ORCHID INTERNATIONAL College

(Affiliated to Tribhuvan University)

A

Project Report

on

Advanced Java Programming

Project Title: Movie Ticket System (MovieMazza)

Submitted By:

Name: Narotsit Karki

Roll No: 238677/076

Semester: 7th

Submitted To:

Department of CSIT

Orchid International College

____

Introduction

Movie Ticketing System (MovieMazza) is a Swing and AWT based Java GUI application that allows users to scroll through latest released movies and allow them to buy tickets at a reasonable price, The project focuses on different aspects of GUI and Database applications like fast responding GUI, transaction handlings, and some basic security aspects for securing the application.

Technologies Used

- Javax Swing
- Java AWT
- Java SQL

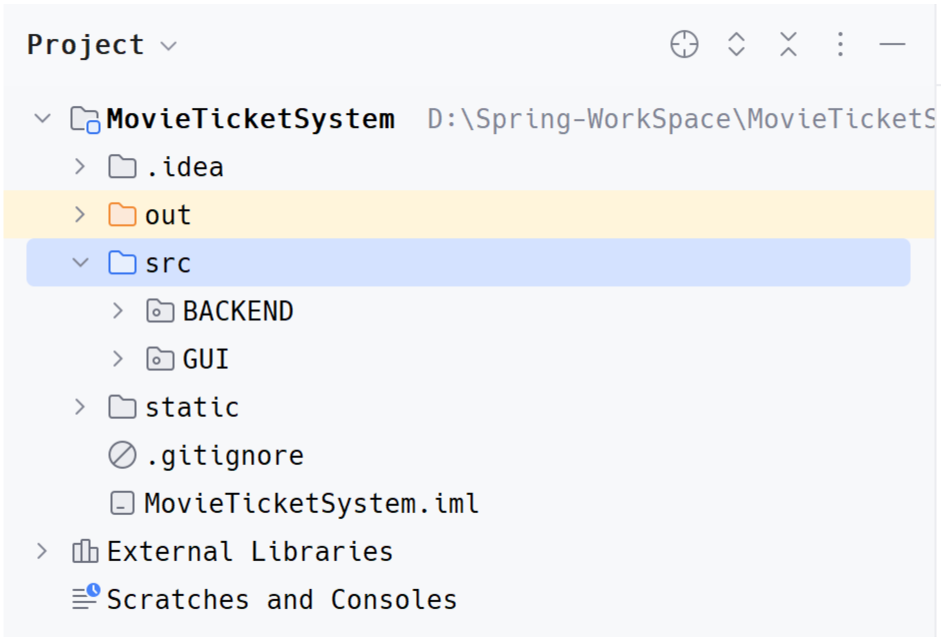
Features

- User driven GUI application, for user usability, efficiency and faster response.
- Ticket booking and payment integrated within the application.
- Use of Singleton Architecture to creating Many Data Base connection classes and Data Base Operation classes to minimize memory usage and enhance concept of object reusability.
- Let users track their bookings, and get up to date with their movie ticket booking payments.
- Let user change their personal information , and also password for security measures.

Limitations And Further Development

- Currently Lacks other effective ways for notifying users so we plan to add features , such as mail through Java Mail API to notify customers about their bookings and payments.
- Currently Lacks proper security measures, currently only password based authentication is in place, to further secure the application we can use authorization concepts and rules and policies implementation.
- Currently Lacks proper session management, currently only user will authenticate and object corresponding to an authenticated User gets created which acts as security object.
- Currently Lacks proper payment system, so we can integrate 3rd party -payment systems in the future.

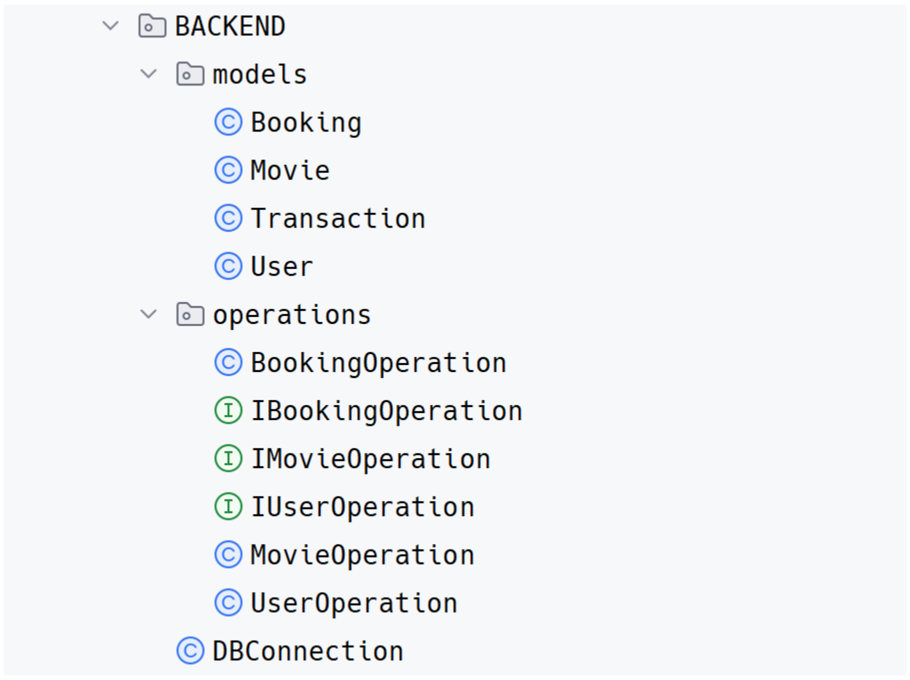
Project Structure



```
Project
MovieTicketSystem  D:\Spring-WorkSpace\MovieTicketS
  .idea
  out
  src
    BACKEND
    GUI
  static
  .gitignore
  MovieTicketSystem.iml
External Libraries
Scratches and Consoles
```

Project Structure for the Movie Ticketing System is divided into mainly **Two Parts / Packages**.

- **BACKEND**

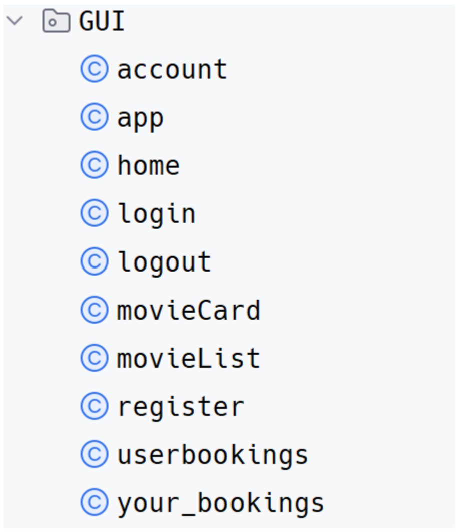


```
BACKEND
  models
    Booking
    Movie
    Transaction
    User
  operations
    BookingOperation
    IBookingOperation
    IMovieOperation
    IUserOperation
    MovieOperation
    UserOperation
  DBConnection
```

The Backend package has 3 main parts

- **models**:
  This package consists of all the class representation of the tables in the database.
- **operations**:
  This package consists of interfaces and class implementations of database operation on tables defined in the **models** class.
- **DBconnection**:
  This class is what helps to create singleton database connection.

- **GUI**



```
GUI
    account
    app
    home
    login
    logout
    movieCard
    movieList
    register
    userbookings
    your_bookings
```

The GUI also called the front end is the part where UI/UX handling takes place.

- **account** class for user account setting and changing password , email and other info.
- **app** class is the entry point for the whole project (i.e starter for the application).
- **home** class is the main frame which consists of all other components.
- **login** , **register** and **logout** are basic forms for user authentication and user registration.
- **movieCard** is a component that displays movie information in a card like structure.
- **movieList** components shows all the movies in Db as list of **movieCard**.
- **your_bookings** shows list of **userbookings (**i.e list of bookings made from user).

Database Schema

Here is the Class Diagram for the Movie Ticketing System Project.

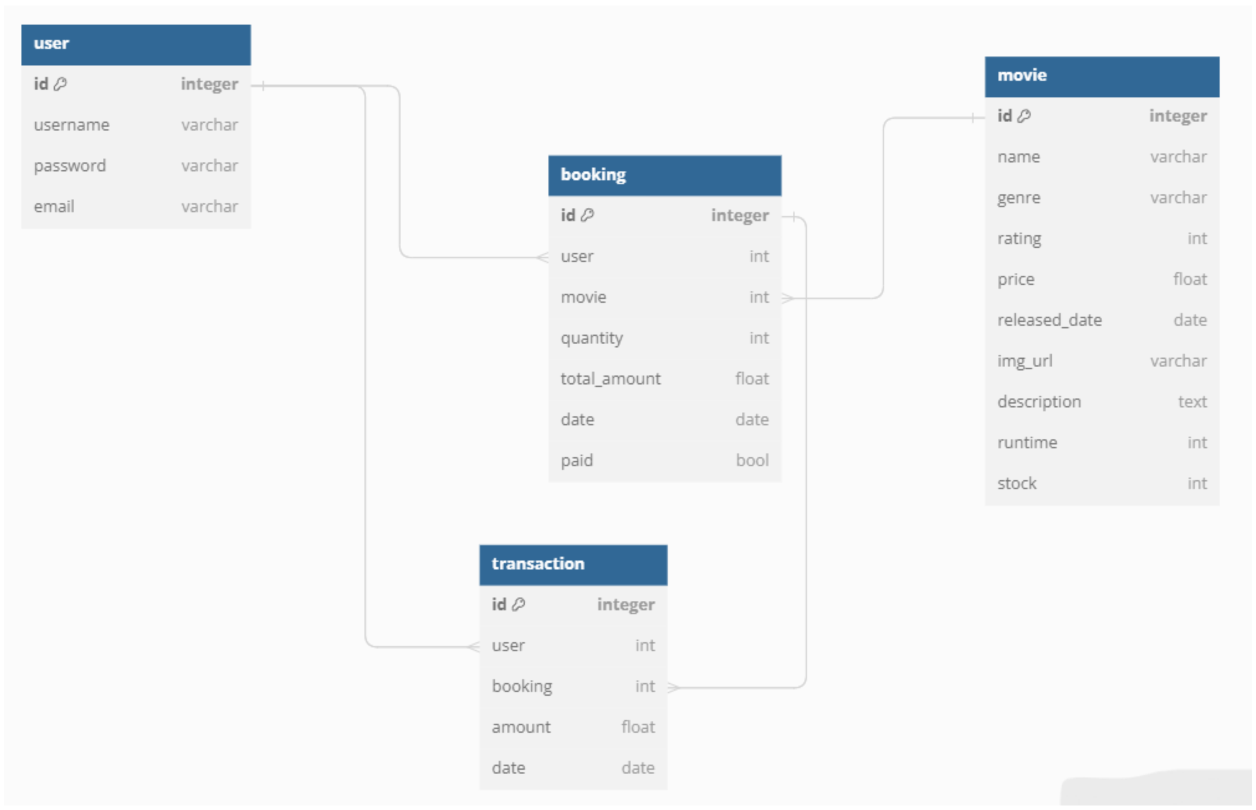


Figure 1 Class Diagram for Movie Ticketing System

Code Details

Backend

- DBConnection (Establish Singleton Database connection)

```
public class DBConnection {

    private static String _connection_class =
"com.mysql.cj.jdbc.Driver";
    private static String _connection_url =
"jdbc:mysql://localhost:3306/moviedb";
    private static String _user = "root";
    private static String _password = "root";

    private static Connection _conn = null;

    public static Connection getConnection(){
        if (_conn != null)
            return _conn;
        try {
            Class.forName(_connection_class);
            _conn =
DriverManager.getConnection(_connection_url,_user,_password);
        }catch(ClassNotFoundException ex){
            ex.printStackTrace();
        }catch (SQLException ex){
            ex.printStackTrace();
        }

        return _conn;
    }
}
```

- Models (Class Representation for Database tables)

1. User Class:

```
public class User {
    public String email;
    public String username;
    public String password;
    private static User newUser = null;

    public int id;
    public User(String email,String username,String password){
        this.email = email;
        this.username = username;
        this.password = User.hashPassword(password);
    }
    public User(int id,String email,String username,String
password){
        this.email = email;
        this.username = username;
        this.password = password;
        this.id = id;
    }

//    SHA-256 based hasing for security purposes
    public static String hashPassword(String password){
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-
256");
            byte[] hash =
digest.digest(password.getBytes(StandardCharsets.UTF_8));
            String encoded =
Base64.getEncoder().encodeToString(hash);
            return encoded;
        }catch(NoSuchAlgorithmException ex) {
            return "";
        }
    }

//    global user variable
    public static User getInstance(){
        return newUser;
    }
    public static User createInstance(int id,String email,String
username,String password){
        if(newUser == null){
            newUser = new User(id,email,username,password);
        }
        return newUser;
    }

//    for logout purposes
    public static boolean removeInstance(){
        if (newUser != null){
            newUser = null;
            return true;
        }
        return false;
    }
}
```

2. Movie Class

```
public class Movie {
    public int id;
    public String name;
    public String genre;
    public String description;
    public String released_date;
    public float price;
    public int rating;
    public int runtime;
    public int stock;
    public String image_url;
    String base_path = "D:/Spring-
WorkSpace/MovieTicketSystem/static/images/";
    public Movie(int id, String name, String genre, String
description, String released_date, float price, int runtime,int
rating,int stock,String image_url) {
        this.id = id;
        this.name = name;
        this.genre = genre;
        this.description = description;
        this.released_date = released_date;
        this.price = price;
        this.runtime = runtime;
        this.rating = rating;
        this.stock = stock;
        this.image_url = base_path+image_url;
    }
}
```

3. Transaction Class:

```
public class Transaction {
    public int id;
    public int user;
    public int booking;
    public float amount;
    public String date;

    public Transaction(int user, int booking, float amount,
String date) {
        this.user = user;
        this.booking = booking;
        this.amount = amount;
        this.date = date;
    }

    public Transaction(int id, int user, int booking, float
amount, String date) {
        this.id = id;
        this.user = user;
        this.booking = booking;
        this.amount = amount;
        this.date = date;
    }
}
```

4. Booking Class:

```
public class Booking {

    public int id;
    public int quantity;
    public int user;
    public int movie;
    public float total_amount;
    public String date;
    public boolean paid;

    public Booking(int quantity, int user, int movie, float
total_amount,String date) {
        this.quantity = quantity;
        this.user = user;
        this.movie = movie;
        this.total_amount = total_amount;
        this.date = date;
        this.paid = false;
    }

    public Booking(int id, int quantity, int user, int movie,
float total_amount,String date,boolean paid) {
        this.id = id;
        this.quantity = quantity;
        this.user = user;
        this.movie = movie;
        this.total_amount = total_amount;
        this.date = date;
        this.paid = paid;
    }
}
```

- Operations (Interfaces and Class Implementation for all models described above)

1. User database operations

```
public interface IUserOperation {

    public boolean register(User user);
    public User authenticate(String username,String password);
    public boolean changeInformation(User user);
    public boolean logout();
    public User getById(int id);

}
```

```
public class UserOperation implements IUserOperation{
    private final Connection _conn;

    public static UserOperation udb = null;

    public static UserOperation createOperation(Connection conn){
        if(udb == null){
            udb = new UserOperation(conn);
        }
        return udb;
    }

    public static  UserOperation getOperation(){
        return udb;
    }
    public UserOperation(Connection conn){
        this._conn = conn;
    }

    @Override
    public boolean register(User user) {
        try{
            PreparedStatement pstmt =
this._conn.prepareStatement("INSERT INTO
user(email,username,password) VALUES(?,?,?)");

                pstmt.setString(1,user.email);
                pstmt.setString(2,user.username);
                pstmt.setString(3,user.password);
                int result = pstmt.executeUpdate();
                if (result > 0 )
                    return  true;

        }catch(SQLException ex){
            return false;
        }
        return false;
    }
    public User authenticate(String username,String password){

        try{
            PreparedStatement pstmt =
this._conn.prepareStatement("SELECT * FROM user WHERE username = ?
LIMIT
1",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
            pstmt.setString(1,username);
            ResultSet rst = pstmt.executeQuery();

            if (!rst.isBeforeFirst() ) {
                return null;
            }else{
                rst.absolute(1);
                String current_password = rst.getString("password");
                String hashed_password =
User.hashPassword(password);


                if(current_password.equals(hashed_password)){

                    return User.createInstance(rst.getInt("id"),
                            rst.getString("email"),
                            rst.getString("username"),
                            current_password);
                }

            }


        }catch (SQLException ex){
            return null;
        }
        return null;
    }
    public boolean logout(){
        return User.removeInstance();
    }

    @Override
    public User getById(int id) {
        try{
            PreparedStatement pstmt =
this._conn.prepareStatement("SELECT * FROM user WHERE id =
?",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
            pstmt.setInt(1,id);
            ResultSet rst = pstmt.executeQuery();

            if (!rst.isBeforeFirst() ) {
                return null;
            }else{
                rst.absolute(1);
                return new User(rst.getInt("id"),

rst.getString("email"),rst.getString("username"),
                        rst.getString("password"));
            }

        }catch(SQLException ex){
            return null;
        }
    }

    public boolean changeInformation(User user){
        try{

            PreparedStatement pstmt =
this._conn.prepareStatement("UPDATE user SET username = ?,password =
?,email = ? WHERE id = ?");

            pstmt.setString(1, user.username);
            pstmt.setString(2, user.password);
            pstmt.setString(3, user.email);
            pstmt.setInt(4,user.id);

            int result = pstmt.executeUpdate();
            if (result > 0)
                return true;


        }catch(SQLException ex){
            return false;
        }
        return false;
    }
}
```

2. Movie database operations

```
   public interface IMovieOperation{
       public List<Movie> getAllMovies();
       public Movie getById(int id);

   }

public class MovieOperation implements IMovieOperation{
    private final Connection _conn;

    public MovieOperation(Connection _conn){
        this._conn = _conn;
    }

    public static MovieOperation mdb = null;

    public static MovieOperation createOperation(Connection conn){
        if(mdb == null){
            mdb = new MovieOperation(conn);
        }
        return mdb;
    }

    public static  MovieOperation getOperation(){
        return mdb;
    }
    public List<Movie> getAllMovies(){
        List<Movie> mov = new ArrayList<>();
        try{

            Statement stmt = this._conn.createStatement();
            ResultSet rst = stmt.executeQuery("SELECT * FROM
movie");
            while(rst.next()) {
                mov.add(
                        new Movie(
                                rst.getInt("id"),
                                rst.getString("name"),
                                rst.getString("genre"),
                                rst.getString("description"),
                                rst.getString("released_date"),
                                rst.getFloat("price"),
                                rst.getInt("runtime"),
                                rst.getInt("rating"),
                                rst.getInt("stock"),
                                rst.getString("img_url")
                        )
                );
            }
            }catch(SQLException ex){
                return mov;
            }

    return mov;
    }

    @Override
    public Movie getById(int id) {
        try{
            PreparedStatement pstmt =
this._conn.prepareStatement("SELECT * FROM movie WHERE id =
?",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
            pstmt.setInt(1,id);
            ResultSet rst = pstmt.executeQuery();

            if (!rst.isBeforeFirst() ) {
                return null;
            }else{
                rst.absolute(1);
                return new Movie(rst.getInt("id"),

rst.getString("name"),rst.getString("genre"),
                        rst.getString("description"),
rst.getString("released_date"),
                        rst.getFloat("price"),rst.getInt("runtime"),
                        rst.getInt("rating"),rst.getInt("stock"),
                        rst.getString("img_url"));
            }

        }catch(SQLException ex){
            ex.printStackTrace();
            return null; }}}
```

3. Booking database operations

```
public interface IBookingOperation {
    public boolean createBooking(Booking newBooking);
    public List<Booking> getByUserId(int id);
    public boolean bookingTransaction(Transaction tr);
    public Booking getById(int id);
}

public class BookingOperation implements IBookingOperation{
    private final Connection _conn;

    public static BookingOperation bdb = null;
    public static BookingOperation createOperation(Connection conn){
        if(bdb == null){
            bdb = new BookingOperation(conn);
        }
        return bdb;
    }

    public static  BookingOperation getOperation(){
        return bdb;
    }

    public BookingOperation(Connection _conn){
        this._conn = _conn;
    }
    @Override
    public boolean createBooking(Booking newBooking) {
        try{
            PreparedStatement pstmt =
this._conn.prepareStatement("INSERT INTO
booking(user,movie,quantity,total_amount,date) VALUES(?,?,?,?,?)");
            pstmt.setInt(1,newBooking.user);
            pstmt.setInt(2,newBooking.movie);
            pstmt.setInt(3,newBooking.quantity);
            pstmt.setFloat(4,newBooking.total_amount);
            pstmt.setString(5, newBooking.date);
            int result = pstmt.executeUpdate();

            if (result > 0)
                return true;

        }catch(SQLException ex){
            ex.printStackTrace();
            return false;
        }
        return false;
    }

    @Override
    public List<Booking> getByUserId(int id) {
        List<Booking> bookings = new ArrayList<>();
```

```
            try{


                PreparedStatement pstmt = 
    this._conn.prepareStatement("SELECT * FROM booking WHERE user = 
    ?",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
                pstmt.setInt(1,id);
                ResultSet rst = pstmt.executeQuery();

                while(rst.next()){
                    bookings.add( new Booking(rst.getInt("id"),
                            rst.getInt("quantity"),rst.getInt("user"),
                            rst.getInt("movie"),
    rst.getFloat("total_amount"),
                            rst.getString("date"),
    rst.getBoolean("paid"))
                    );
                }

            }catch(SQLException ex){
                return bookings;
            }
            return bookings;
        }

        public Booking getById(int id) {
            try{
                PreparedStatement pstmt = 
    this._conn.prepareStatement("SELECT * FROM booking WHERE id = 
    ?",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
                pstmt.setInt(1,id);
                ResultSet rst = pstmt.executeQuery();

                if (!rst.isBeforeFirst() ) {
                    return null;
                }else{
                    rst.absolute(1);
                    return new 
    Booking(rst.getInt("id"),rst.getInt("quantity"),
                            rst.getInt("user"),rst.getInt("movie"),
                            
    rst.getFloat("total_amount"),rst.getString("date"),
                            rst.getBoolean("paid"));
                }

            }catch(SQLException ex){
                return null;
            }
        }

    public boolean bookingTransaction(Transaction tr){
        try{
            PreparedStatement pstmt = this._conn.prepareStatement("INSERT
INTO transaction(user,booking,amount,date) VALUES(?,?,?,?)");
            PreparedStatement pstmt2 = this._conn.prepareStatement("UPDATE
booking SET paid = 1 WHERE id = ? AND user = ?");
            PreparedStatement pstmt3 = this._conn.prepareStatement("SELECT
stock FROM movie WHERE id =
?",ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
            PreparedStatement pstm4 = this._conn.prepareStatement("UPDATE
movie SET quantity = ? WHERE uid = ?");

            this._conn.setAutoCommit(false);

            Booking booking = this.getById(tr.booking);

            pstmt.setInt(1,tr.user);
            pstmt.setInt(2,tr.booking);
            pstmt.setFloat(3,tr.amount);
            pstmt.setString(4,tr.date);

            pstmt2.setInt(1,tr.booking);
            pstmt2.setInt(2,tr.user);

            pstmt3.setInt(1,booking.movie);

            pstm4.setInt(2,booking.movie);

            ResultSet rst = pstmt3.executeQuery();
            rst.absolute(1);
            int stock = rst.getInt("stock");
            stock -= booking.quantity;
            pstm4.setInt(1,stock);

            int result1 = pstmt.executeUpdate();
            int result2 = pstmt2.executeUpdate();
            int result3 = pstmt.executeUpdate();

            if(result1 > 0 && result2 > 0 && result3 > 0) {
                this._conn.commit();
                return true;
            }

        }catch(Exception ex){
            ex.printStackTrace();
            try {
                this._conn.rollback();
                return false;
            }catch(SQLException exv2){
                return false;
            }
        }
        return false;
    }
}
```

GUI (Front End)

- app (main application runner / entry point)

```
public class app {
//    main application runner point
    public static void main(String[] args) throws SQLException {
        Connection _conn = DBConnection.getConnection();
        MovieOperation mdb = MovieOperation.createOperation(_conn);
        BookingOperation bdb = 
BookingOperation.createOperation(_conn);
        UserOperation udb = UserOperation.createOperation(_conn);

        new home();
    }
}
```

- home (main JFrame that holds all panel as tabbed pane)

```
public class home extends JFrame{
    JTabbedPane _jtb;
    public home(){
        _jtb = new JTabbedPane();
        _jtb.add("movies", new movieList(_jtb));


        if(User.getInstance() != null){
            _jtb.add("logout",new logout(_jtb));
            _jtb.add(User.getInstance().username, new 
account(_jtb));
        }else{
            _jtb.add("login",new login(_jtb));
            _jtb.add("register", new register(_jtb));
        }

        add(_jtb);
        setVisible(true);
        setSize(1600,1600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);
        setTitle("MovieMazza");

    }

}
```

- account (user account settings handling UI)

```
public class account extends JPanel {
    JPanel settingsPanel;
    JLabel _emailLabel;
    JTextField _email;
    JPasswordField _currentpass,_newpass;
    JLabel _currentpassLabel,_newpassLabel;
    User user;

    JButton _btnChange;
    public account(JTabbedPane jtb){
        settingsPanel = new JPanel();
        setLayout(new BorderLayout());
        user = User.getInstance();
        settingsPanel.setLayout(null);


settingsPanel.setBorder(BorderFactory.createTitledBorder("Welcome, "
+ user.username));
        _emailLabel = new JLabel("Email Address: ");
        _currentpassLabel = new JLabel("Enter Current Password");
        _newpassLabel = new JLabel("Enter new password");
        _btnChange = new JButton("change");

        _email = new JTextField(user.email);
        _currentpass = new JPasswordField();
        _newpass = new JPasswordField();

        settingsPanel.add(_emailLabel);
        settingsPanel.add(_email);
        settingsPanel.add(_currentpassLabel);
        settingsPanel.add(_currentpass);
        settingsPanel.add(_newpassLabel);
        settingsPanel.add(_newpass);
        settingsPanel.add(_btnChange);


        _emailLabel.setBounds(500,300,100,50);
        _email.setBounds(500,340,200,20);
        _currentpassLabel.setBounds(500,360,200,50);
        _currentpass.setBounds(500,400,200,20);
        _newpassLabel.setBounds(500,420,200,50);
        _newpass.setBounds(500,460,200,20);
        _btnChange.setBounds(550,520,100,30);

        _btnChange.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                if(!_email.getText().equals("") ){
                    user.email = _email.getText();
                    String current_password = new
String(_currentpass.getPassword());
```

```
                    String new_password = new
String(_newpass.getPassword());

                    if(!current_password.equals("") &&
!new_password.equals("")){
                        System.out.println(user.password + " " +
User.hashPassword(current_password));

if(User.hashPassword(current_password).equals(user.password)){
                            user.password =
User.hashPassword(new_password);
                        }else{

JOptionPane.showMessageDialog(jtb,"Password doesnot match with
current one");
                        }
                    }


if(UserOperation.getOperation().changeInformation(user)){

JOptionPane.showMessageDialog(jtb,"Information Changed");
                        _email.setText(user.email);
                    }
                }
            }
        });

        add(settingsPanel);
        setVisible(true);
    }
}
```

- login (user account authentication handling UI)

```
public class login extends JPanel{

    JLabel _userLabel;
    JLabel _passwordLabel;
    JTextField _username;
    JPasswordField _password;
     JButton _loginButton;
     JPanel _loginPanel;

     IUserOperation _udb;

    public login(JTabbedPane jtb){
        this._udb = UserOperation.getOperation();

        setLayout(new BorderLayout());
        _loginPanel = new JPanel();
_loginPanel.setBorder(BorderFactory.createTitledBorder("Welcome
Back,"));
        _loginPanel.setLayout(null);
        _userLabel = new JLabel("Username: ");
        _passwordLabel = new JLabel("Password: ");
        _username = new JTextField();
        _password = new JPasswordField();
        _loginButton = new JButton("Login");
        _loginPanel.add(_userLabel);
        _loginPanel.add(_passwordLabel);
        _loginPanel.add(_username);
        _loginPanel.add(_password);
        _loginPanel.add(_loginButton);

         add(_loginPanel,BorderLayout.CENTER);

        _userLabel.setBounds(500,200,100,50);
        _username.setBounds(500,240,200,20);
        _passwordLabel.setBounds(500,260,100,50);
        _password.setBounds(500,300,200,20);
        _loginButton.setBounds(550,340,100,30);
        add(_loginPanel);

        _loginButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                User user =
_udb.authenticate(_username.getText(),_password.getText());

                if(user != null){
                    jtb.remove(1);

JOptionPane.showMessageDialog(_loginPanel,"Logged in");
                    jtb.add("logout",new logout(jtb));
                    jtb.add(user.username, new account(jtb));
                    jtb.add("bookings", new your_bookings(jtb));
                    jtb.setSelectedIndex(0);
                }else{

JOptionPane.showMessageDialog(_loginPanel,"Username or Password not
valid");
                }
            }
        });

        setVisible(true);

    }


}
```

- logout (user account authentication revoke handling UI)

```
public class logout extends JPanel{
    JButton _logoutButton;
    IUserOperation _udb;
    public logout(JTabbedPane jtb){
        this._udb = UserOperation.getOperation();

        setLayout(new BorderLayout());
        _logoutButton = new JButton("logout");
        add(_logoutButton,BorderLayout.CENTER);
        _logoutButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                if(_udb.logout()) {
                    jtb.remove(3);
                    jtb.remove(3);

                    JOptionPane.showMessageDialog(jtb, "logged 
out");
                    jtb.setSelectedIndex(0);
                }else{
                    JOptionPane.showMessageDialog(jtb,"unable to 
logout");
                }
            }
        });

    }
}
```

- movieCard (card that holds movie information)

```
public class movieCard extends JPanel {
    JLabel _movietitle;
    JLabel _movieGenre;
    JLabel _movieRuntime;
    JLabel _moviePoster;
    JLabel _movieRating;
    JButton _buyTicket;
    String  default_poster = "";
    IBookingOperation _bdb;

    public movieCard(Movie mov, JPanel frame,JTabbedPane jtb){
        this._bdb = BookingOperation.getOperation();

        setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));
        _movietitle = new JLabel("Name:  " + mov.name);
        _movieGenre = new JLabel("Genre:  "+ mov.genre);
        _movieRuntime = new JLabel("Runtime: "  +
Integer.toString(mov.runtime));
        _movieRating = new JLabel("Rating: " +
Integer.toString(mov.rating));
        _buyTicket = new JButton("Buy Tickets");

        try {
            BufferedImage poster = ImageIO.read(new
File(mov.image_url));
            ImageIcon icon = new ImageIcon(poster);
            Image scaleImage =
icon.getImage().getScaledInstance(150, 200,Image.SCALE_DEFAULT);
            _moviePoster = new JLabel(new ImageIcon(scaleImage));
        }catch(IOException ex){
            ex.printStackTrace();
        }

        _buyTicket.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                User user = User.getInstance();
                if(user == null){
                    JOptionPane.showMessageDialog(frame,"you must be
logged in first");
                }else{
                    String quantity =
JOptionPane.showInputDialog(frame,"Number of tickets to buy");
                    try{
                        int quantity_num =
Integer.parseInt(quantity);
                        if(quantity_num == 0)
                           throw  new Exception("zero value
quantity");

                        int confirm =
JOptionPane.showConfirmDialog(frame,"Do you want to book ?");

                        if(confirm == JOptionPane.YES_OPTION) {
                            String timeStamp = new
SimpleDateFormat("yyyy-MM-dd HH-mm-ss").format(new
java.util.Date());

                            Booking newBooking = new
Booking(quantity_num, user.id, mov.id, quantity_num * mov.price,
                                    timeStamp);

                            if (_bdb.createBooking(newBooking)) {
                                JOptionPane.showMessageDialog(frame,
"booked tickets");
//                                referesh booking
                                jtb.remove(5);
                                jtb.add("bookings",new
your_bookings(jtb));
```

```
                                }else{
                                    JOptionPane.showMessageDialog(frame,
"some error occured during booking please try again");
                                }
                            }
                        }catch (Exception ex){
                            ex.printStackTrace();
                            JOptionPane.showMessageDialog(frame,"provide
appropriate quantity");
                        }

                    }
                }
            });
            GridBagConstraints gbc = new GridBagConstraints();
            add(_moviePoster);
            add(_movietitle);
            add(_movieGenre);
            add(_movieRuntime);
            add(_movieRating);
            add(_buyTicket);
            setVisible(true);
            setBorder(BorderFactory.createTitledBorder(mov.name));


        }
}
```

- movieList(component that holds list of movie cards)

```
public class movieList extends JPanel{

    JPanel moviePanel;
    JScrollPane scrollPane;
    IMovieOperation _mdb;
    IBookingOperation _bdb;

    public movieList(JTabbedPane jtb){
        this._bdb = BookingOperation.getOperation();
        this._mdb = MovieOperation.getOperation();

        setLayout(new BorderLayout());

        moviePanel = new JPanel();
        moviePanel.setLayout(new GridLayout(0,3,2,2));
        moviePanel.setBorder(BorderFactory.createTitledBorder("Newly
Released"));

        this._mdb.getAllMovies().forEach(movie -> {
            moviePanel.add(new movieCard(movie,this,jtb));
        });
        scrollPane = new JScrollPane(moviePanel);

scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROL
LBAR_AS_NEEDED);

scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR
_AS_NEEDED);

        add(scrollPane);
        setVisible(true);

    }

}
```

- userbookings(component that holds customer bookings informations)

```
public class userbookings extends JPanel{
    JLabel name;
    JLabel total_amount;
    JLabel date;
    JButton pay;
    JLabel paid;
    public userbookings(Booking booking,IMovieOperation
_mdb,JTabbedPane jtb){
        setLayout(new GridLayout(0,5,12,8));
        setBorder(BorderFactory.createCompoundBorder(new
EmptyBorder(10, 10, 10, 10),  new EtchedBorder()));
        Movie movie = _mdb.getById(booking.movie);
        User user = User.getInstance();

        if (movie == null || user == null){
            JOptionPane.showMessageDialog(jtb,"Some error
occurred");
            jtb.setSelectedIndex(0);
        }
        name = new JLabel(movie.name);
        total_amount = new JLabel("Total:
"+Float.toString(booking.total_amount));
        date = new JLabel("Booked at: "+ booking.date);

        add(name);
        add(total_amount);
        add(date);
        if(booking.paid){
            paid = new JLabel("paid");
            add(paid);
        }else{
            pay = new JButton("pay");
            add(pay);

            pay.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    String timeStamp = new SimpleDateFormat("yyyy-
MM-dd HH-mm-ss").format(new java.util.Date());
                    String payment_msg = "Do you want to pay ?
Amount: " + Float.toString(booking.total_amount) + " Enter cvc:";
                    var cvc = JOptionPane.showInputDialog(jtb,
payment_msg);

                    if (!cvc.isBlank() || !cvc.isEmpty()) {
                        var pin = JOptionPane.showInputDialog(jtb,
"Enter pin");
                        if (pin.length() == 4) {
                            Transaction tr = new
Transaction(user.id, booking.id,
                                    booking.total_amount,
timeStamp);

                            if
(BookingOperation.getOperation().bookingTransaction(tr)) {
                                JOptionPane.showMessageDialog(jtb,
"Payment Complete");
                            } else {
                                JOptionPane.showMessageDialog(jtb,
"Payment Failed");
                            }

                        } else {
                            JOptionPane.showMessageDialog(jtb,
"enter valid pin");
                        }
                        jtb.remove(5);
                        jtb.add("bookings", new your_bookings(jtb));
                        jtb.setSelectedIndex(5);

                    } else {
                        JOptionPane.showMessageDialog(jtb, "Enter
valid cvc number");
                    }
                }
            });
        }

        setVisible(true);
    }
}
```

- your_bookings(component that holds list of customer bookings)

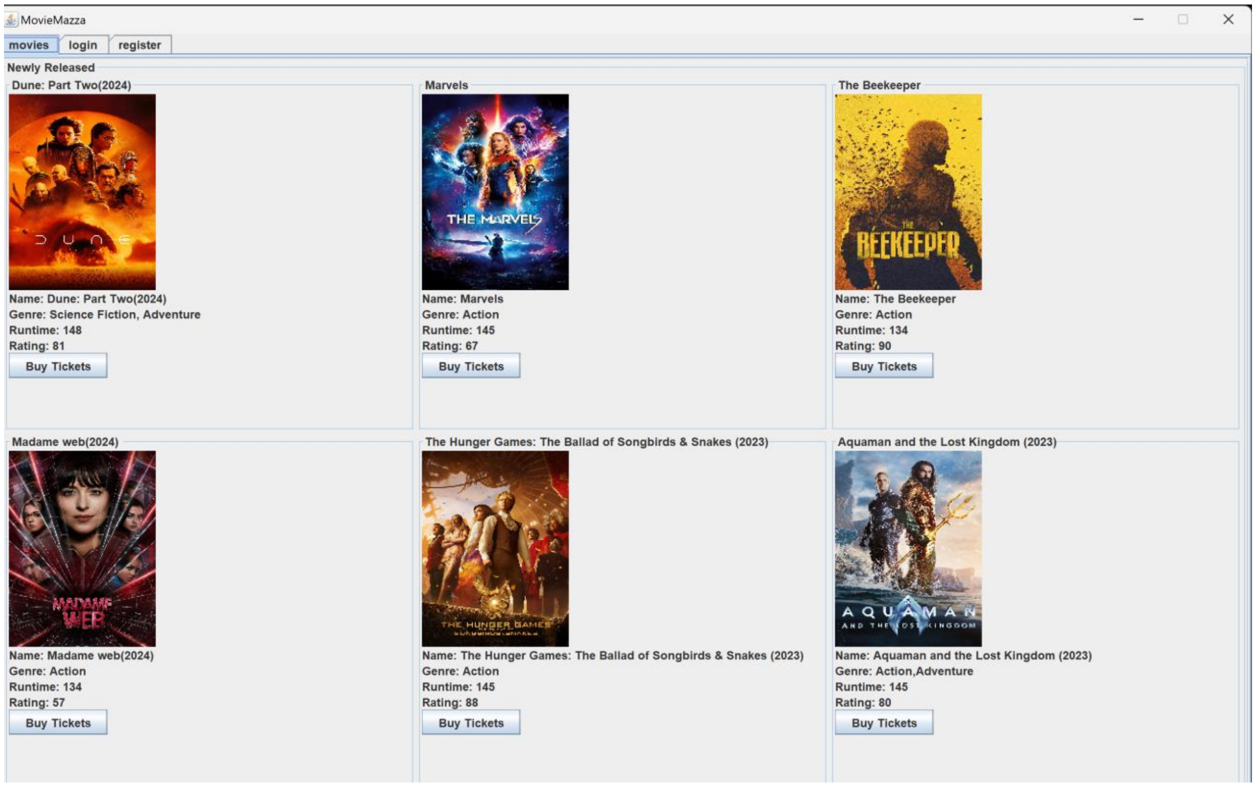
```
public class your_bookings extends JPanel{

    JPanel _bookingPanel;
    IBookingOperation _bdb;
    IMovieOperation _mdb;
    JTable table;
    DefaultTableModel _model;
    JScrollPane scrollPane;
    JPanel topPanel;

    public your_bookings(  JTabbedPane jtb) {
        setLayout(new BorderLayout());
        this._bdb = BookingOperation.getOperation();
        this._mdb = MovieOperation.getOperation();

        _bookingPanel = new JPanel();
        scrollPane = new JScrollPane(_bookingPanel, 
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED, 
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        List<Booking> bookings = 
this._bdb.getByUserId(User.getInstance().id);

        if (bookings.isEmpty()) {
            setBorder(BorderFactory.createTitledBorder("No 
Bookings"));
        } else {
            setBorder(BorderFactory.createTitledBorder("Your 
Bookings"));
            bookings.forEach(booking -> {
                _bookingPanel.add(new userbookings(booking, 
this._mdb, jtb));
            });
        }
        add(scrollPane, BorderLayout.CENTER);

    }
}
```

- register(component for registering users)

```
public class register extends JPanel{
    JLabel _userLabel;
    JLabel _passwordLabel;
    JTextField _username;
    JPasswordField _password;
    JLabel _emailLabel;
    JTextField _email;
    JButton _registerButton;
    JPanel _registerPanel;
    IUserOperation _udb;
    public register(JTabbedPane jtb){
        this._udb = UserOperation.getOperation();

        setLayout(new BorderLayout());
        _registerPanel = new JPanel();

_registerPanel.setBorder(BorderFactory.createTitledBorder("Welcome
Back,"));
        _registerPanel.setLayout(null);
        _userLabel = new JLabel("Username: ");
        _passwordLabel = new JLabel("Password: ");
        _username = new JTextField();
        _password = new JPasswordField();
        _registerButton = new JButton("Login");
        _emailLabel = new JLabel("Email Address: ");
        _email = new JTextField();
        _registerPanel.add(_userLabel);
        _registerPanel.add(_passwordLabel);
        _registerPanel.add(_emailLabel);
        _registerPanel.add(_email);
        _registerPanel.add(_username);
        _registerPanel.add(_password);
        _registerPanel.add(_registerButton);

        add(_registerPanel,BorderLayout.CENTER);

        _userLabel.setBounds(500,200,100,50);
        _username.setBounds(500,240,200,20);
        _emailLabel.setBounds(500,260,100,50);
        _email.setBounds(500,300,200,20);
        _passwordLabel.setBounds(500,320,100,50);
        _password.setBounds(500,360,200,20);
        _registerButton.setBounds(550,420,100,30);
        add(_registerPanel);

        _registerButton.addMouseListener(new MouseAdapter() {
            @Override

            public void mouseClicked(MouseEvent e) {

                if (_email.getText().equals("") ||
_username.getText().equals("") || new
String(_password.getPassword()).equals("")) {

JOptionPane.showMessageDialog(e.getComponent(),"Field cannot be
empty");

                } else {

                    boolean ok = _udb.register(new
User(_email.getText(), _username.getText(), new
String(_password.getPassword())));
                    if (ok) {
                        User.removeInstance();
                        JOptionPane.showMessageDialog(jtb, "Now a 
registered user");
                        jtb.setSelectedIndex(0);

                    } else {
                        JOptionPane.showMessageDialog(jtb, "Could 
not register check everything");
                    }
                }
            }
        });

        setVisible(true);
    }
}
```

Screenshots

Home



Dune: Part Two(2024)
Name: Dune: Part Two(2024)
Genre: Science Fiction, Adventure
Runtime: 148
Rating: 81
Buy Tickets

Marvels
Name: Marvels
Genre: Action
Runtime: 145
Rating: 67
Buy Tickets

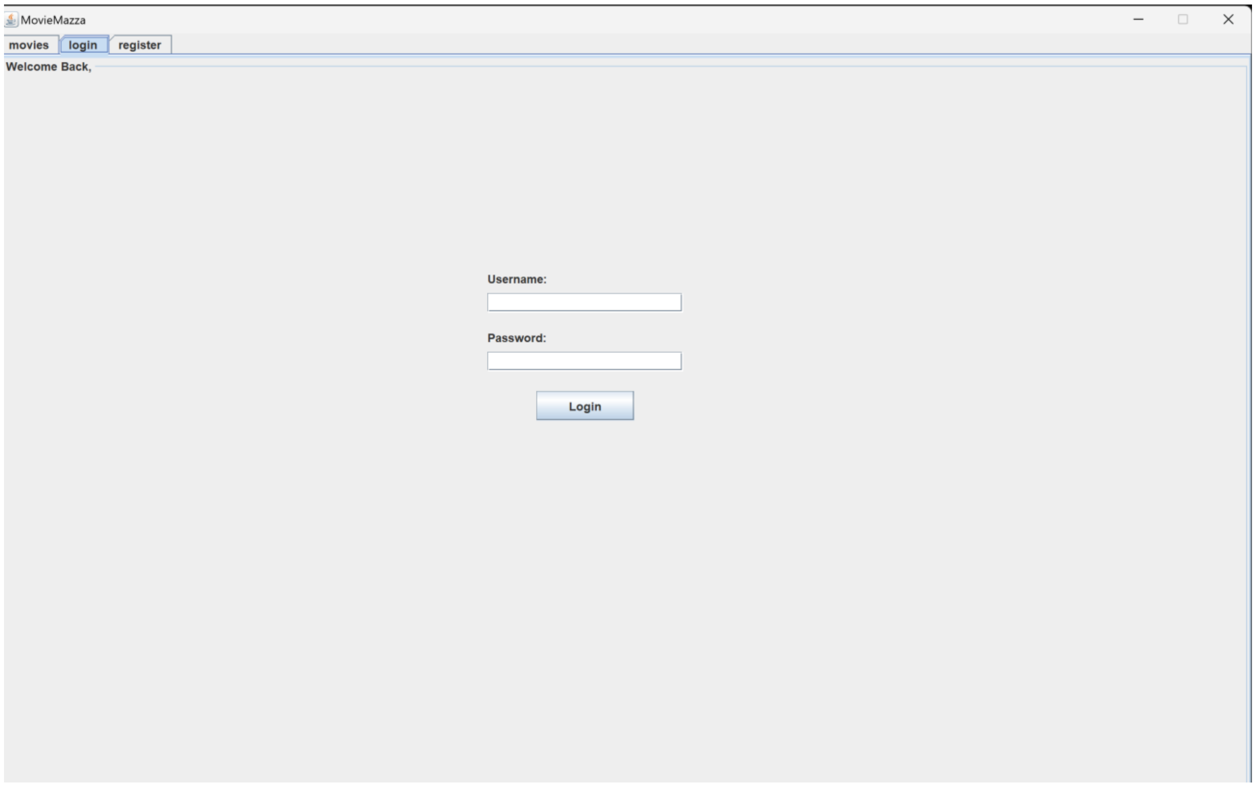
The Beekeeper
Name: The Beekeeper
Genre: Action
Runtime: 134
Rating: 90
Buy Tickets

Madame web(2024)
Name: Madame web(2024)
Genre: Action
Runtime: 134
Rating: 57
Buy Tickets

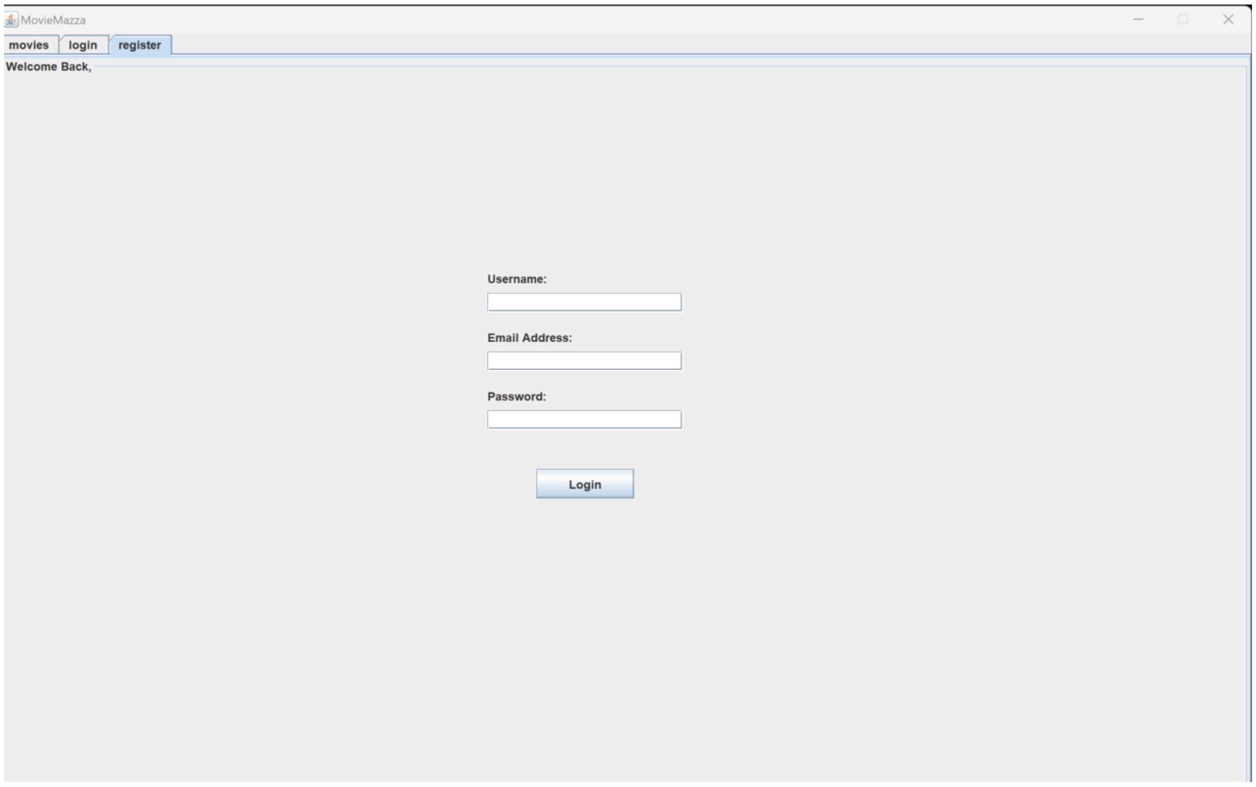
The Hunger Games: The Ballad of Songbirds & Snakes (2023)
Name: The Hunger Games: The Ballad of Songbirds & Snakes (2023)
Genre: Action
Runtime: 145
Rating: 88
Buy Tickets

Aquaman and the Lost Kingdom (2023)
Name: Aquaman and the Lost Kingdom (2023)
Genre: Action,Adventure
Runtime: 145
Rating: 80
Buy Tickets

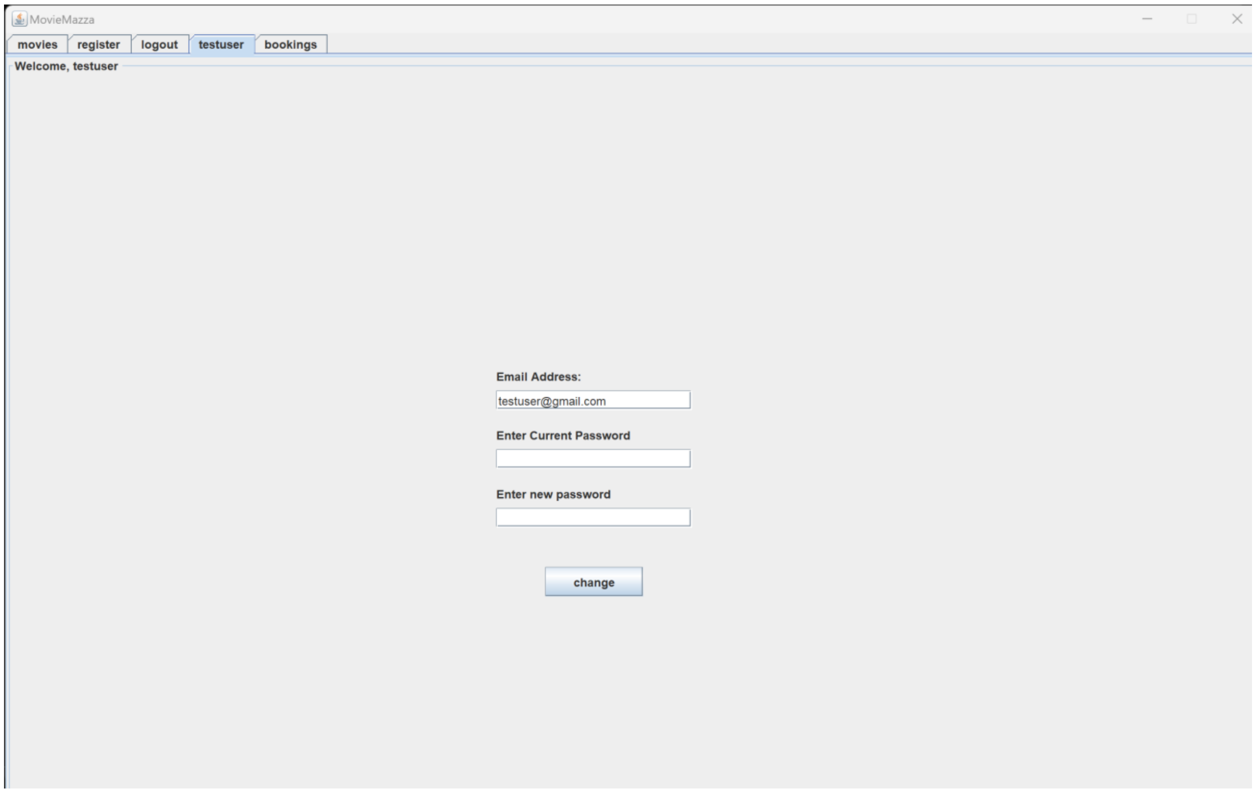
Login



Register



Account Settings



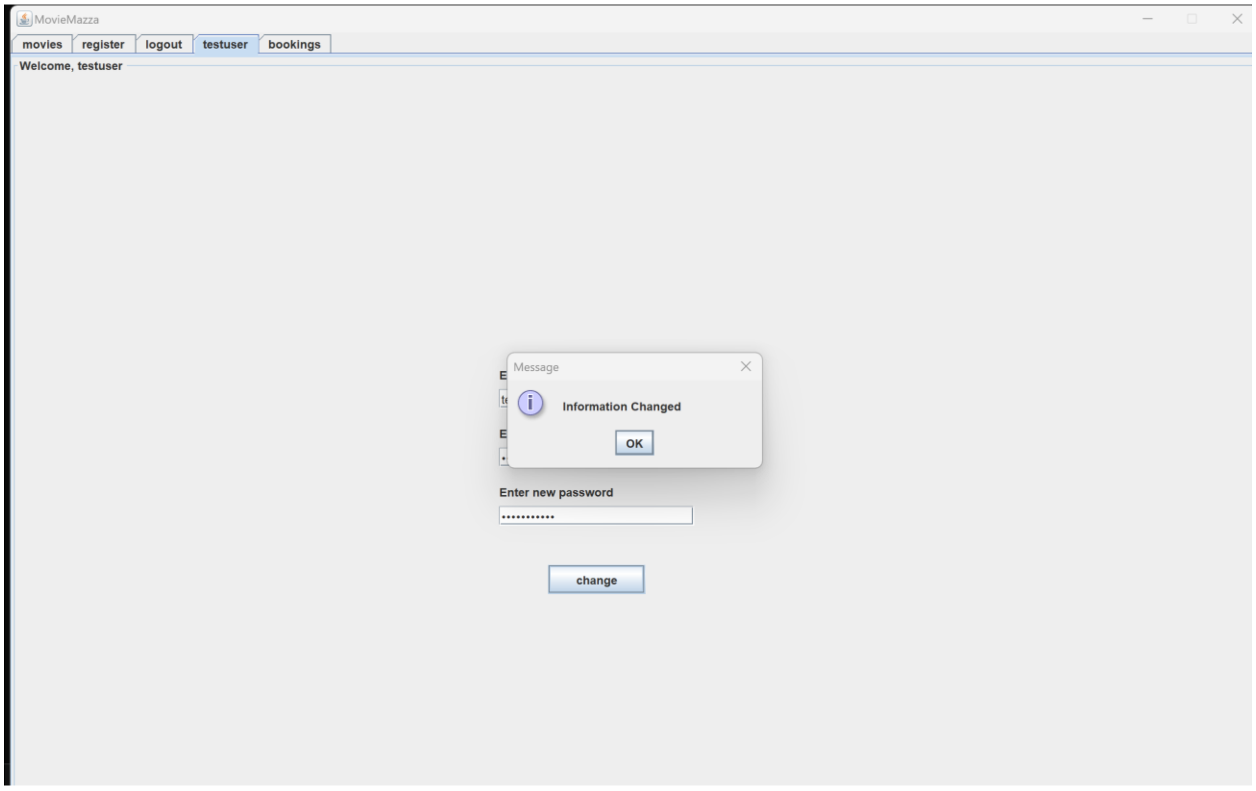
Welcome, testuser

Email Address:

testuser@gmail.com

Enter Current Password

Enter new password



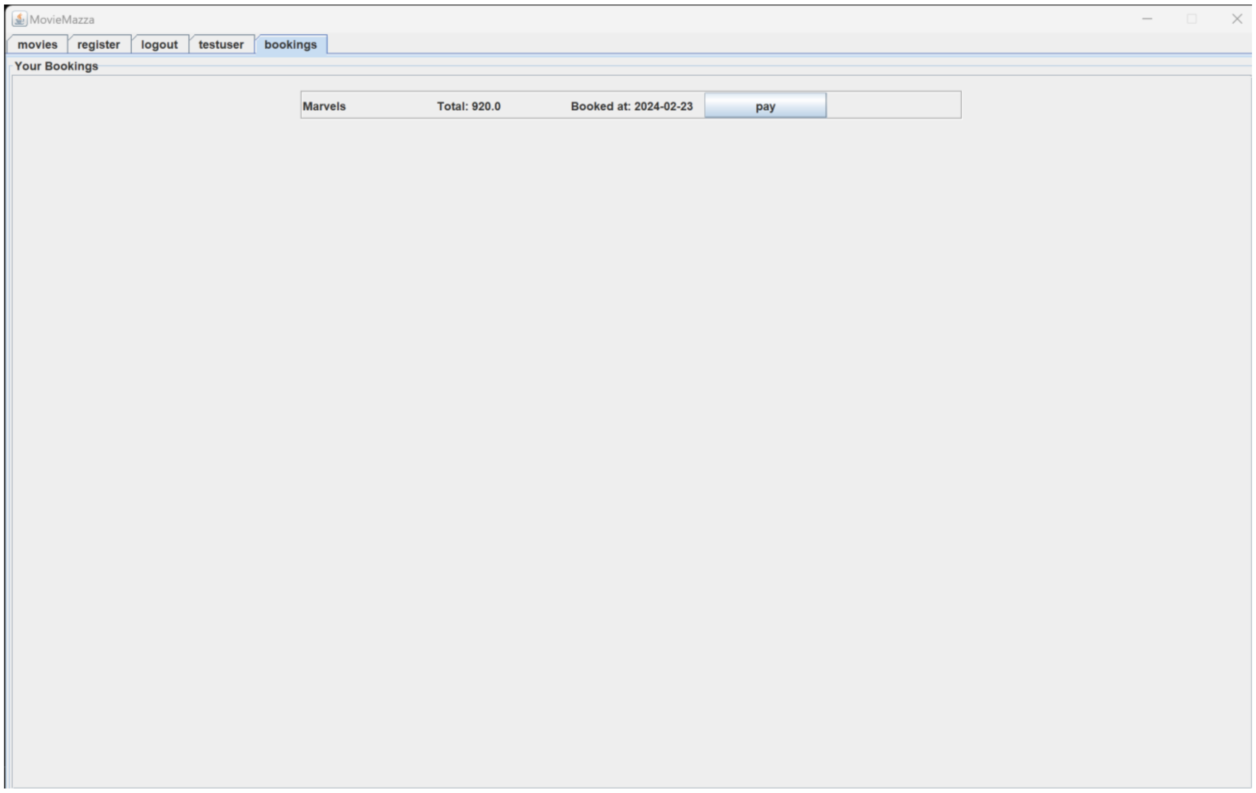
Welcome, testuser

Information Changed

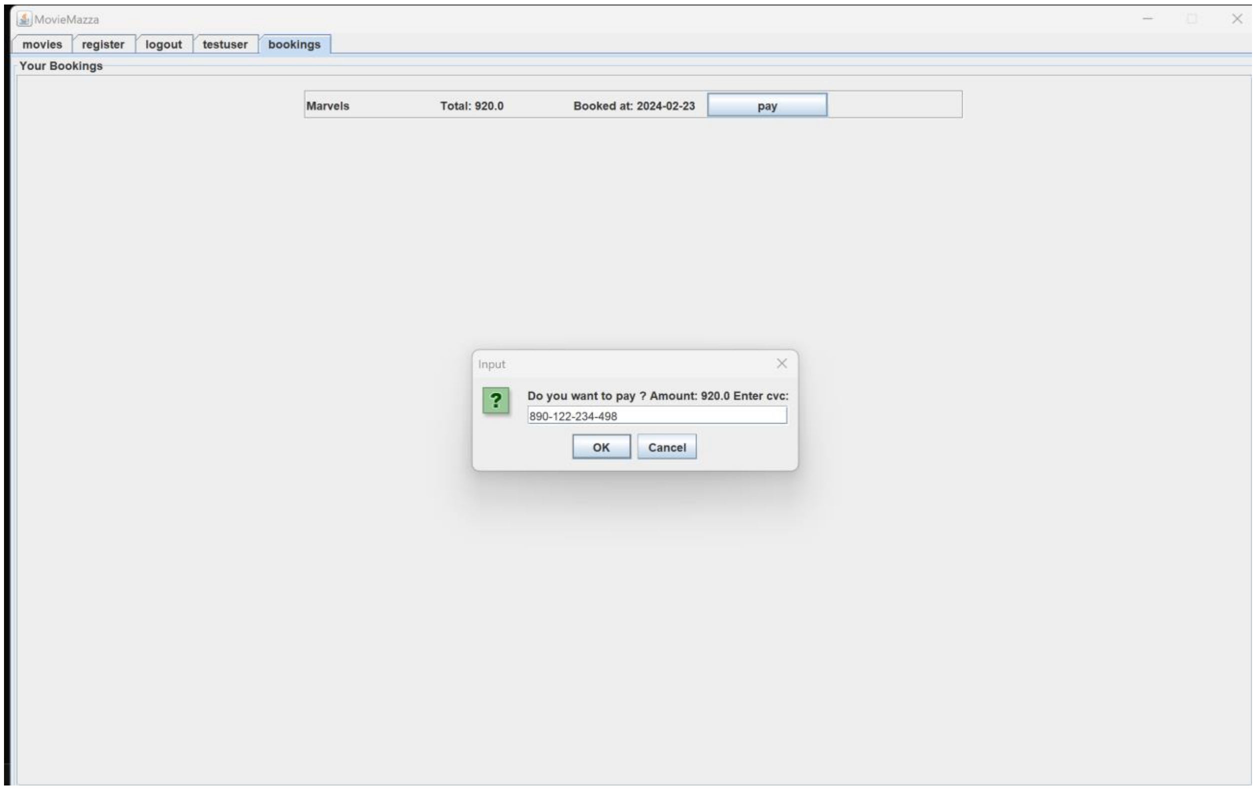
Enter new password

•••••••••••

Bookings



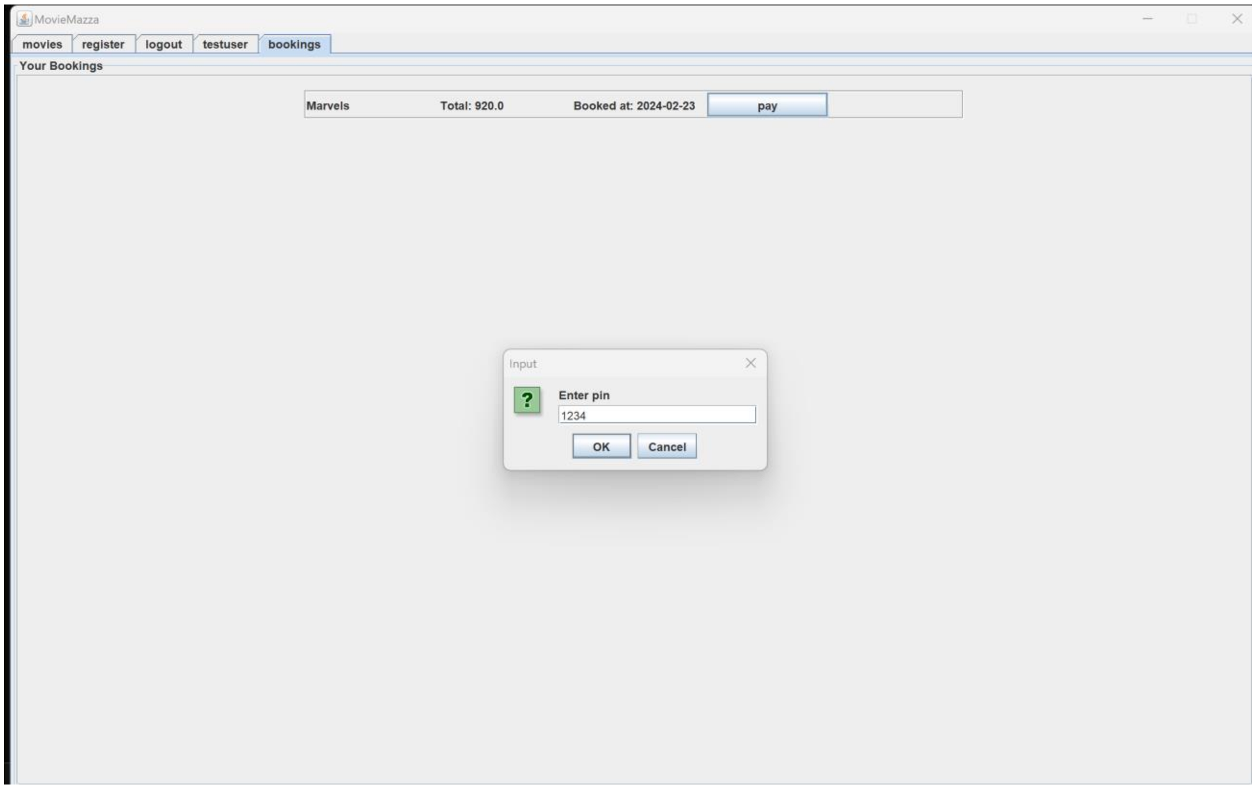
Marvels Total: 920.0 Booked at: 2024-02-23

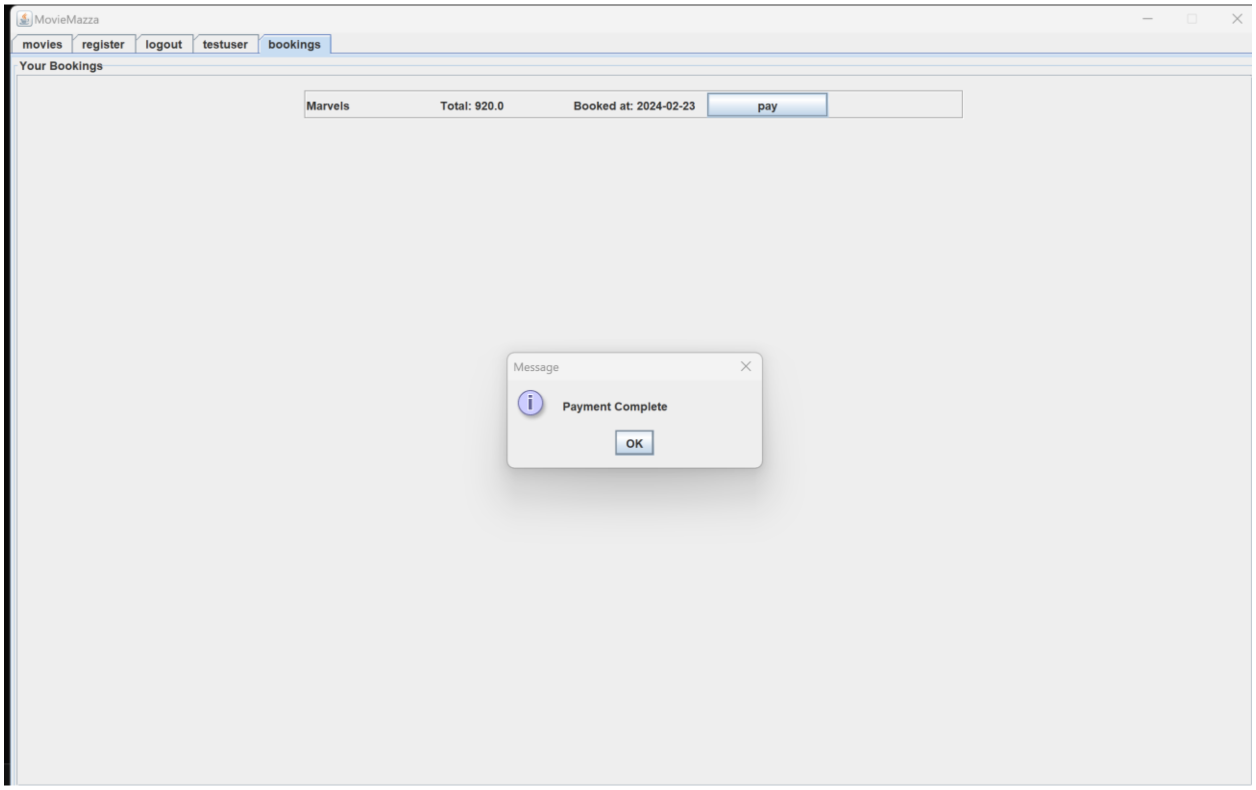


Marvels Total: 920.0 Booked at: 2024-02-23

Do you want to pay ? Amount: 920.0 Enter cvc:

890-122-234-498


MovieMazza
movies
register
logout
testuser
bookings
Your Bookings
Marvels
Total: 920.0
Booked at: 2024-02-23
pay
Input
Enter pin
1234
OK
Cancel


MovieMazza
movies
register
logout
testuser
bookings
Your Bookings
Marvels
Total: 920.0
Booked at: 2024-02-23
pay
Message
Payment Complete
OK