PROJECT PHASE II

Team Name: Easy Doctor

Team: 37

Team members:

Johnny Salazar, ZhiYue Wang, XiuQi Yang, William Whittaker, Colton Hutchins

Use Cases and the Use Case Diagram

Actors:

Patient: Patient uses the system to access patient-specific functions such as scheduling visits, viewing visits, managing personal information, and managing prescriptions.

Doctor/nurse: Doctors/nurses use the system to manage work-related tasks such as viewing and managing visits, patient records, activity sessions, and managing prescriptions.

Use cases:

Sign in Page

Actor: Patient or Doctor/nurse

Basic flow:

The user enters their username and password into the provided text boxes.

The user decides whether to select the “Remember Me” checkbox to stay logged in for future sessions.

If the user is a doctor or nurse, they select the “I am a doctor/nurse” checkbox to indicate their role.

The user clicks the “Log In” submit button to authenticate.

If credentials are incorrect, an alert displays the text “Incorrect Username or Password.”

Data:

Username, Password, Remember Me, I am a doctor/nurse.

Stimulus:

User submits login credentials.

Condition: User selects “I am a doctor/nurse” if they belong to that category.

Response:

The user is directed to the correct portal (Work Portal for doctors/nurses, Patient Portal for patients) or an alert is shown if the username or password is incorrect.

Sign Up Page:

Actor: Patient

Basic flow:

The prospective patient selects the "Sign Up" option from the main page.

The patient fills in all the required fields: first name, last name, mother's name, father's name, date of birth, phone number, and address.

The patient inputs optional information: pharmacy and preferred doctor.

The patient chooses a password and enters it into both the password and confirm password fields.

The patient uploads an optional profile picture.

The patient reviews all information and either selects "Save" to submit the information or "Cancel" to abort the sign-up process.

Data:

Personal Information: First name, last name, mother's name, father's name, date of birth, phone number, address.

Optional Information: Email address, pharmacy information, preferred doctor.

Security Information: Password and password confirmation.

Profile Picture: Image file for the profile picture.

Stimulus:

Patient's desire to register on the portal to access medical services and manage their healthcare online.

Response:

Successful Registration: The system confirms that all required fields are filled in correctly, the passwords match, and the email is not already in use. The patient is registered, and an account is created.

Error: If there is missing or incorrect information, or if the passwords do not match, the system displays an appropriate error message. If the patient already exists, an alert notifies them of this.

Forget My Password Page

Actor: Patient or Doctor/Nurse

Basic flow:

The user navigates to the "Forgot My Password" page.

The user enters their username or email into the provided field.

If the user is a doctor or nurse, they select the “I am a Doctor” checkbox to indicate their role.

The user clicks the "Retrieve Password" button to initiate the password retrieval process.

Data:

Username/Email: The input provided by the user to identify their account.

Role Confirmation: Boolean value captured by the “I am a Doctor” checkbox.

Stimulus:

Users need to recover access to their account due to a forgotten password.

Response:

If the account is found, the system sends an email with password retrieval instructions to the user’s email address, and the user is notified of the email sent.

If there is no account associated with the provided username/email, the system alerts the user that the account could not be found and highlights the field in red.

Portal Main page

Actor: Patient

Basic flow:

The patient logs into the portal and is directed to the main page.

The patient is greeted with a welcome message.

The patient has the option to navigate to different sections of the portal: "My Visits," "My Info," "Schedule," "Inbox," "My Pills."

The patient can select "Submit Feedback" to provide feedback to the system administrators.

The patient can log out of the portal using the "LogOut" button.

Data:

The patient's choice of tabs to navigate through the portal.

The information entered by the patient in the feedback popup text area.

Stimulus:

The need to access healthcare-related information or services prompts the patient to interact with the various tabs and options.

Response:

When the patient selects a tab, the corresponding subpage is loaded.

Upon selecting the “Submit Feedback” button, a popup appears for the patient to enter and send their feedback.

When the patient clicks the “LogOut” button, they are signed out and redirected to the Login Page.

Portal Inbox Tab

Actor:

Patient

Basic flow:

The patient clicks on the "Inbox" tab to view their messages.

The patient browses through the list of messages, which are displayed in chronological order within a scroll pane.

The patient can click on any message to view its content in a popup.

If the message is received, a "Reply" button is available to respond to the message.

The patient can click the "New Message" button to compose a new message.

Upon composing a message, the patient can send it or cancel the action.

Data:

Message List, Message Content, Reply Content

Stimulus:

The user's need to view and manage communication with doctors or the healthcare system through the portal.

Response:

When a message is clicked, its content is shown in a popup window, where received messages can be replied to, and sent messages can be reviewed.

A "New Message" button allows composing a new message with a character limit.

Sending a message or replying to a message triggers a process to deliver this communication to the intended recipient.

Patient Portal My Visits Tab

Actor:

Patient

Basic flow:

The patient navigates to the "My Visits" tab within the portal.

The system displays a list of visits, including completed, missed, and upcoming appointments, organized chronologically.

Each visit entry shows relevant details such as the doctor's name, day, time, and a brief description.

The patient clicks on a visit entry to view more details.

For completed visits, a popup would appear containing detailed information, including the doctor, nurse, reason for the visit, scheduled time, and visit summary.

Data:

Visit Information: Data pertaining to each visit, including the healthcare provider's name, appointment day and time, and the reason for the visit.

Stimulus:

The patients need to review their past and upcoming appointments.

Response:

On selection of a visit entry, the system presents additional details, potentially through a popup for completed visits.

The patient can interact with the popup by clicking a "Done" button to acknowledge the information and close the popup.

Patient Portal My Information Tab

Actor:

Patient

Basic flow:

The patient navigates to the "My Info" tab.

The portal displays the patient's information as retrieved from the database.

The patient reviews their information, which includes editable and non-editable fields.

To update any editable information, the patient clicks the "Edit" button.

After making changes, the patient clicks “Save Changes” to update their information in the database or “Cancel” to discard changes.

Data:

Editable Fields: Phone number, address, pharmacy list, preferred doctor, insurance number.

Non-Editable Fields: Username, email, first name, last name, mother’s name, father’s name, date of birth, prescriptions.

Stimulus:

The patient needs to review or update their personal and medical information.

Response:

Upon clicking “Edit,” editable fields become active, allowing the patient to input new information.

Clicking “Save Changes” saves the new data to the database and updates the patient’s profile.

Clicking “Cancel” discards any unsaved changes and reverts the information to its previous state.

Patient Portal My Prescription Tab

Actor:

Patient

Basic flow:

The patient navigates to the "My Prescriptions" tab within the portal.

The system displays all the patient's current prescriptions along with the designated pharmacy.

The patient can review the details of their prescriptions, such as medication names, schedule, and purpose.

If the patient decides to change their pharmacy, they can click the "Edit" button to modify the pharmacy list.

After making changes, the patient can save the updated information by clicking the "Save Changes" button.

If the patient changes their mind or makes a mistake, they can click the "Cancel" button to revert any changes.

Data:

Prescriptions List: Medication names, the days they are taken, times for taking them, and the purpose of the medication.

Pharmacy Information: Name and details of the patient’s preferred pharmacy.

Stimulus:

The patient's need to review their medication schedule or to update their preferred pharmacy details.

Response:

The portal provides an editable view of the pharmacy information, which the patient can update and save or cancel as needed.

Work Portal Visits Tab

Actor:

Doctor/Nurse

Basic flow:

The healthcare professional logs into the Work Portal and selects the "Visits" tab.

The portal displays a list of upcoming patient visits for the current day, organized in chronological order.

The list shows the patient's name and a brief description, possibly indicating the need for action, such as "Reschedule."

The healthcare professional can click on a patient's visit to view more details.

A popup with additional information about the patient and visit appears upon selection.

For the first pending visit, there is a "Start Session" button, allowing the healthcare professional to initiate the visit.

Clicking "Start Session" prompts a confirmation message. If confirmed, the visit's status updates to "active."

Data:

Upcoming Visits Information: Patient names, days and times of visits, and short descriptions for each appointment.

Stimulus:

The healthcare professional's need to review the day's schedule and manage patient visits.

Response:

The portal updates to show visit statuses, with color-coding indicating different statuses such as pending, active, missed, or complete.

A "Start Session" button allows the professional to begin the visit process.

Work Portal Patient Records Tab

Actor:

Doctor/Nurse

Basic flow:

The healthcare professional navigates to the "Info" or "Patient Records" tab within the Work Portal.

They are presented with a search interface to look up a client by entering the required fields: first name, last name, date of birth, and email.

After entering the search criteria, the healthcare professional clicks a "Lookup" button to retrieve the patient's records.

The system queries the database and, if the patient is found, displays their medical information in a scrollable popup window.

If no record is found, the system alerts the healthcare professional that the patient is not in the database.

Data:

Patient Search Criteria: First name, last name, date of birth, and email.

Patient Record Information: Includes first name, last name, age, previous health issues, previously prescribed medications, history of immunization, current prescriptions, pharmacy list, and insurance number.

Stimulus:

Doctors/nurses need to access a patient's medical history and records.

Response:

On a successful search, the portal displays the patient's detailed information for review or updates.

On an unsuccessful search, the system notifies the professional that there is no matching patient record.

Work Portal Active Visits Tab

Actor:

Doctor/nurse

Basic flow:

The healthcare professional assesses the "Active Visits" tab in the work portal.

The portal presents a list of active visits, which are displayed in chronological order.

Each visit entry contains the doctor’s name, patient’s name, appointment time, and a brief description.

The healthcare professional clicks on a visit entry, and a popup appears.

This popup contains fields for inputting or viewing various patient details, which differ based on the patient's age and the visit's requirements.

Upon completing the necessary fields and concluding the visit, the professional can click “End Session” to finalize the visit.

If the visit needs to be returned to a pending status, the “Make Pending” button can be selected, prompting a confirmation before the action is executed.

Data:

Information about the doctor, patient, visit time, and a description of the issue or reason for the visit.

Stimulus:

Doctors/nurses need to manage the workflow of patient visits that are currently active.

Response:

The system displays detailed visit information upon interaction and allows the status of the visit to be updated to either "complete" or reverted to "pending."

Work Portal Prescription Tool Tab

Actor:

Doctor/nurse

Basic flow:

The healthcare professional assesses the "Prescription Tool" tab.

A search field is available to input the patient's first name, last name, and date of birth to locate their prescription records.

The current patient's name and associated pharmacy information are displayed.

The professional can see a list of the patient’s prescriptions, including details such as the medication name, days of intake, times, and descriptions.

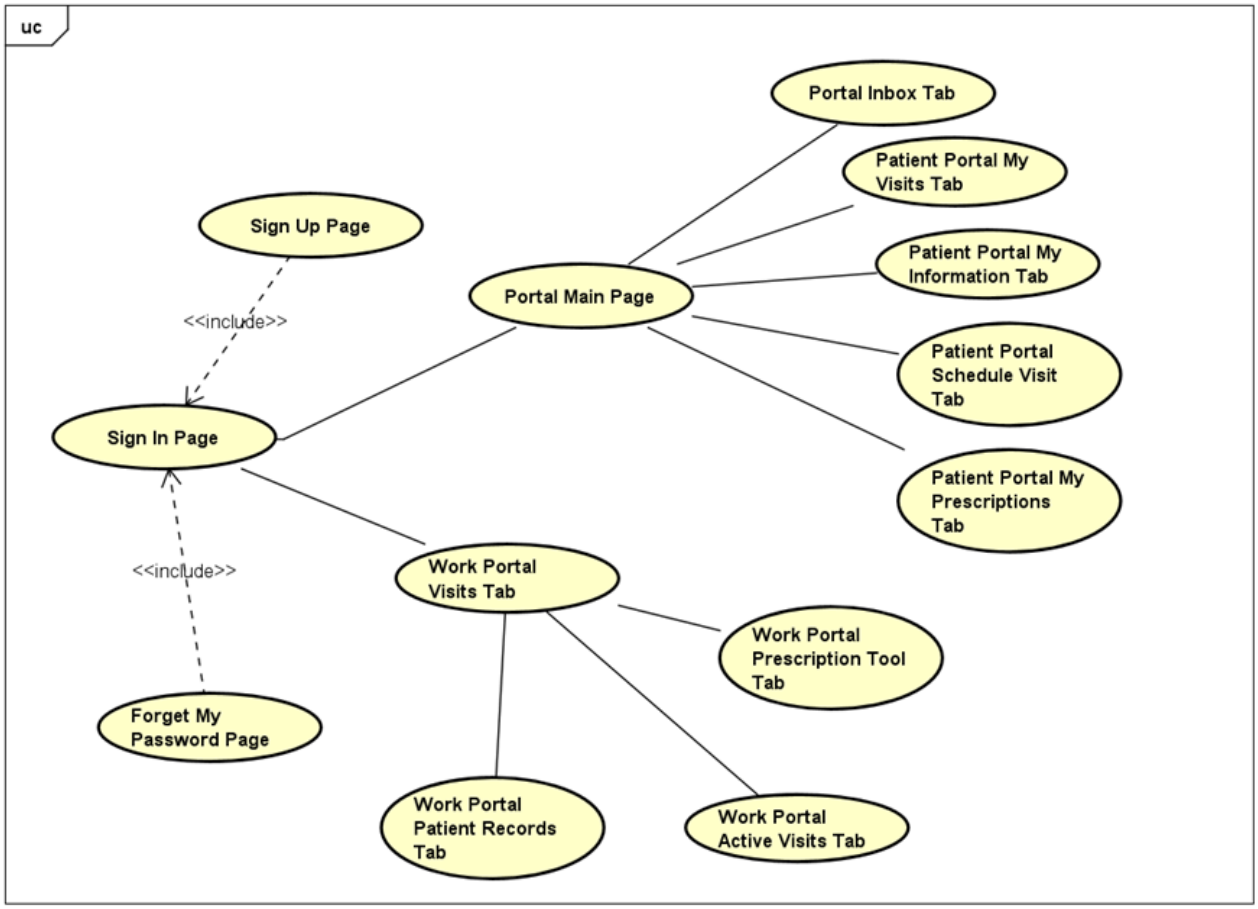
The professional can edit the prescription information by clicking on the "Edit" button.

After making changes, the professional can save the updates with the "Save Changes" button or cancel any modifications with the "Cancel" button.

Data:

Patient Information: Used for searching and identifying the correct patient record.

Prescription Details: The medications prescribed to the patient, including the schedule and purpose.



either save th

Object Identification

App (Main):

The App class is responsible for connecting to the database as “NEUTRAL” and loading the SignIn page. Loading pages and popups is done by the App class. When the user wishes to close the application or force quits, the App class makes sure to sign out the user (if signed in) and disconnects from the database. It also contains the main method to start the application.

Database:

The Database class is responsible for establishing the connection between the database and the application. The Database will also be used to create Table objects that represent each table in the database, as well as managing the currently logged in user and the Encryption class.

Table:

The Table class is responsible for retrieving, updating, storing and deleting Rows from the database. It will use the Connection (from an external library) provided by the Database to facilitate this.

Row:

The Row class is an abstract class that serves as the base for several implementing classes. It contains key data, like the ID of the row, the database and table it belongs to, as well as several utility methods that simplify saving updates, as well as facilitating the connection between the Row class and Datum class.

User, Conversation, Allergy, VaccineRecord, Prescription, Drug, Logbook, Surgery, Employee, Patient, HealthCondition:

These classes store table-specific data. They provide getters and setters for this data, as well as override methods from Row in order to facilitate connections to the Datum class. They also provide a constructor, used for creating new entries. The User and Conversation classes communicate with the Database’s Encryption class to hash passwords and encrypt messages.

Datum:

The Datum class is responsible for containing the original value from the database, and the new value if the user changes it. It can be used to create a ValueField object which is a text field that contains the original value and has the option to be updatable. For example, when retrieving a row from the vaccineRecords table, Datum objects for the creationTime, patientID, vaccineGroup, date, provider, and notes will be created, each which can then be turned into a ValueField. The Datum objects remember what table and row they came from to make it easy to update the database.

ValueField:

The ValueField class is an extension of JavaFX's TextField class. Its main responsibilities are to display the value of the Datum object it holds, and to allow itself to be edited and updatable depending on the current user's permissions. For example, when ValueField objects of a patient's health conditions are displayed, since the doctor has permissions, they will be allowed to click on edit, enter in a new value, and click save. This will update the patient's health conditions on the database.

UpdateButtonGroup:

The UpdateButtonGroup class is a JavaFX GridPane and consists of an edit button, a cancel button, and a save button. ValueField objects can connect to an UpdateButtonGroup which will then respond when one of these buttons are clicked.

TableViewBuilder:

The TableViewBuilder class makes it easy to create TableView objects that will be displayed (UI component). The programmer has the option to display the table name, header, make it scrollable, and have the row perform an action when clicked on.

TableView:

The TableView class is an extension of JavaFX's GridPane that contains Rows which are made of Datum objects. The table can be stylized and have functionality added to it via the TableViewBuilder.

FormBuilder:

The FormBuilder class makes it easy to create Form objects and adds functionality to the ValueFields inside the form. If a name for the Form is provided, it will be displayed.

Form:

The FormBuilder class is an extension of JavaFX's GridPane that contains ValueFields and Labels. When a piece of data is retrieved from the database, the Label will display the column name and the ValueField will display the value. ValueFields can be editable so long as the user has permissions.

WelcomeController:

The WelcomeController is responsible for displaying two buttons: Sign In or Sign Up. Then the appropriate page will be loaded.

SignInController:

The SignInController is responsible for validating the username and password entered by the user and signing the user in if there is a match.

SignUpController:

The SignUpController is responsible for validating the sign up form filled out by the user. Once all the required fields are entered, the information will get passed onto the Database's methods to determine if an account could be made or not.

MyVisitPopupController:

The MyVisitPopController is responsible for visually loading details of the patient's visit. It allows the patient to reschedule or cancel by using Database's methods.

VisitPopupController:

The VisitPopController is responsible for visually loading the visit details of a patient's visit. It allows the user to start a visit and to reschedule the visit using Database's methods.

MyPillPopupController:

The MyPillPopupController is responsible for visually loading the details of a patient's prescription. It allows the patient to send a message to their doctor.

ForgotUsernamePasswordController:

The ForgotUsernamePasswordController is responsible for allowing the user to retrieve their password by entering their email. It uses Database's methods to validate the email and send an email to the user with their password.

PatientPortalController:

The PatientPortalController is responsible for displaying information about the patient such as their visits, their medical information, their chat with their doctor, etc.

WorkPortalController:

The PatientPortalController is responsible for displaying information about the clinic, such as upcoming visits, medical records for patients, messages from patients, etc.

CRC Diagram

Database	
• Manages the connection between the application and the MySQL database • Contains basic utility methods for inserting, updating, finding and removing data from the database • Creates Table objects to be used by the application • Stores basic user information	• SignInController • PatientPortalController • WorkPortalController • PopupController • Table • Encryption

Encryption	
• Provides methods for hashing and verifying passwords • Allows for encryption of data using a public and private key	• Database

Table	
• Represents a table from the mysql database • Helps retrieve multiple rows from the database • Creates rows from retrieved ResultSets from the Database class	• Datum • Database • ValueField • Value • Row (and it's implementations) • ResultSet

Row (& implementations)	
• Represents a Row from the MySQL database • Allows for easy insertion and updating • Provides methods for easy integration with the interface • Stores key data, extended by new classes for each specific table in the database schema to contain the table-specific data	• Table • ResultSet • Datum

App	
• Connects to mysql server as a neutral user • Loads FXML views and Controllers • When application is closed, the mysql connection is closed and user is signed out	• Database • Controller (interface)

TableViewBuilder	
• Builds tables containing information from the database with limited customization options • Allows the rows of a table to have event listeners • Uses the Label class to display the values	• TableView • TableData • Database

FormBuilder	
• Builds forms containing information from the database with limited customization options • Allows the rows of a form to have event listeners • Uses ValueField objects to create the form	• Form • FormData • Field • Database

Form	
• Contains the ValueFields (editable or not) of a table from the database. • To update the database, the user needs permissions and use the edit button	• Form • FormData • Field • Database • ValueField

Datum	
• Holds information retrieved from the database such as the rowID, columnName, tableName • Used to properly update or delete data from the database • Remembers its parent (the row it came from)	• Row (and it's implementations) • Database

ValueField	
• Represents a value from the database • Has the option to be updatable, and if it is, it will update the database using the table name, column name, and rowID • Rows consists of Datum objects and its rowID	• Datum • Database • UpdateButtonGroup

UpdateButtonGroup	
• Contains the edit, cancel, and save buttons which are used to update the database with the new value from a ValueField object • Holds all the editable ValueField objects to speed up updating the database • Its ValueFields are listening for when the edit, cancel, or save button are clicked	• ValueField

WelcomeController	
• Uses the Controller interface • Displays the option to choose either to sign in or sign up • Uses App to load the appropriate FXML views and controllers	• Controller • App

SignInController	
• Uses the Controller interface • Allows users to sign in with their credentials • Ensures that the appropriate User for the sql server is used, either patient or employee, since each user has different permissions • Uses App to load other FXML views	• Controller • Database • ForgotPasswordController • App

SignInController	
• Uses Database to keep track of users' sign in attempts	

SignUpController	
• Schedules an exam for a patient • Creates a new Patient object if patient is not in the database • Checks a patient in for their exam • Checks for available dates for an exam • Cancels an exam upon a patient's request	• Database • Form • ValueField • Controller

MyVisitPopupController	
• Loads the details for a patient's visit using a table or form for PatientPortal • Connects to the database to make any necessary changes such as canceling a visit	• Controller • Database • TableView • Form

VisitPopupController	
• Loads the details for a patient's visit using a table or form for WorkPortal • Allows the receptionist or nurse to begin the session (make it active) • Allows the receptionist to send a message or call the patient if they did not show up • Uses the schedule datetime from the database to calculate if the patient missed their visit • Allows nurse to easily reschedule for the patient	• Controller • Database • TableView • Form

MyPillPopupController	
• Loads the details for a patient's visit using a table or form • Connects to the database to make any necessary changes such as canceling a visit	• Controller • Database • TableView

	• Form

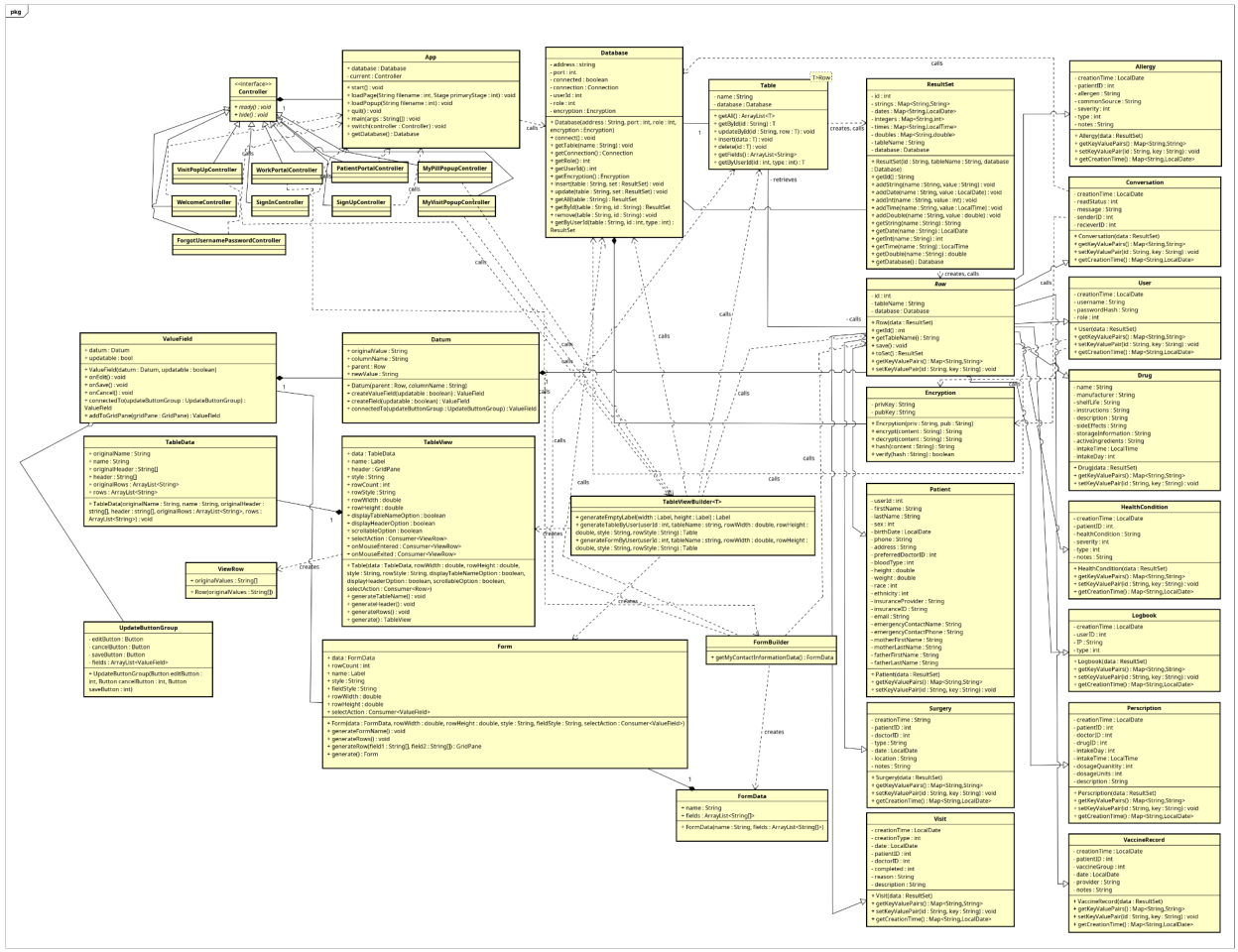
ForgotUsernamePasswordController	
• Loads the details for a patient's visit using a table or form • Connects to the database to make any necessary changes such as canceling a visit	• Controller • Database • TableView • Form

PatientPortalController	
• Uses the database to retrieve data about the patient and displays it using the FormBuilder and TableViewBuilder • Uses UpdateButtonGroup to allow the patient to update the information stored on the database (contact info) • When patient signs out, the sign in page is loaded • Allows patients to see their prescriptions • Allows patients to send messages to the clinic and also the messages they have received from the clinic • Allows patients to see their account details	• Controller • Database • TableView • Form • App

WorkPortalController	
• Loads data from the database to display information about patients such as their medical information or upcoming visits • Displays the data using TableViewBuilder and FormBuilder • Uses the VisitPopupController to display details about a visit • Uses the UpdateButtonGroup to allow the professional to update information about a patient • Allows the nurse to initiate a visit and begin recording information about the patient such as their weight, blood pressure, etc • Allows doctors to prescribe patients pills (only doctors have this permission)	• Controller • Database • TableView • Form • App

WorkPortalController	
• Allows a professional to send or respond to messages sent from patients • Allows professionals to see their account details	

Class Diagram



Test Plan for Functional Testing

Requirement: Sign Up Page

Successful Account Creation:

1. User fills out the form with an available username and email to sign up and clicks the sign up button
2. The application connects to the database to verify that the username and email are available and successfully creates the account for the user
3. A message is displayed about the successful account creation and the user is redirected to the sign in page

Hashed Password:

1. User enters available username and email to sign up and clicks the sign up button
2. The account is successfully created and the hashed password rather than the raw password is stored onto the database

Brute Force Resistance:

1. A user with the username and password as “user1” and “password123” exists on the database.
2. A hacker knows the username is “user1” and uses brute force to guess the password.
3. System uses rate limiting to prevent the brute force attack

Password Requirements:

1. A user attempts to create an account with password “password”.
2. The app denies the user and alerts them that the password needs to meet the following criteria:
    a. At least 8 characters long
    b. At least 2 numbers
    c. At least 2 special characters
3. The user then follows these requirements and successfully creates their account

Text Formatting:

1. The user enters their first name and last name as “johNNy” “salaZAR”.
2. The app formats the first name and last name as “Johnny” “Salazar”
3. The app asks the user to verify that their information is correct before clicking the sign up button
4. This happens to all other fields such as address, pharmacy, insurance provider, etc.

Text Validation:

1. When signing up, the user enters in their address as “1234 W Fake St, Tempe, AZ”.

2. The google API checks to see if this is a valid address. The app determines that it is not and so the account creation attempt is denied and the user is alerted about the invalid address.
3. A similar case will occur with the Pharmacy address they enter
4. A similar case will occur with the birth date which needs to be in the format: “mm/dd/yyyy”
5. All fields must be no longer than 64 characters long

Requirement: Sign In Page

Successful Sign In:

1. User enters their valid username and password and clicks the sign in button
2. The application connects to the database to verify the username and password
3. The app determines that the credentials are correct and redirects the user to the appropriate portal
4. The user is signed in and this is logged in the database

Unsuccessful Sign In:

1. User enters their invalid username and password and clicks the sign in button
2. The application connects to the database to verify the username and password
3. The app determines that the credentials are incorrect and alerts the user that the sign in attempt was unsuccessful
4. The username and password fields are highlighted in red
5. This sign in attempt is logged in the database

Remember Me:

1. User checks the remember me checkbox and signs in
2. The user closes the application and reopens it
3. The username and password fields are already filled in from the last time the user signed in

Forgot Username/Password:

1. User clicks the hyperlink and is redirected to a page where they can enter their email
2. The app sends an email to the user with their username and password

Brute Force Resistance:

1. A hacker attempts to sign in with the username “user1” and password “password123” 100 times in a row
2. The system detects this and locks the account for 5 minutes by the hacker’s IP address

Requirement: Patient Portal

View My Visits:

1. The My Visits page is the first page the user sees when they sign in

2. The user sees a list of their visits including the doctor's name, the date of the visit, the reason for the visit, and the status of the visit
3. The user can click on a visit to view more details and can click on a button to cancel the visit or to reschedule the visit

Cancel Visit:

1. The user clicks on a visit and clicks the cancel visit button
2. The app asks the user to confirm that they want to cancel the visit
3. The user confirms and the visit is successfully canceled

Reschedule Visit:

1. The user clicks on a visit and clicks the reschedule visit button
2. The app asks the user to enter a new date and time for the visit
3. The user enters the new date and time and clicks the reschedule button
4. The app confirms that the visit has been successfully rescheduled

View My Information:

1. The user clicks on the My Information tab
2. The user sees their information including contact information, medical information, health conditions, vaccine record, allergies, surgeries
3. The user can click on a button to edit editable information

Edit Information:

1. The user clicks on the edit button for their contact information
2. The user changes their phone number and clicks the save button
3. The app confirms that the information has been successfully updated
4. Only editable fields can be updated

Schedule Visit:

1. The user clicks on the Schedule Visit tab
2. The user sees a form to schedule a visit
3. The user enters the date and time of the visit, the reason for the visit, and the doctor they want to see
4. The user clicks the schedule visit button
5. The app confirms the availability of the visit and alerts the user that the visit has been successfully scheduled

Patient Already Scheduled:

1. The user attempts to schedule a visit with a doctor they have already scheduled a visit with
2. The app alerts the user that they have already scheduled a visit with this doctor and asks if they would like to reschedule the visit

View Chat:

1. The user clicks on the Chat tab
2. The user sees the chat with their doctor inside a scroll pane
3. The scroll pane is at the bottom of the chat so the user can see the most recent messages
4. The user can scroll through the chat and see the messages from the doctor

Send Message:

1. The user clicks on the Chat tab
2. The user sees the chat with their doctor inside a scroll pane
3. The user types a message and clicks the send button
4. The app sends the message to the doctor and the message is displayed in the chat

View My Pills:

1. The user clicks on the My Pills tab
2. The user sees a list of their pills including the name of the pill, the intake day and time, the dosage, and the reason
3. The user can click on a pill to view more details and can click on a button to contact the pharmacy or their doctor

Contact Pharmacy:

1. The user clicks on a pill and clicks the contact pharmacy button
2. The app opens the user's email client with the pharmacy's email address already filled in

Contact Doctor:

1. The user clicks on a pill and clicks the contact doctor button
2. The app opens the Chat tab and the user can send a message to the doctor

View Account Settings:

1. The user clicks on the Account Settings tab
2. The user sees a list of their account settings including their username, email, password, and the option to delete their account
3. The user can click on a setting to edit it

Edit Account Settings:

1. The user clicks on the edit button for their email
2. The user changes their email and clicks the save button
3. The app confirms that the email has been successfully updated
4. Only editable fields can be updated

Delete Account:

1. The user clicks on the delete account button
2. The app asks the user to confirm that they want to delete their account
3. The user confirms and the account is successfully deleted
4. The app redirects the user to the sign in page

Sign Out:

1. The user clicks on the sign out button
2. The user is signed and this is logged in the database
3. The app reconnects as the neutral role and redirects the user to the sign in page

Requirement: Work Portal

View Visits:

1. The user clicks on the visits tab
2. The user sees a list of upcoming visits in chronological order by date.
3. The list includes the doctor, the patient, the date, the reason, and the status of the visit.
4. The user can click on a visit to view more details and can click on a button to cancel the visit, reschedule the visit, or start the visit.

Missed Visits:

1. The user clicks on the visits tab
2. The app calculates the current date and time and determines that the user has missed a visit
3. The missed visit's status changes to "Missed" and when the user clicks on the visit, they have the option to send a message to the patient to reschedule the visit or contact the patient

Contact Patient to Reschedule:

1. The user clicks on a missed visit and clicks the contact patient to reschedule button
2. The inbox tab opens with a message to the patient to reschedule the visit already filled in

Cancel Visit:

1. The user clicks on a visit and clicks the cancel visit button
2. The app asks the user to confirm that they want to cancel the visit
3. The user confirms and the visit is successfully canceled

Reschedule Visit:

1. The user clicks on a visit and clicks the reschedule visit button
2. The app asks the user to enter a new date and time for the visit

Start Visit:

1. The user clicks on a visit and clicks the start visit button
2. The visit's status becomes "In Progress" and the user is redirected to the Active Sessions page

View Patient Records:

1. The users clicks on the Patient Records tab

2. The user sees a text field to enter the patient's first name, last name, birth date or the patient's ID
3. The user then clicks on the search button to retrieve the patient's records
4. The app determines that the patient exists and retrieves the patient's records

Edit Patient Records:

1. The user clicks on the edit button for the patient's records to allow the editable fields to become editable depending on the role of the user
2. The user changes the patient's records and clicks the save button
3. The app confirms that the patient's records have been successfully updated
4. Only editable fields can be updated

View Active Sessions:

1. The user sees a list of active sessions in chronological order by date.
2. The list includes the doctor, the patient, the date, and the reason for the visit.
3. The user can click on a session to view more details and can click on a button to end the session or a undo button to put the session back to the scheduled visits list

End Session:

1. The user clicks on a session and clicks the end session button
2. The app determines if the all the fields are filled out
3. The app asks the user to confirm that they want to end the session
4. The user confirms and the session is successfully ended
5. The database changes the status of the patient's visit to "Completed" and stores the session's details

Undo Session:

1. The user clicks on a session and clicks the undo button
2. The app asks the user to confirm that they want to undo the session
3. The user confirms and the session is successfully undone and put back into the scheduled visits list

View Chat:

1. The user clicks on the Chat tab
2. The user sees a list of chats with their patients in order of most recent
3. The user clicks on a chat to view the messages with the patient inside a scroll pane
4. The scroll pane is at the bottom of the chat so the user can see the most recent messages

Send Message:

1. The user clicks on the Chat tab
2. The user sees a list of chats with their patients in order of most recent
3. The user clicks on a chat for one of the patients
4. The user types a message and clicks the send button

5. The app sends the message to the patient and the message is displayed in the chat

View Prescription Tool:

1. The user clicks on the Prescription Tool tab
2. The user sees a text field to enter the patient's first name, last name, birth date or the patient's ID
3. The user then clicks on the search button to retrieve the patient's prescriptions
4. The app determines that the patient exists and retrieves the patient's prescriptions
5. The user is a doctor and has the permissions to add, delete, or modify the patient's prescriptions

Add Prescription:

1. The user who is a doctor clicks on the add prescription button
2. The user sees a form to add a prescription
3. The user enters the name of the pill, the intake day and time, the dosage, and the reason
4. The user clicks the add prescription button
5. The app confirms the availability of the pill and alerts the user that the pill has been successfully added

Delete Prescription:

1. The user who is a doctor clicks on a prescription and clicks the delete prescription button
2. The app asks the user to confirm that they want to delete the prescription
3. The user confirms and the prescription is successfully deleted

Edit Prescription:

1. The user who is a doctor clicks on the edit button for a prescription
2. The user changes the prescription and clicks the save button
3. The app confirms that the prescription has been successfully updated
4. Only editable fields can be updated

View Account Settings:

1. The user clicks on the Account Settings tab
2. The user sees a list of their account settings including their username, email, password, role
3. The user attempts to edit these values but they are not editable

Sign Out:

1. The user clicks on the sign out button
2. The user is signed and this is logged in the database
3. The app reconnects as the neutral role and redirects the user to the sign in page

Testing Plan for EasyDoctor Program

Objective
To ensure the functionality, usability, and reliability of the healthcare system application by systematically testing each feature against its requirements.

Test Environment
- Java: Java 21.0
- Database: MySQL 8.x

Functionality Testing

1. Test Case: Sign Up Functionality

- Objective: To verify that users can successfully create a new account.
- Steps:
  1. Navigate to the Sign Up view from the Welcome Screen.
  2. Fill in all required fields with valid data.
  3. Click the "Sign Up" button.
- Expected Result: The system should create a new account and redirect the user to the Sign In page with a success message.

2. Test Case: Sign In Functionality

- Objective: To ensure that users can sign in with valid credentials.
- Steps:
  1. Navigate to the Sign In page from the Welcome Screen.
  2. Enter valid username and password.
  3. Click the "Sign In" button.
- Expected Result: The system should authenticate the user and redirect them to their respective portal based on their role.

3. Test Case: Forgot Password Functionality

- Objective: To test the "Forgot My Password" feature for account recovery.
- Steps:
  1. Go to the Sign In page and click on "Forgot My Password".
  2. Enter a registered email or username and submit the form.

- Expected Result: The system should send an email with password reset instructions to the user's registered email address. If the user's email address is not valid, the system will do nothing.

4. Test Case: Update Personal Information

- Objective: To verify that patients can update their personal patient information.
- Steps:
  1. Sign in as a patient and navigate to the "My Information" tab.
  2. Click the “Edit” button, make changes, and then click the “Save” button
- Expected Result: The system should update the information in the database and reflect the changes in the user interface.

5. Test Case: Schedule a Visit

- Objective: To ensure that patients can schedule visits.
- Steps:
  1. Sign in as a patient and navigate to the "Schedule" tab.
  2. Select a date, time, and doctor, then submit the form.
- Expected Result: The system should register the new visit and show it in the "My Visits" tab.

6. Test Case: View All Scheduled Visits

- Objective: To verify that doctors/nurses can view their scheduled visits.
- Steps:
  1. Sign in as a doctor/nurse and navigate to the "Visits" tab.
- Expected Result: The system should display a list of upcoming, pending, and past visits.

7. Test Case: View Patient’s Scheduled Visits

- Objective: To verify that patients can view their currently scheduled visits.
- Steps:
  1. Sign in as a patient and navigate to the "My Visits" tab.
- Expected Result: The system should display a list of upcoming, pending, and past visits for the currently logged in patient.

8. Test Case: Prescription Management

- Objective: To ensure that doctors can manage prescriptions.
- Steps:

1. Sign in as a doctor and navigate to a patient's record.
2. Add, edit, or delete a prescription.

- Expected Result: The system should update the prescription details in the patient's record.

9. Test Case: Send and Receive Messages

- Objective: To test the messaging functionality between patients and healthcare providers.
- Steps:
  1. Sign in and navigate to the "Inbox" tab.
  2. Compose a new message or reply to an existing one.
- Expected Result: The system should deliver the message to the recipient and display it in their inbox.

10. Test Case: Sign Out Functionality

- Objective: To verify that users can securely sign out.
- Steps:
  1. Sign in to the application.
  2. Click the "LogOut" button.
- Expected Result: The system should sign the user out and redirect them to the Sign In page.

Security Testing

11. Test Case: SQL Injection Prevention

- Objective: Ensure the application sanitizes user inputs to prevent SQL injection attacks.
- Steps:
  1. Navigate to any input field in the application (e.g., login, search).
  2. Enter an SQL injection code snippet (e.g., `' OR '1'='1`).
  3. Submit the form or search request.
- Expected Result: The application should not execute the SQL code and should either return an error or ignore the input.

Usability Testing

12. Test Case: Intuitive Navigation

- Objective: Test the intuitiveness and ease of navigation within the application.
- Steps:
  1. Sign into the application.
  2. Without using a direct link, attempt to navigate to a specific feature (e.g., Schedule a Visit).
  3. Note the number of clicks and time taken to reach the desired page.
- Expected Result: Users should be able to reach the desired feature with minimal clicks and within a short time, indicating intuitive navigation.

Integration Testing

13. Test Case: Database Integration Consistency

- Objective: Verify that the application correctly interacts with the database for all basic database operations.
- Steps:
  1. Perform various actions that involve database updates (e.g., creating a new patient record, updating prescription details).
  2. Check the database directly to ensure the updates are reflected accurately.
- Expected Result: The database should accurately reflect the changes made through the application, with data integrity maintained.

Defect Tracking and Reporting
- All identified defects will be logged into a centralized repository, allowing for quick correction of issues.

Credit Sheet:

Team Number Name	**Contributions**
Johnny Salazar	CRC, class identification, partial testing plan, prototype
ZhiYue Wang	Use case and Use case diagram
XiuQi Yang	
William Whittaker	UML diagram, partial CRC, partial class identification
Colton Hutchins	Testing plan