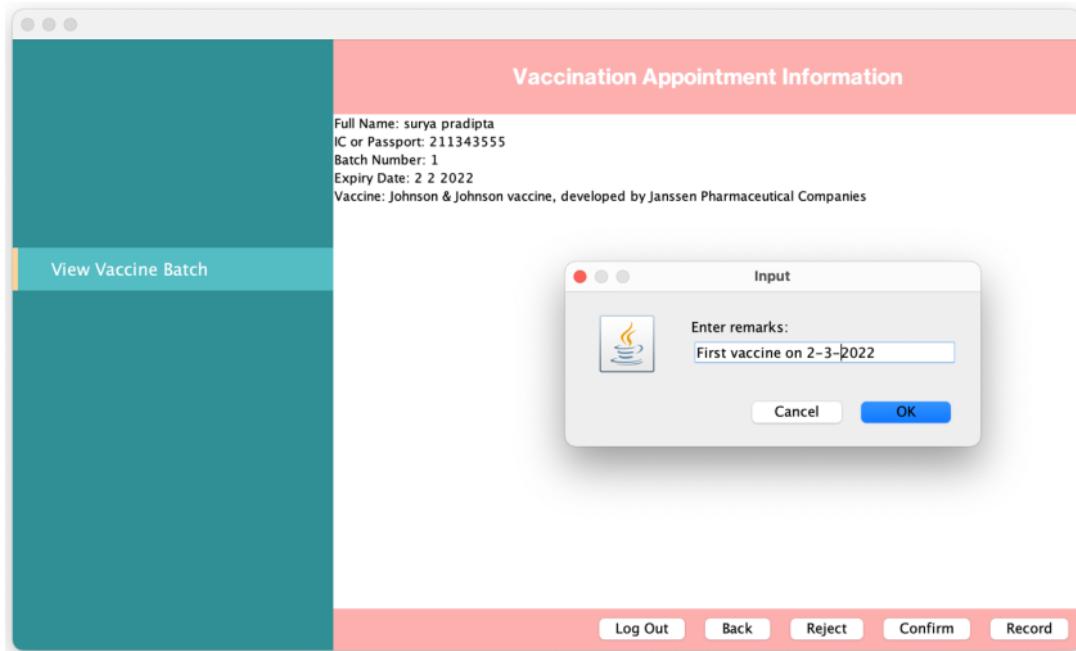
Picture 1: Record Vaccination Administered



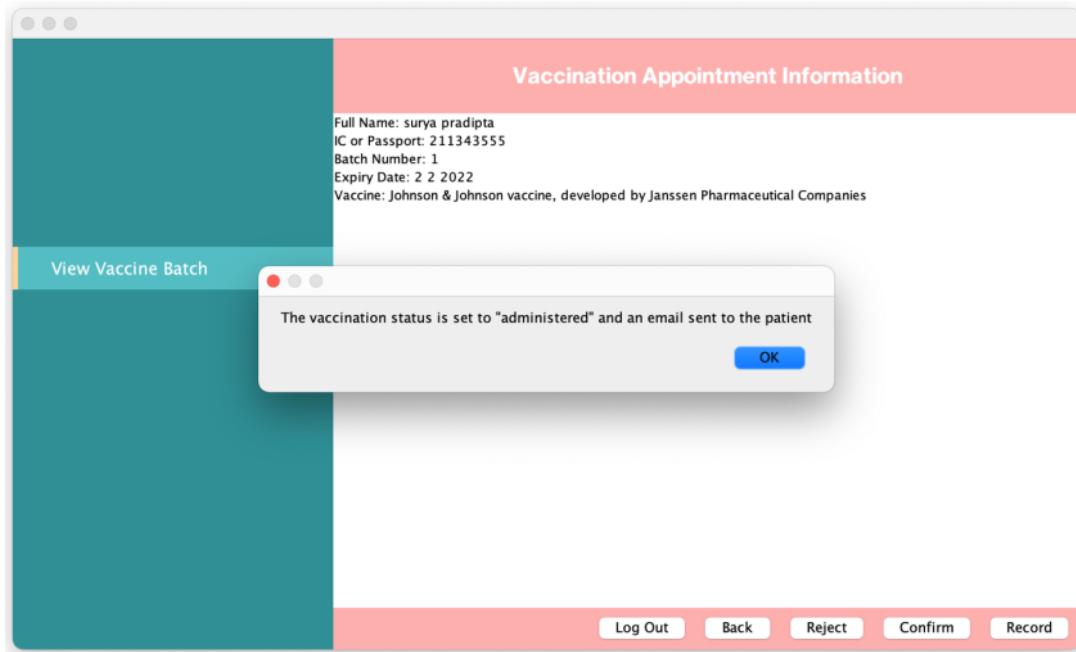
Picture 2: Record Vaccination Administered



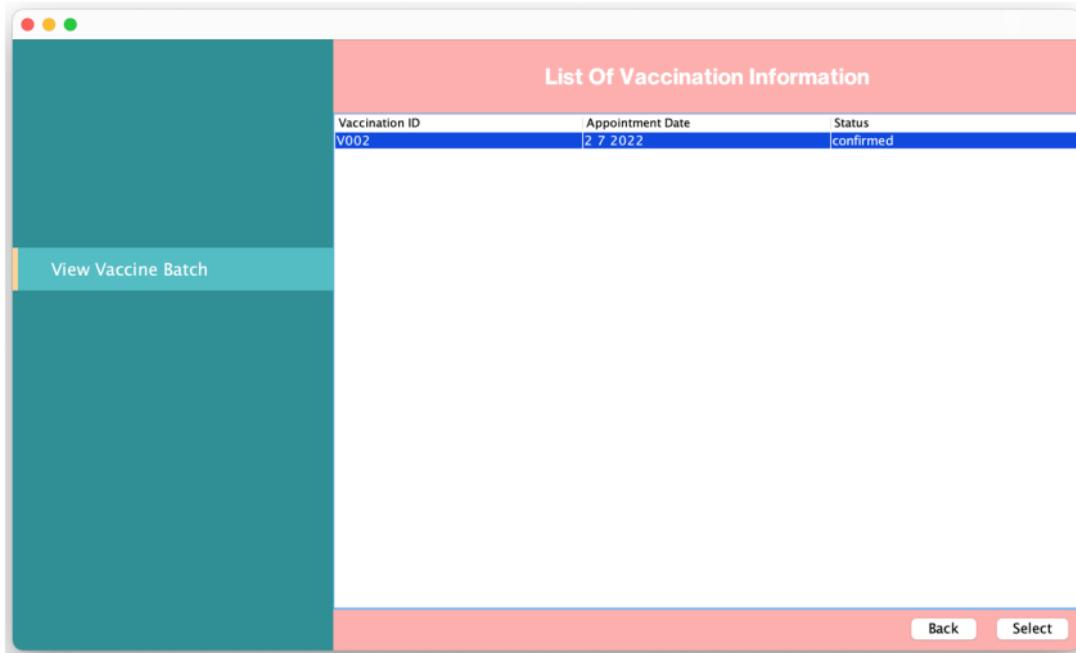
Picture 3: Record Vaccination Administered



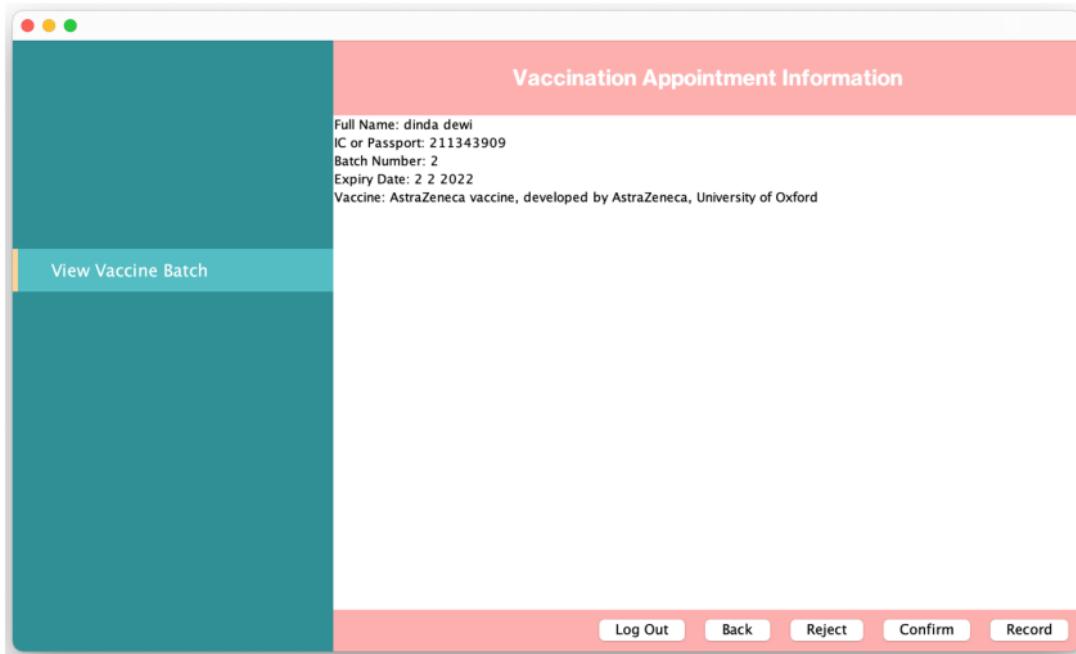
Picture 4: Record Vaccination Administered



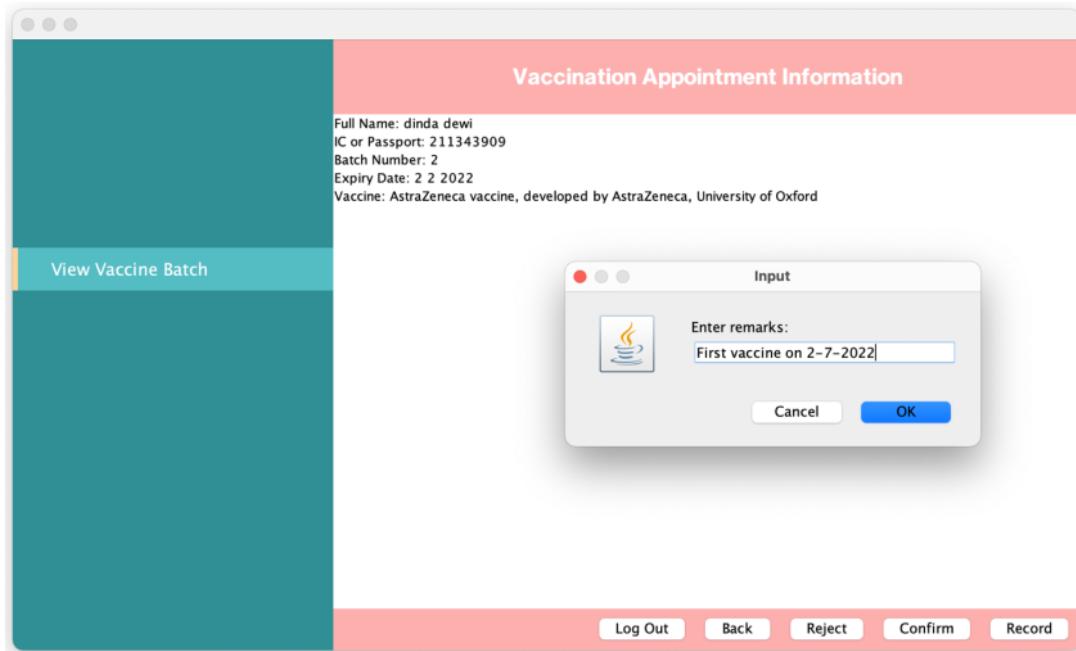
Picture 5: Record Vaccination Administered



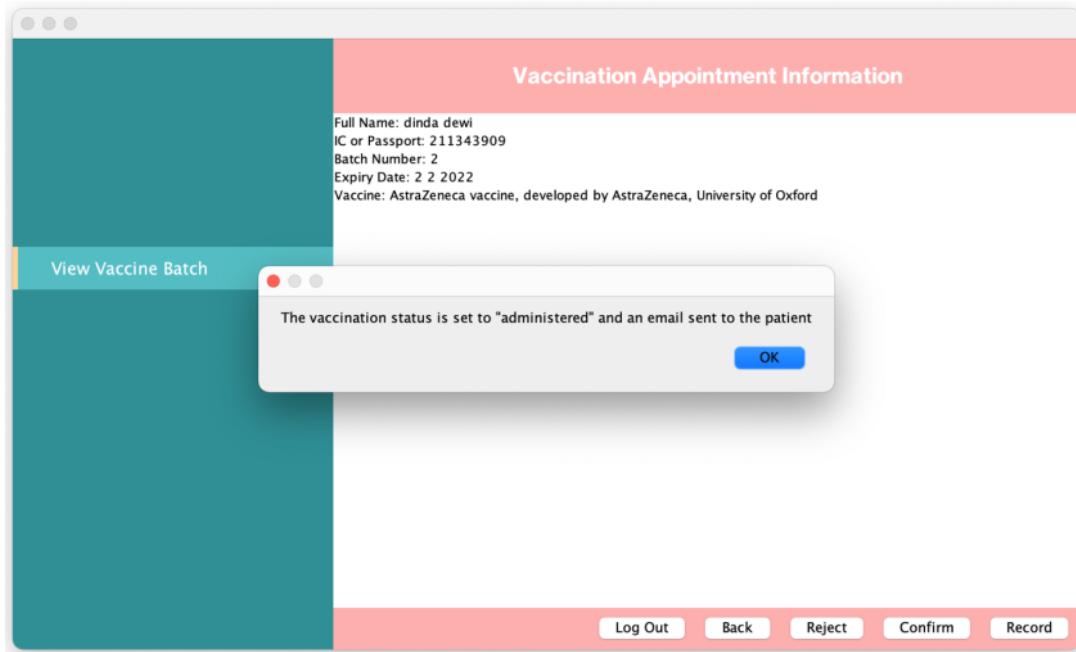
Picture 6: Record Vaccination Administered



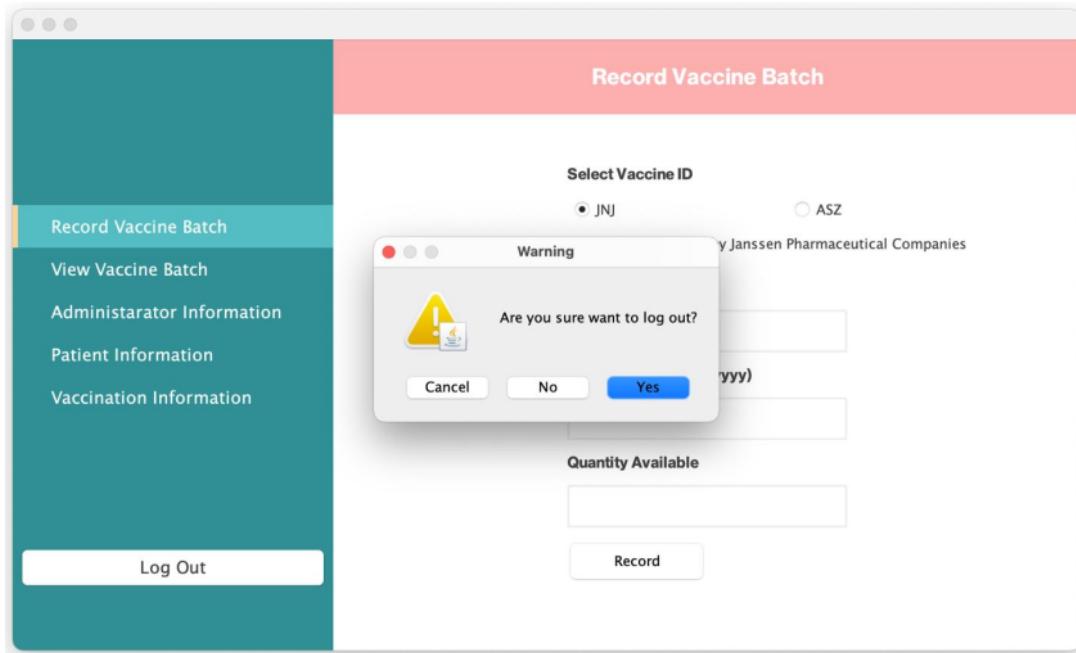
Picture 7: Record Vaccination Administered



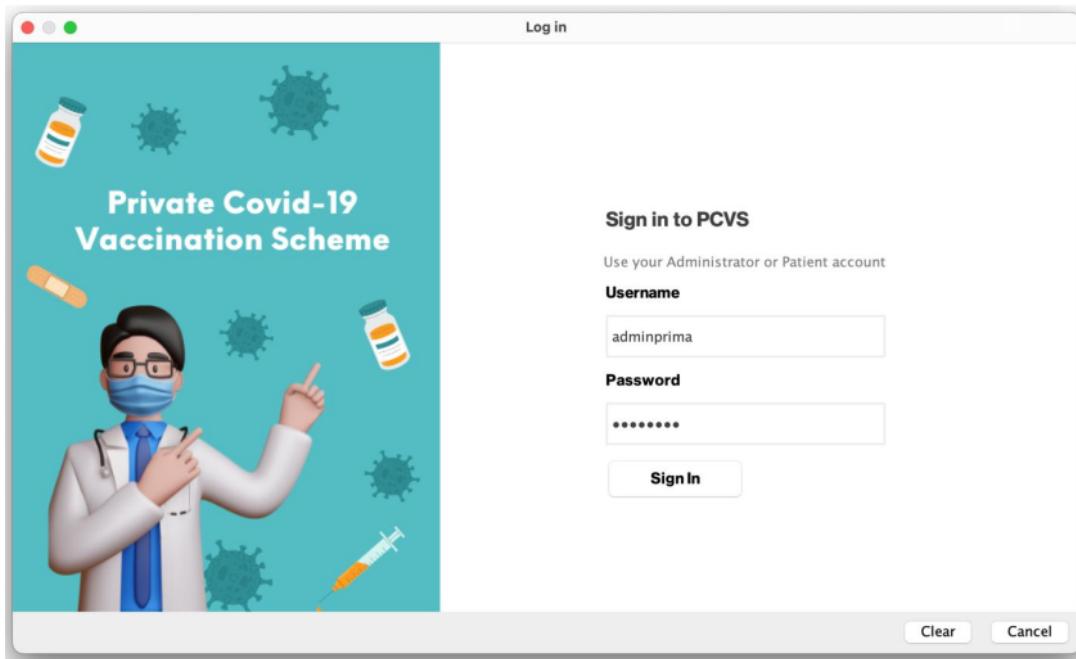
Picture 8: Record Vaccination Administered



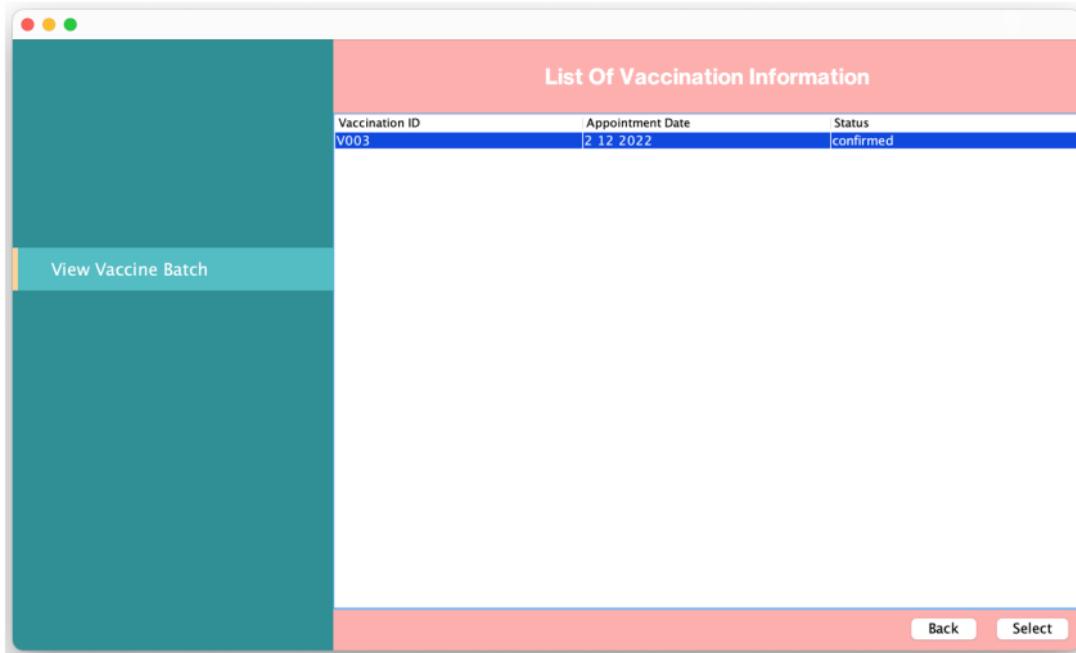
Picture 9: Record Vaccination Administered



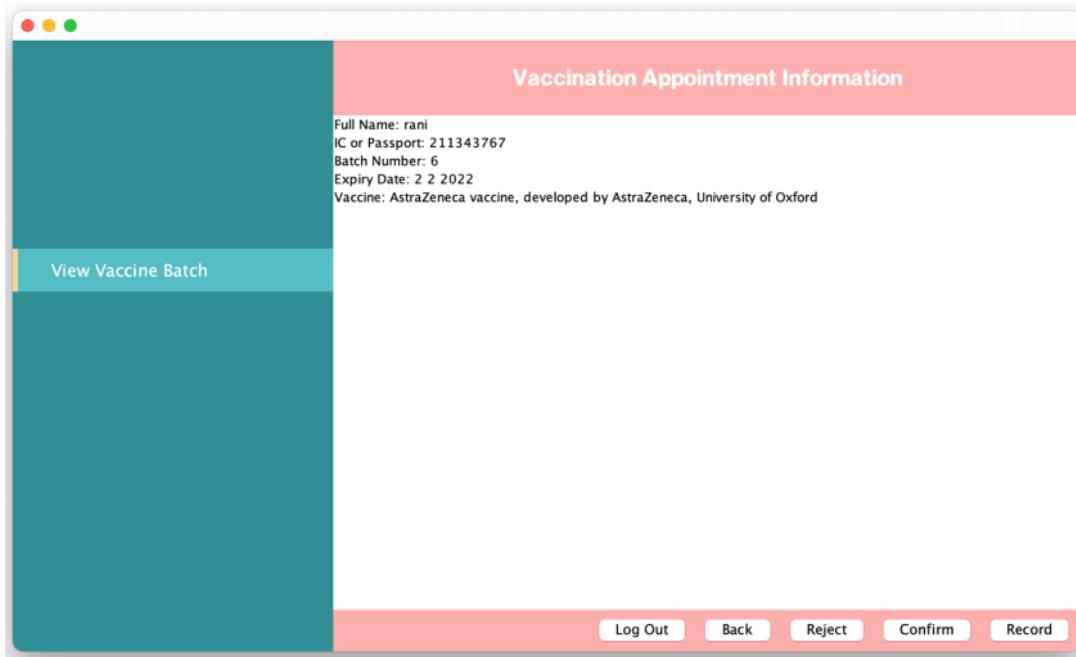
Picture 10: Record Vaccination Administered



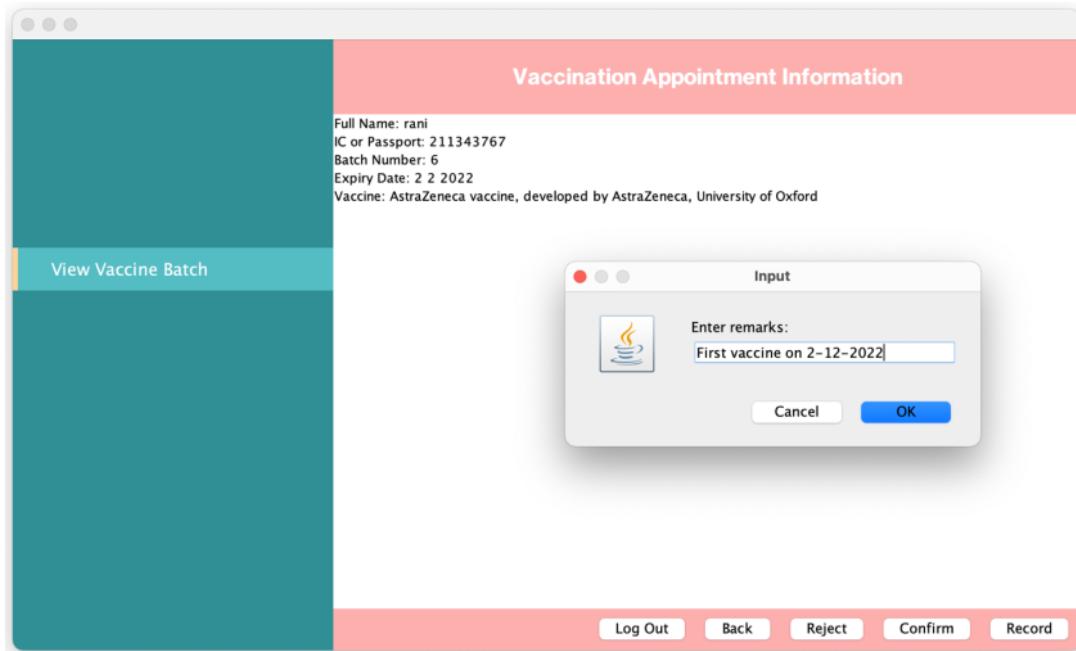
Picture 11: Record Vaccination Administered



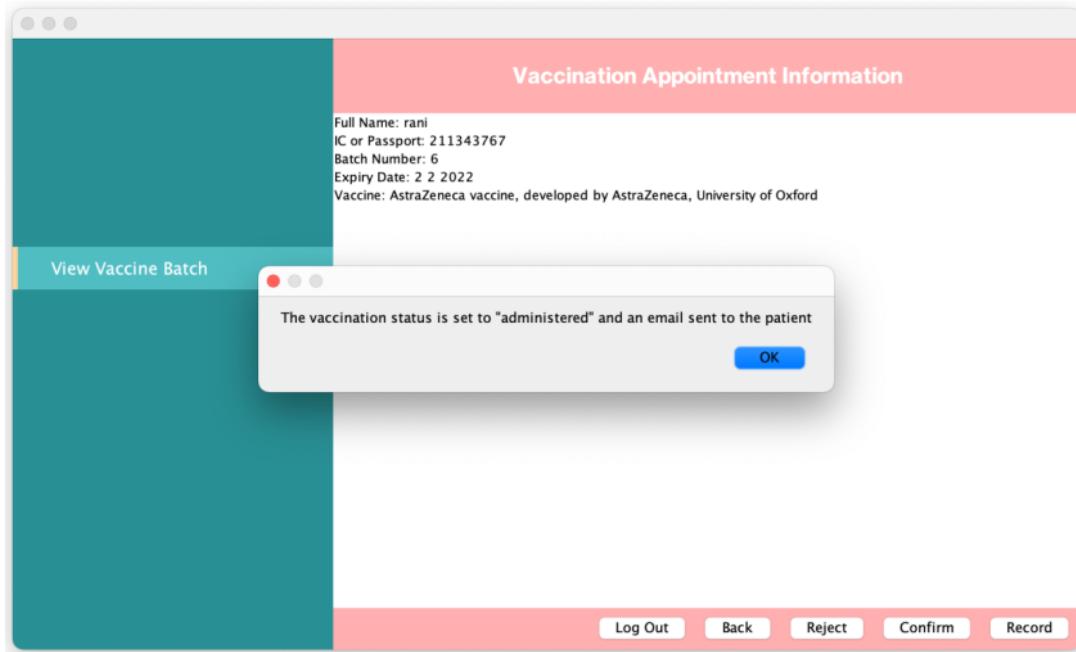
Picture 12: Record Vaccination Administered



Picture 13: Record Vaccination Administered

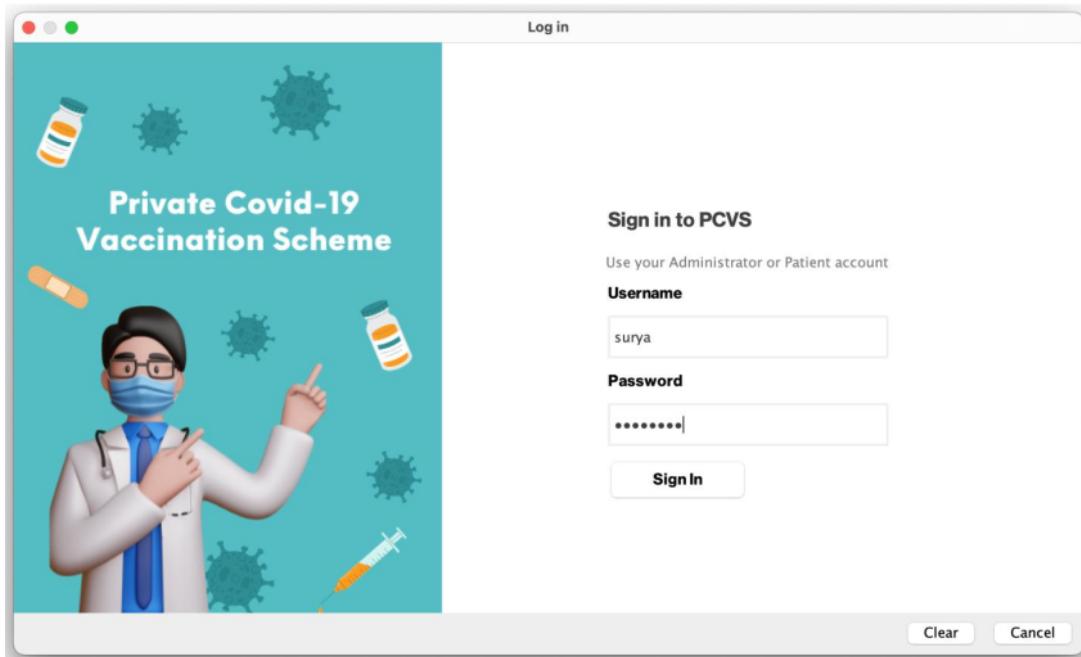


Picture 14: Record Vaccination Administered

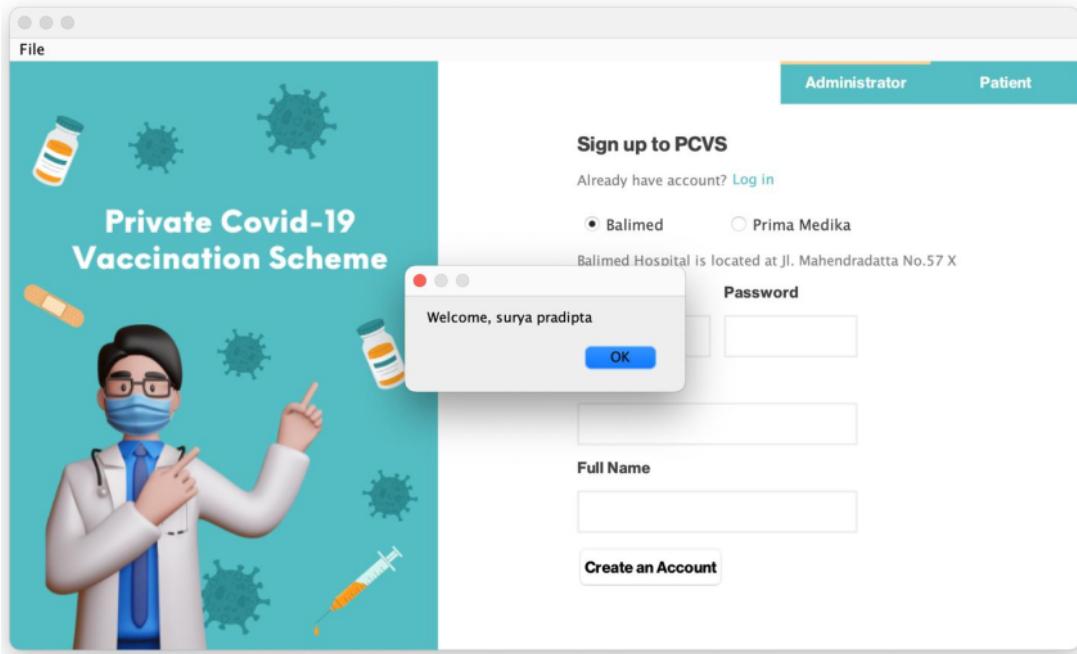


Picture 15: Record Vaccination Administered

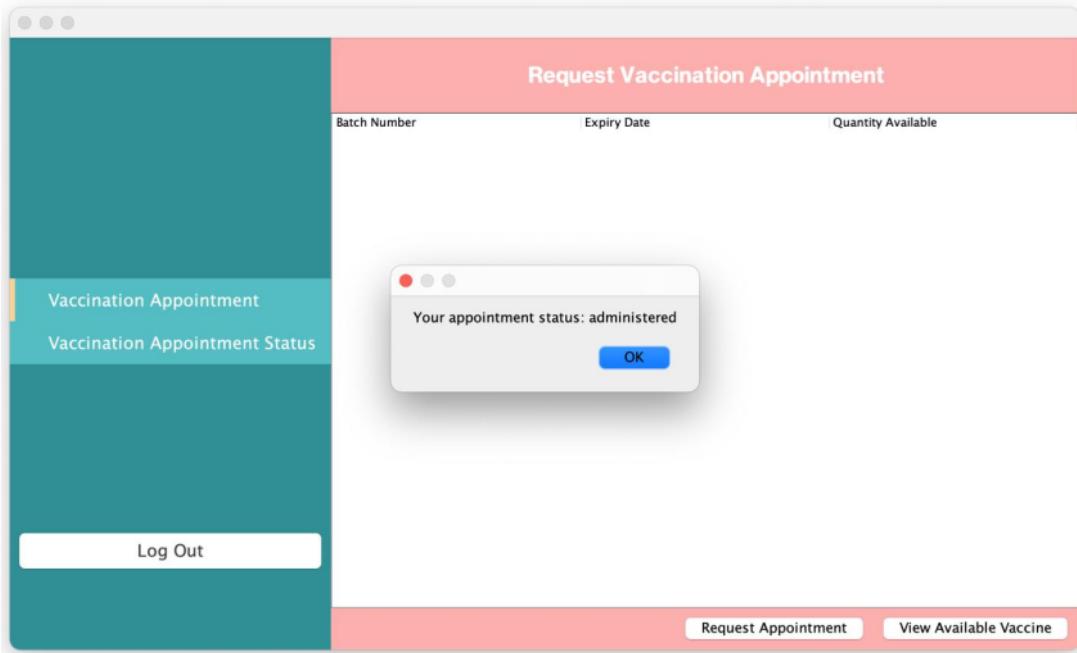
Use Case 7 View Vaccination Appointment Status



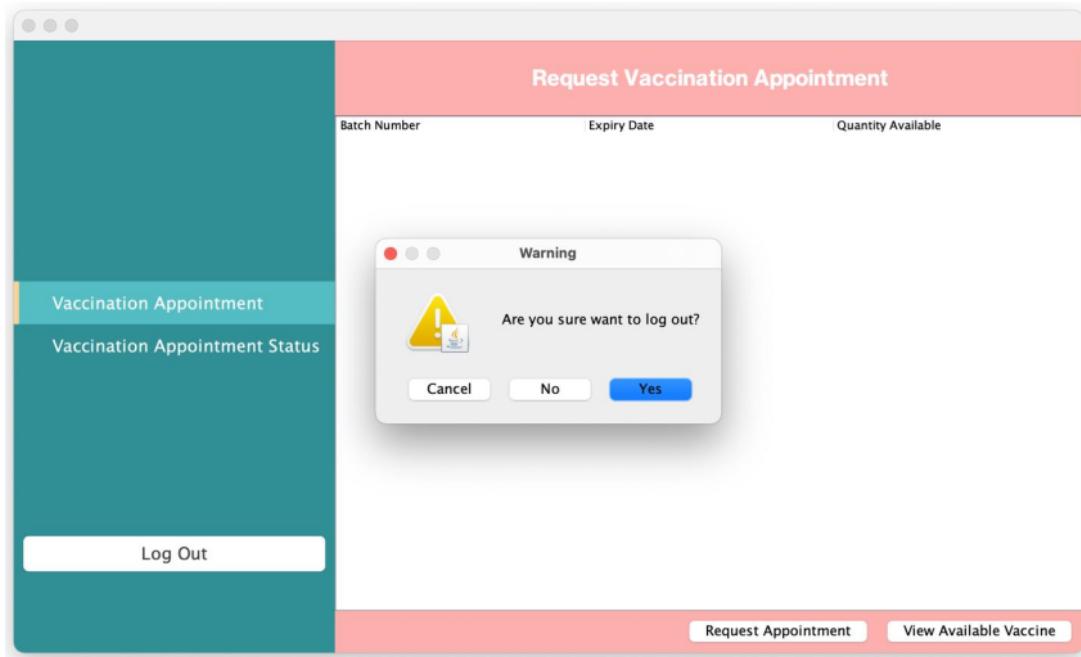
Picture 1: Vaccination Appointment Status



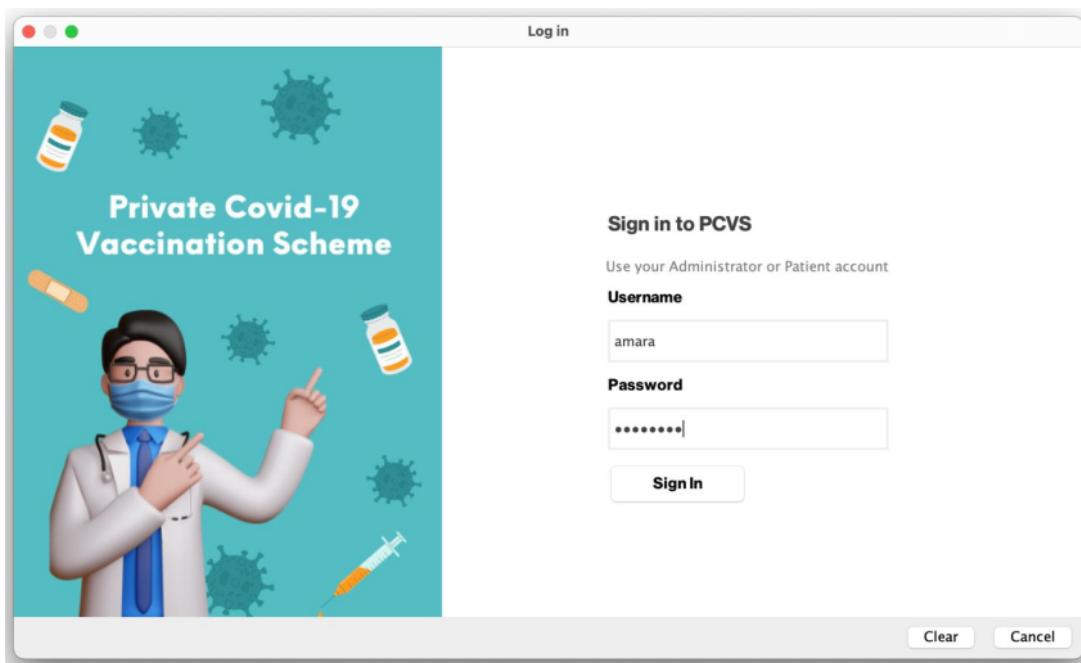
Picture 1.1: Vaccination Appointment Status



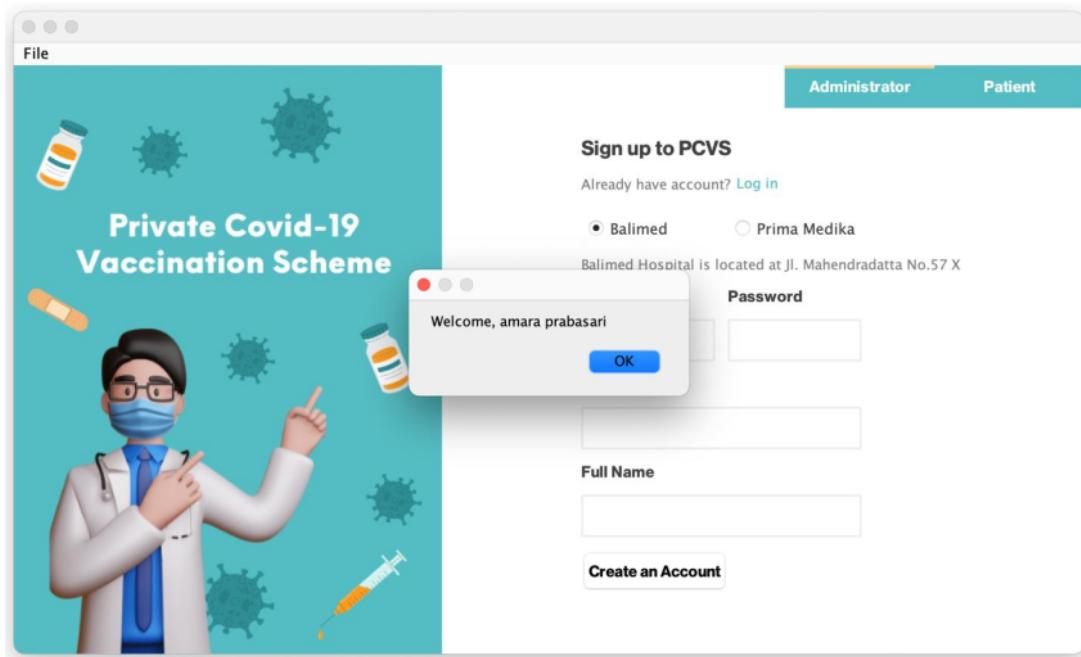
Picture 1.2: Vaccination Appointment Status



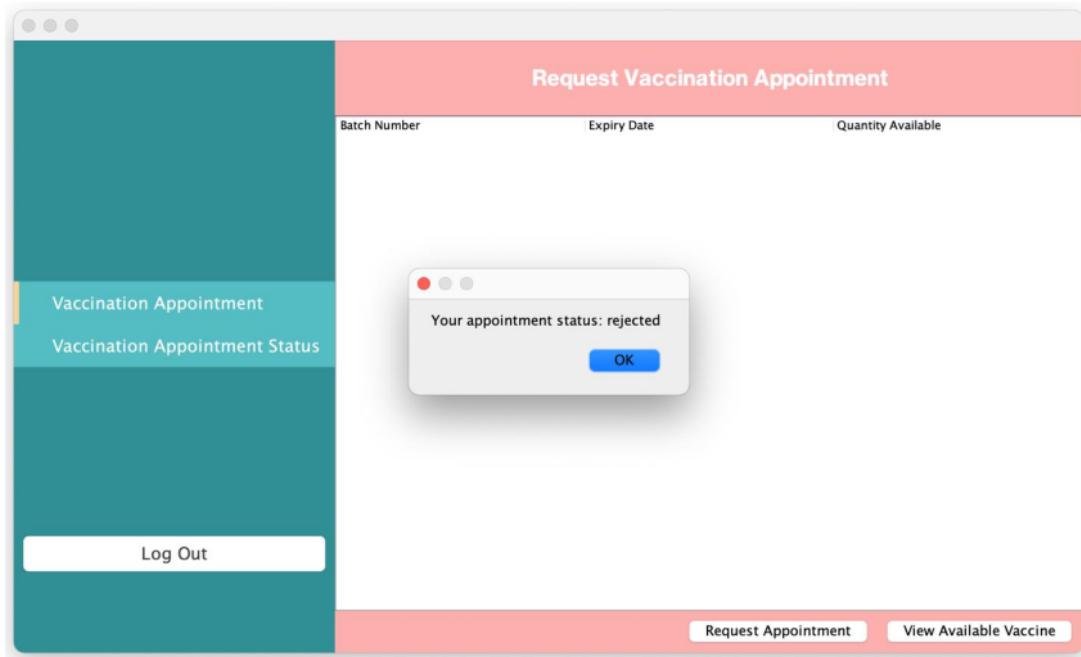
Picture 1.3: Vaccination Appointment Status



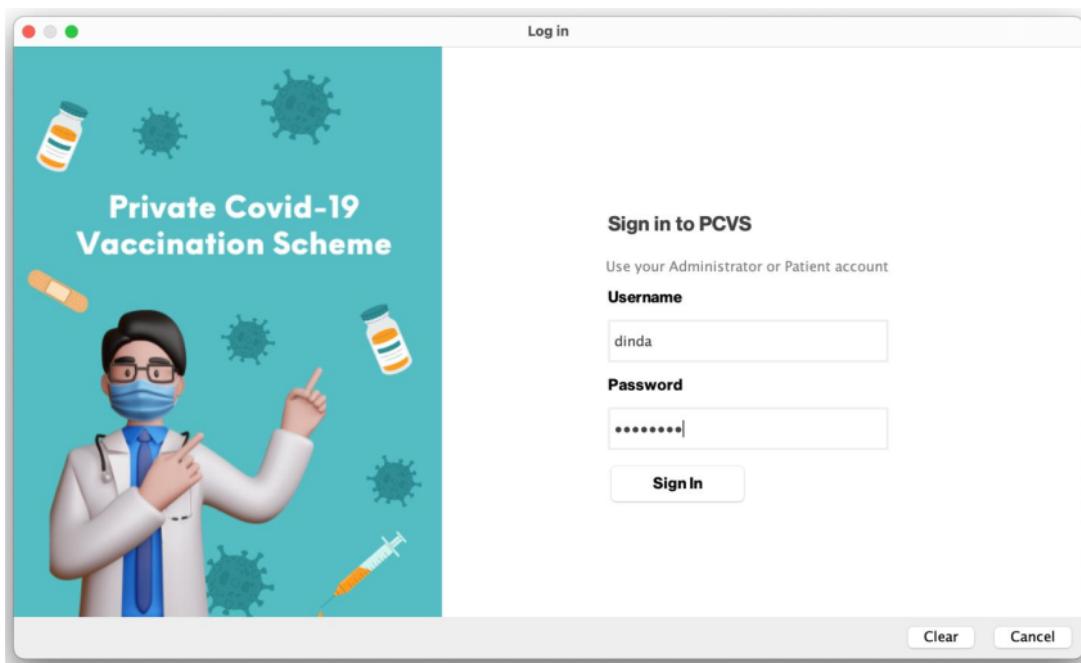
Picture 2: Vaccination Appointment Status



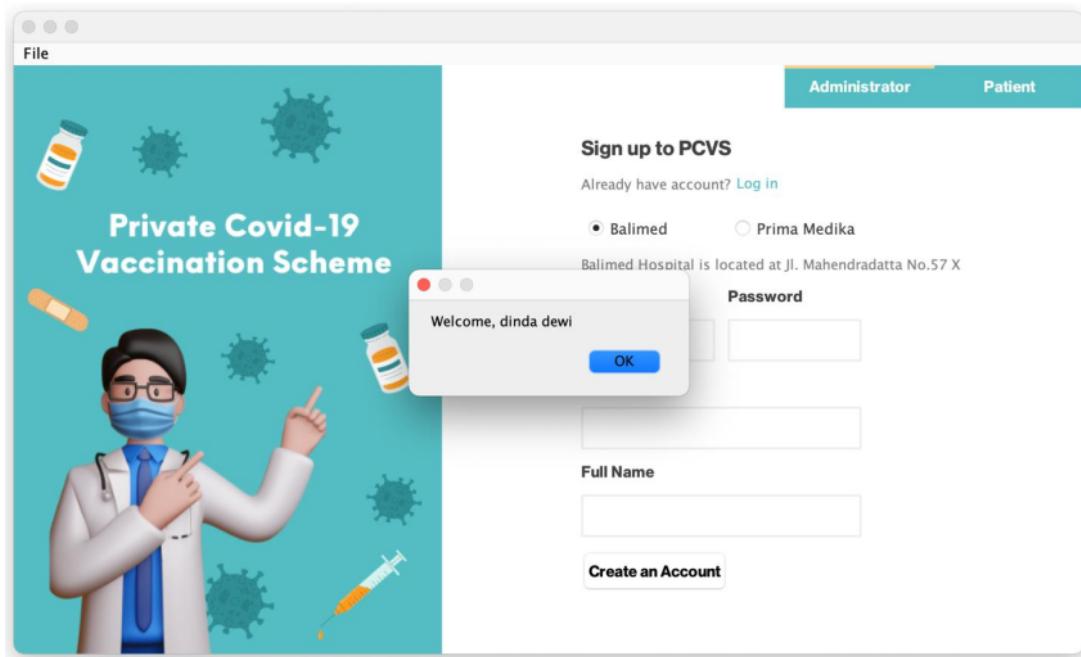
Picture 2.1: Vaccination Appointment Status



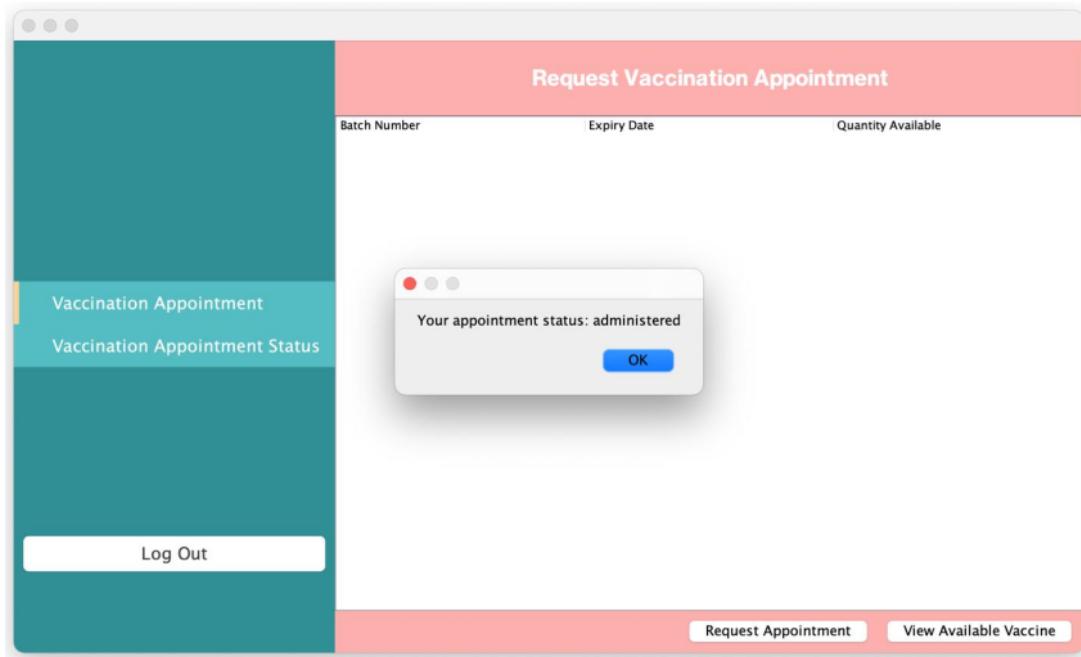
Picture 2.3: Vaccination Appointment Status



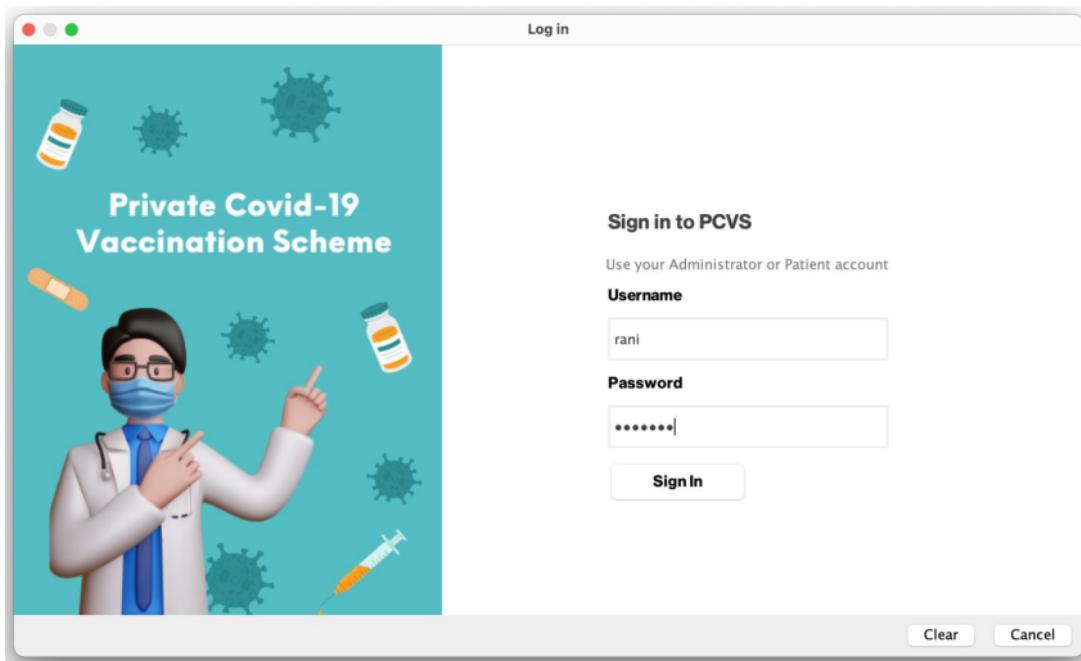
Picture 3: Vaccination Appointment Status



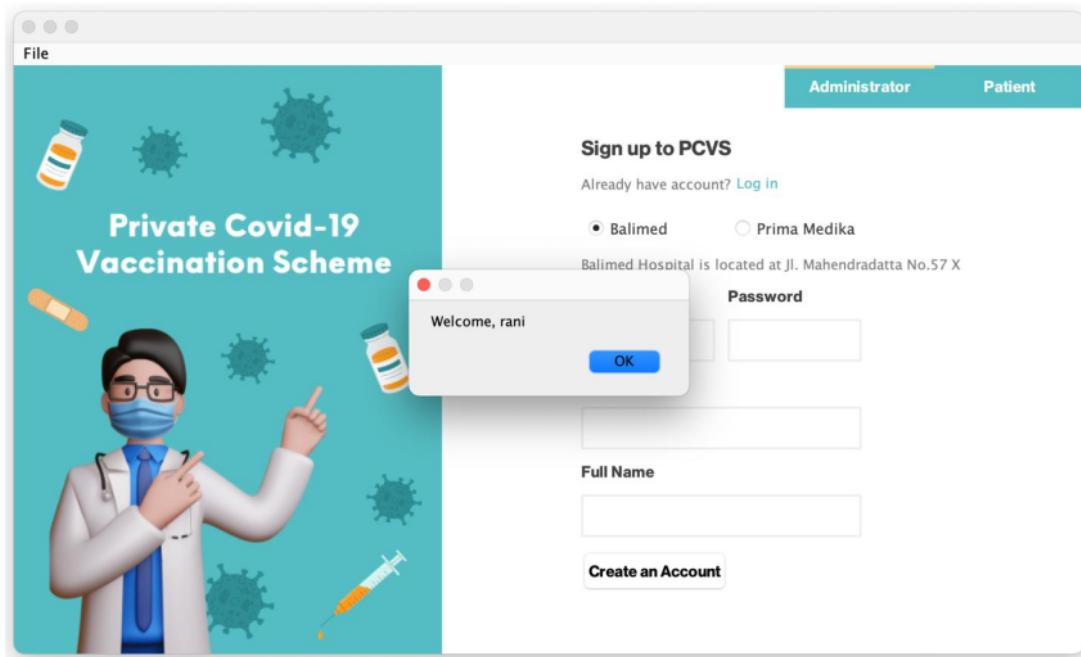
Picture 3.1: Vaccination Appointment Status



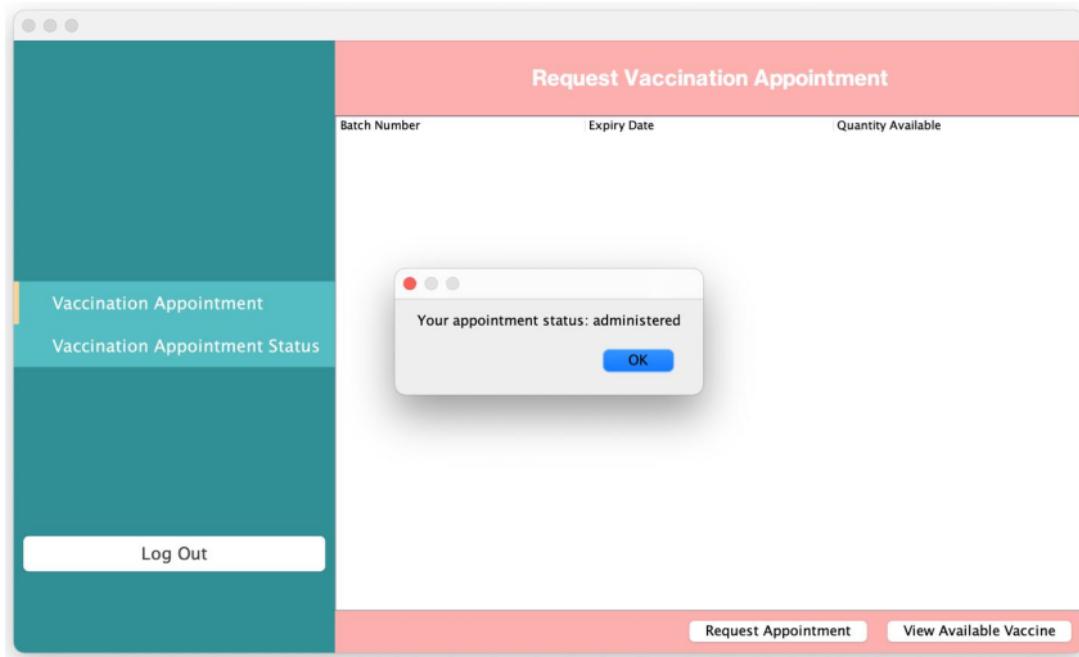
Picture 3.2: Vaccination Appointment Status



Picture 4: Vaccination Appointment Status

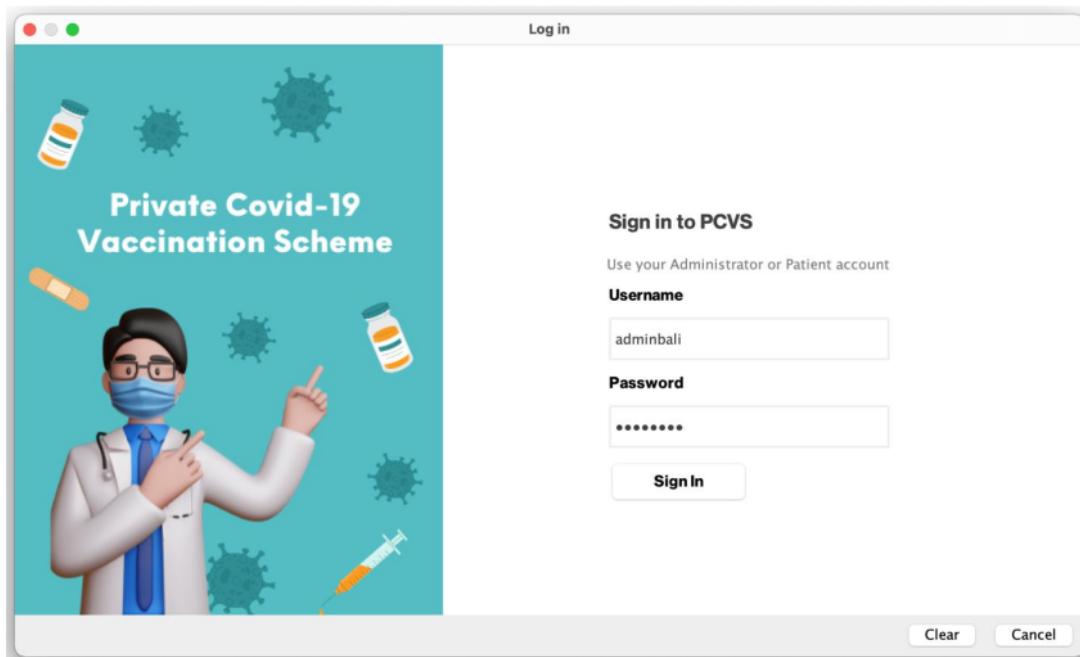


Picture 4.1: Vaccination Appointment Status

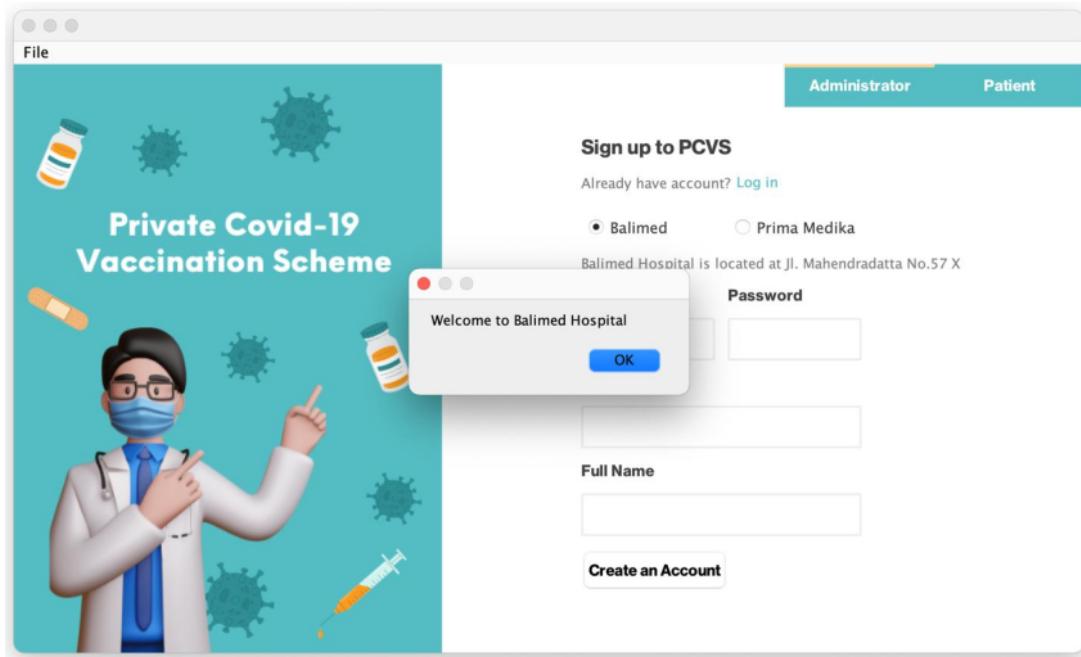


Picture 4.2: Vaccination Appointment Status

Use Case 8 Detail of All User



Picture 1: Detail User Information



Picture 1.2: Detail User Information

Administrator Information				
Username	Password	Email	Full Name	Staff ID
adminprima	admin123	adminprimamedika@... admin prima medika ... ADM000		
adminbali	admin123	adminbalimed@pcvs.... admin balimed hospital ADM001		

Administrator Information

[Original Order](#) [Sorted by Full Name](#) [Delete Selection](#) [Back](#)

Picture 2: Detail User Information Original Order

Administrator Information				
Username	Password	Email	Full Name	Staff ID
adminbali	admin123	adminbalimed@pcvs.... admin balimed hospital ADM001		
adminprima	admin123	adminprimamedika@... admin prima medika ... ADM000		

Administrator Information

[Original Order](#) [Sorted by Full Name](#) [Delete Selection](#) [Back](#)

Picture 2.1: Detail User Information Sorted by Full Name

Patient Information				
Username	Password	Email	Full Name	IC or Passport
surya	surya123	suryapradipta@gmail...	surya pradipta	211343555
amara	amara123	amaraprabasari@gm...	amara prabasari	211343414
dinda	dinda123	dinda@gmail.com	dinda dewi	211343909
rani	rani123	rani@gmail.com	rani	211343767

Picture 3: Detail User Information Original Order

Patient Information				
Username	Password	Email	Full Name	IC or Passport
amara	amara123	amaraprabasari@gm...	amara prabasari	211343414
dinda	dinda123	dinda@gmail.com	dinda dewi	211343909
rani	rani123	rani@gmail.com	rani	211343767
surya	surya123	suryapradipta@gmail...	surya pradipta	211343555

Picture 3.1: Detail User Information Sorted by Full Name

Use Case 9 Display Detail of All Vaccination Appointments

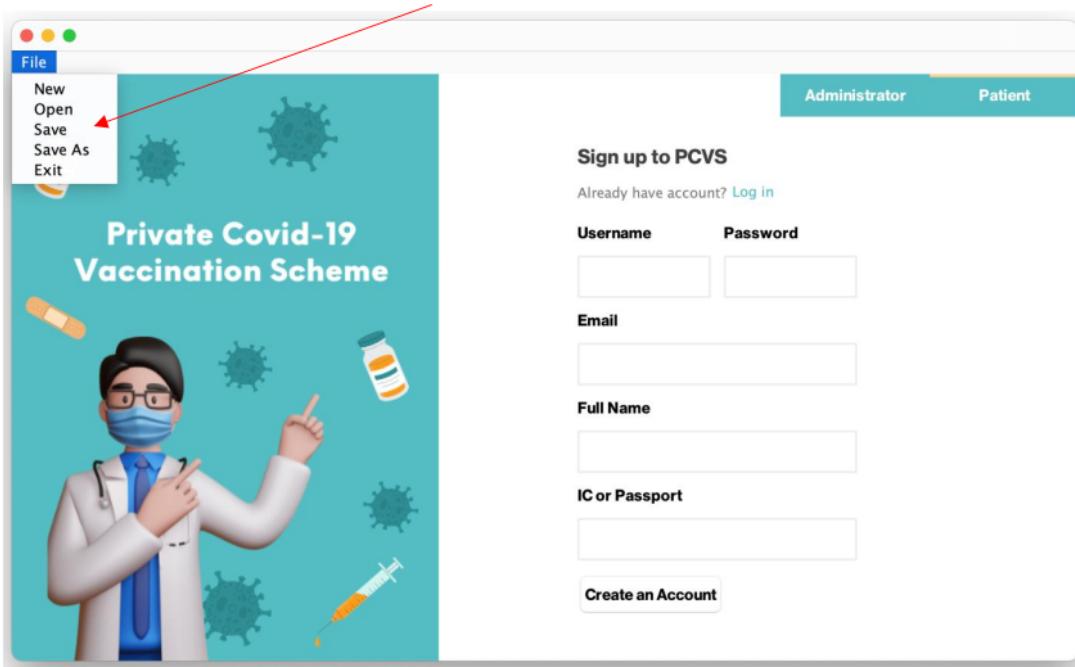
The screenshot shows a software application window with a title bar 'Vaccination Information'. The main content area is a table with four columns: 'Vaccination ID', 'Appointment Date', 'Status', and 'Remarks'. The table contains four rows of data:

Vaccination ID	Appointment Date	Status	Remarks
V000	2 3 2022	administered	First vaccine on 2-3-2022
V001	2 3 2022	rejected	The number of available va...
V002	2 7 2022	administered	First vaccine on 2-7-2022
V003	2 12 2022	administered	First vaccine on 2-12-2022

Picture 1: Detail Vaccination Information

Save File

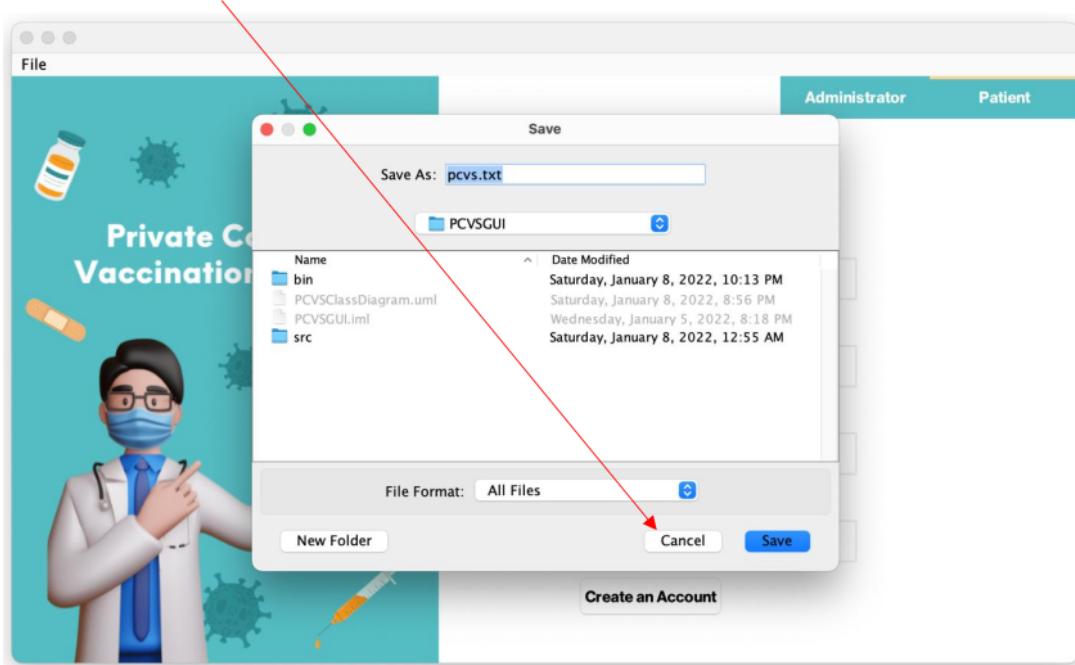
File + Save clicked



Picture 1: Save PCVS Data to File

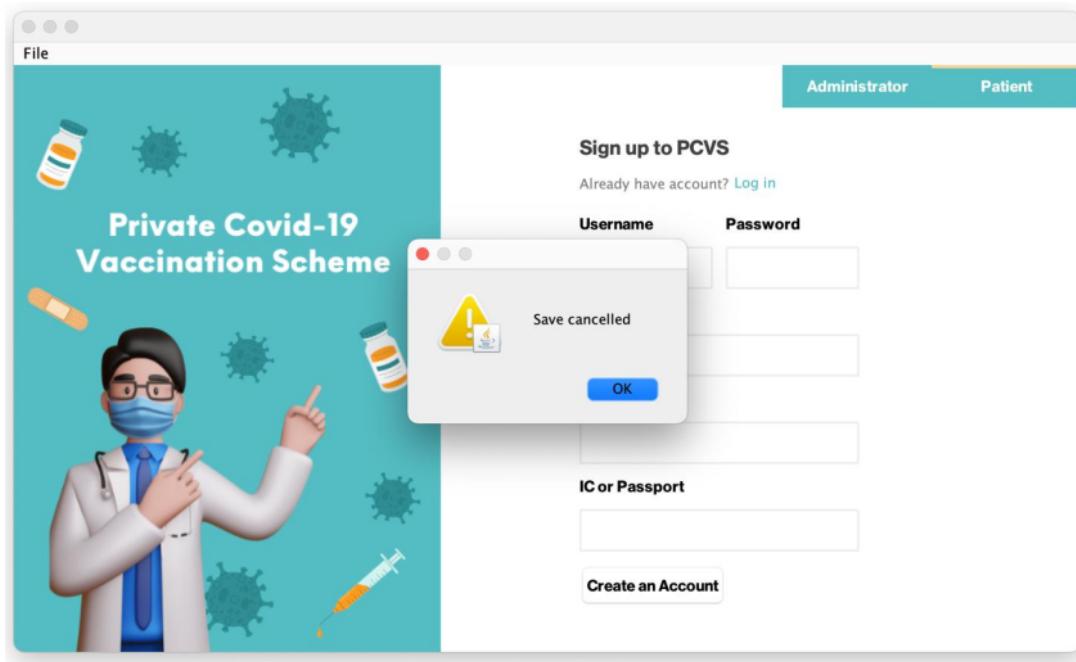
Save File Validation

Cancel clicked



Picture 1.1: Save PCVS Data to File

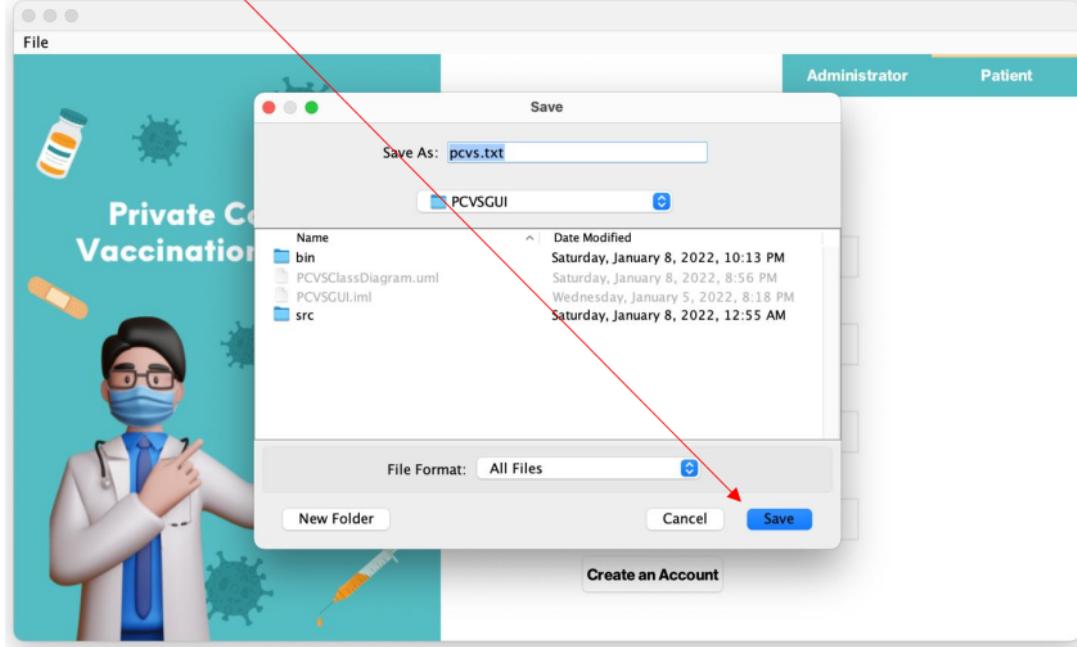
Save File Validation



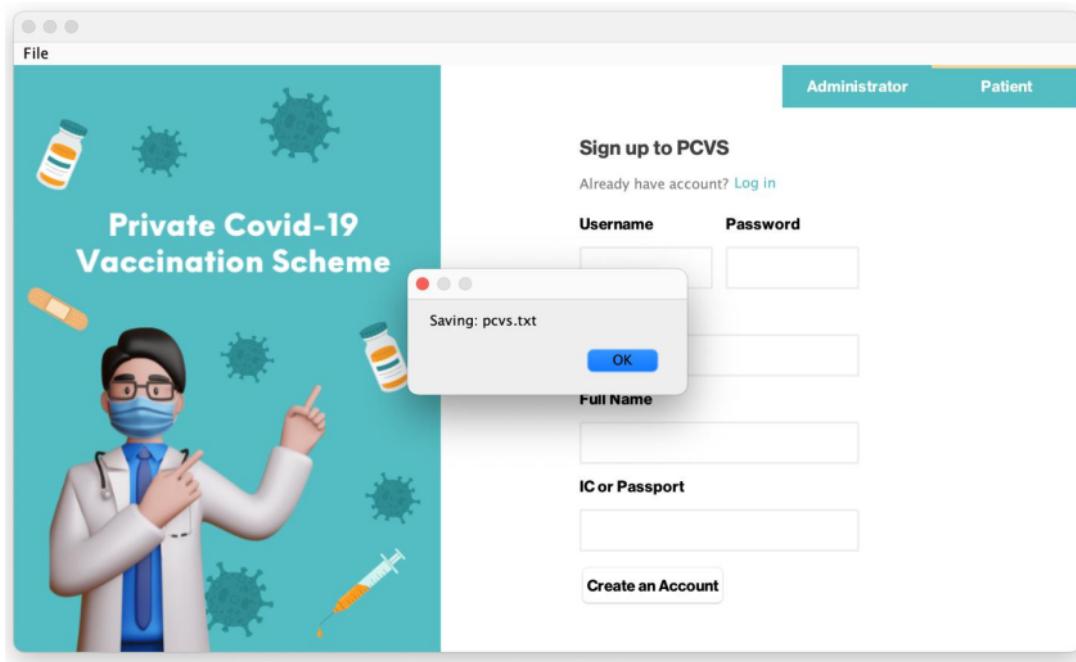
Picture 1.2: Save PCVS Data to File

Save File Validation

Save clicked

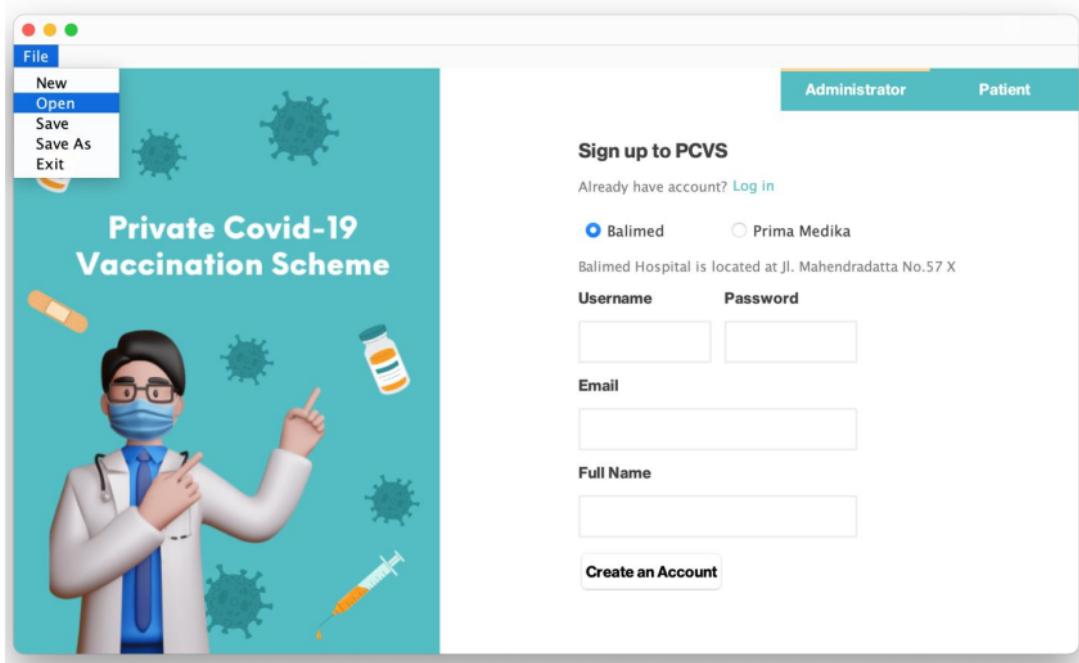


Picture 2: Save PCVS Data to File

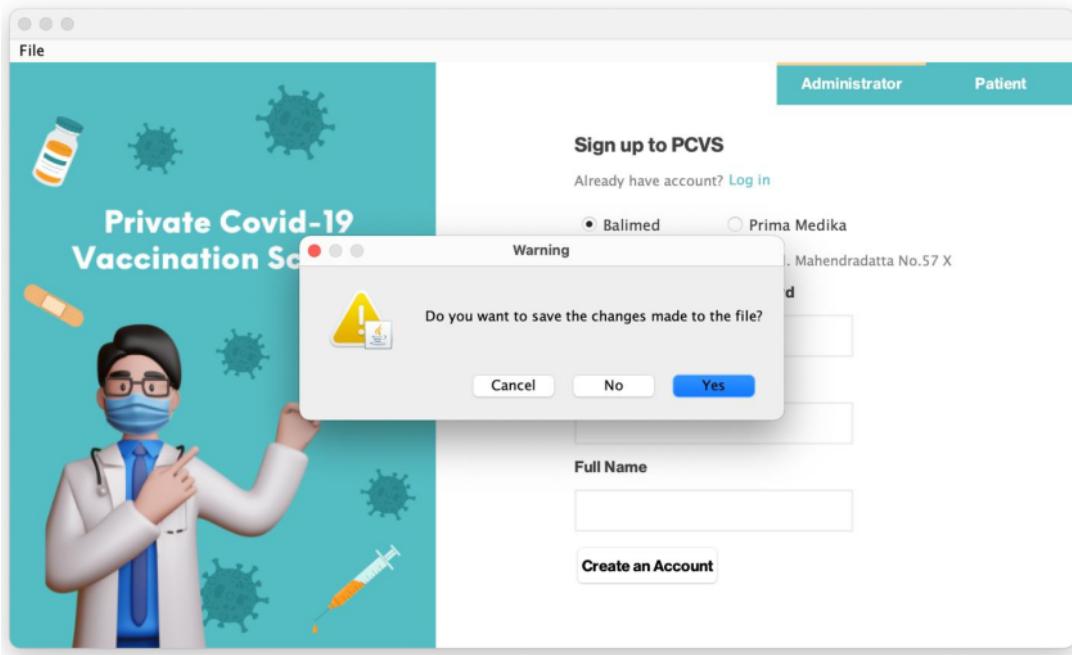


Picture 2.1: Save PCVS Data to File

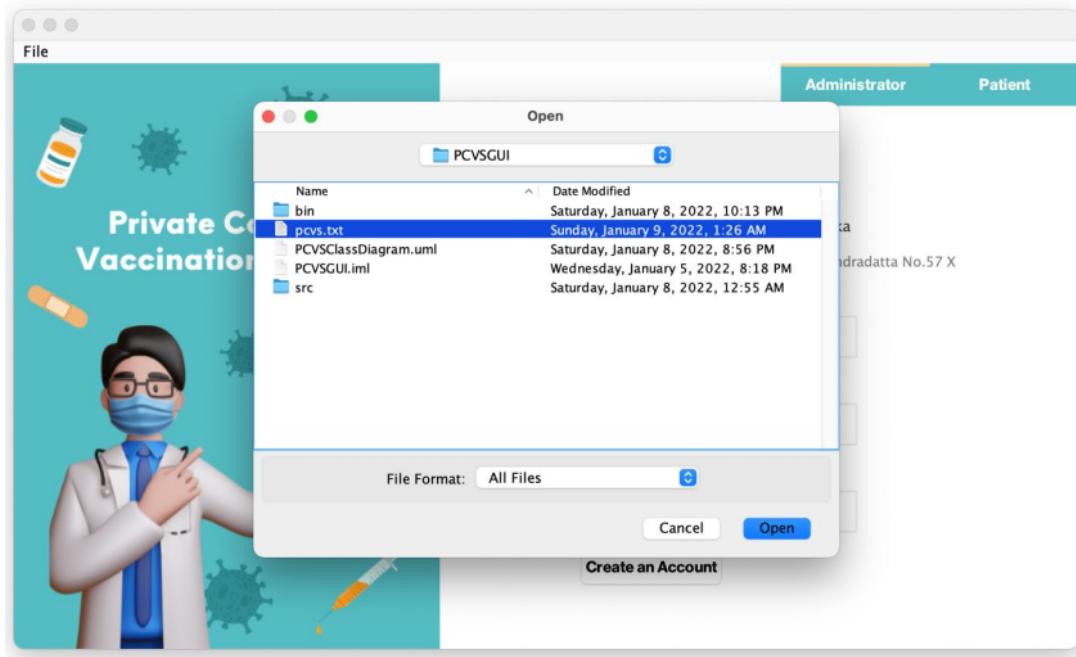
Load File



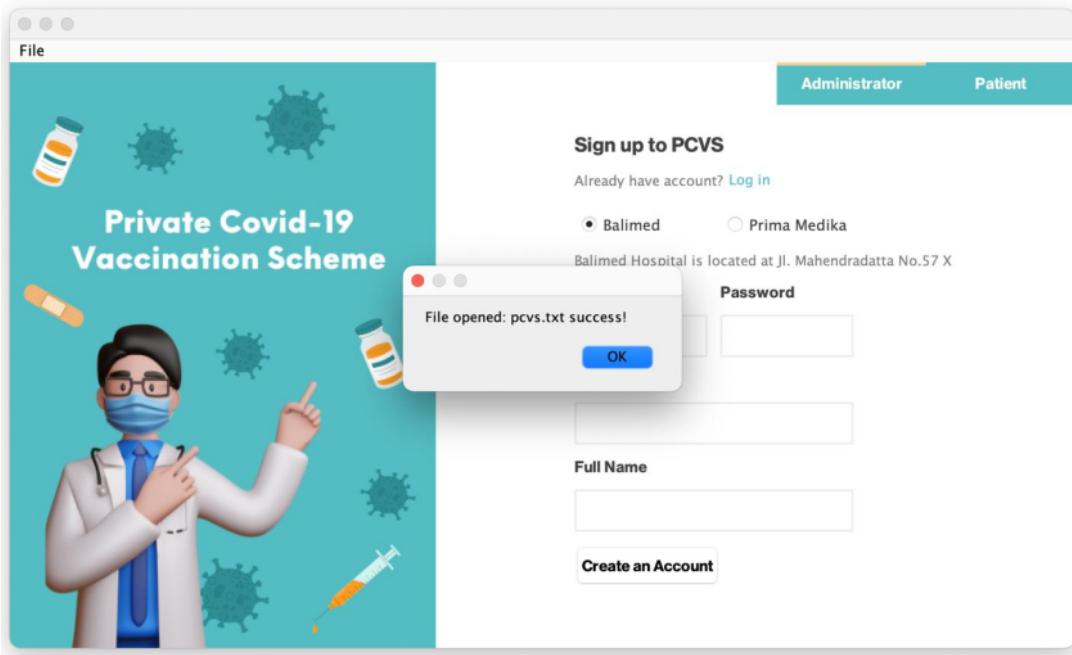
Picture 1: Load PCVS Data from File



Picture 2: Load PCVS Data from File



Picture 3: Load PCVS Data from File



Picture 4: Load PCVS Data from File

The screenshot shows a Java Swing application window titled "Administrator Information". The window has a pink header bar and a teal sidebar on the left labeled "Administrator Information". The main content area displays a table with five columns: Username, Password, Email, Full Name, and Staff ID. The data in the table is as follows:

Username	Password	Email	Full Name	Staff ID
adminprima	admin123	adminprimamedika@...	admin prima medika ...	ADM000
adminbali	admin123	adminbalimed@pcvs....	admin balimed hospital	ADM001

At the bottom of the window, there is a pink footer bar with four buttons: "Original Order", "Sorted by Full Name", "Delete Selection", and "Back".

Picture 5: Load PCVS Data from File

Checking Data

Vaccination Information			
Vaccination ID	Appointment Date	Status	Remarks
V000	2 3 2022	administered	First vaccine on 2-3-2022
V001	2 3 2022	rejected	The number of available va...
V002	2 7 2022	administered	First vaccine on 2-7-2022
V003	2 12 2022	administered	First vaccine on 2-12-2022

Picture 5.1: Load PCVS Data from File

Checking Data

Patient Information

Username	Password	Email	Full Name	IC or Passport
surya	surya123	suryapradipta@gmail...	surya pradipta	211343555
amara	amaral23	amaraprabasari@gm...	amara prabasari	211343414
dinda	dinda123	dinda@gmail.com	dinda dewi	211343909
rani	rani123	rani@gmail.com	rani	211343767

Patient Information

[Original Order](#)
 [Sorted by Full Name](#)
 [Delete Selection](#)
 [Back](#)

Picture 5.2: Load PCVS Data from File

Checking Data

ORIGINALITY REPORT

12%
SIMILARITY INDEX

8%
INTERNET SOURCES

4%
PUBLICATIONS

8%
STUDENT PAPERS

PRIMARY SOURCES

- | | | |
|----------|---|----------------|
| 1 | Submitted to HELP UNIVERSITY
Student Paper | 5% |
| 2 | fr.wikibooks.org
Internet Source | 1 % |
| 3 | gitlab.devops.ifrn.edu.br
Internet Source | 1 % |
| 4 | fanal.jxta.org
Internet Source | <1 % |
| 5 | courses.cs.washington.edu
Internet Source | <1 % |
| 6 | David Parsons. "Chapter 18 Event-Driven Programming", Springer Science and Business Media LLC, 2012
Publication | <1 % |
| 7 | coned.utcluj.ro
Internet Source | <1 % |
| 8 | weblog.plexobject.com
Internet Source | <1 % |

9	Dietmar Abts. "Grundkurs JAVA", Springer Science and Business Media LLC, 2016 Publication	<1 %
10	gitlab.nps.edu Internet Source	<1 %
11	www.anyang-window.com.cn Internet Source	<1 %
12	gitlab.engr.ship.edu Internet Source	<1 %
13	ada.csse.rose-hulman.edu Internet Source	<1 %
14	jnjsite.com Internet Source	<1 %
15	Olaf Musch. "Chapter 5 Observer", Springer Science and Business Media LLC, 2021 Publication	<1 %
16	Submitted to University of Northampton Student Paper	<1 %
17	www.digit.in Internet Source	<1 %
18	Submitted to Middlesex University Student Paper	<1 %
19	Submitted to fultonschools Student Paper	<1 %

20	learn.tech-web.vn Internet Source	<1 %
21	blog.livedoor.jp Internet Source	<1 %
22	alreadyno.tistory.com Internet Source	<1 %
23	"Beginning Java® Programming", Wiley, 2012 Publication	<1 %
24	Kishori Sharan. "Beginning Java 8 APIs, Extensions and Libraries", Springer Science and Business Media LLC, 2014 Publication	<1 %
25	www.java-forum.org Internet Source	<1 %
26	gitlab.stud.idi.ntnu.no Internet Source	<1 %
27	sourceforge.net Internet Source	<1 %
28	Submitted to unina Student Paper	<1 %
29	jexp.ru Internet Source	<1 %
30	Gregory Mashanov, Alla Mashanova. "The role of spatial structure in the infection	<1 %

spread models: population density map of
England example", Cold Spring Harbor
Laboratory, 2020

Publication

- | | | |
|----|--|------|
| 31 | community.esri.com | <1 % |
| 32 | git.tecgraf.puc-rio.br | <1 % |
| 33 | www.programacion.com.py | <1 % |
| 34 | Jay Bryant. "Java 7 for Absolute Beginners",
Springer Science and Business Media LLC,
2011 | <1 % |
| 35 | stackoverflow.com | <1 % |
| 36 | David Parsons. "Foundational Java", Springer
Science and Business Media LLC, 2020 | <1 % |
| 37 | forum.gpwiki.org | <1 % |
| 38 | lists.wald.intevation.org | <1 % |
| 39 | pastebin.com | <1 % |

40	community.oracle.com Internet Source	<1 %
41	gitext.gfz-potsdam.de Internet Source	<1 %
42	javaprogramcodes.blogspot.com Internet Source	<1 %
43	lists.gforge.inria.fr Internet Source	<1 %
44	tech.it168.com Internet Source	<1 %
45	Barry Burd. "Java® For Dummies®", Wiley, 2011 Publication	<1 %

Exclude quotes Off
Exclude bibliography Off

Exclude matches Off