Super Hospital System

By Team ABoT (Always Be on Time)

Final Document

Aadil Alladina

Dustin Green

Ketra King

John Rucker

Don Squirek

Table of Contents

1. Introduction Page 3
2. Requirements Elicitation Page 4

Requirements Traceability Matrix Page 5

1. System Analysis and Design Page 7
   1. Use Cases Page 7
   2. Sequence Diagrams Page 22
   3. Category Interaction Diagram Page 37
2. Object Design Page 38
3. Test Cases Page 48
4. Rationale for Project and Page 57

Implementation

1. Project Cost Analysis Page 59
   1. Function Point Analysis Page 59
   2. COCOMO Page 61
2. Horizontal Prototype Page 62
3. Project Legacy Page 92
4. Work Schedule Diagram Page 93
5. Gantt Chart Page 94
6. Project Dictionary Page 97
7. Team Member Resumes Page 99
8. User Guide Page 104
9. Source Code Page
   1. Text Based Version Page
   2. GUI Version

1. Introduction

Our software is designed primarily for use large hospitals. It contains all the functionality a hospital needs for tracking both staff and patient information, but its signature feature is its ability to access an extensive database of both medicinal and recreational drugs. Currently, any time a staff member needs to access information about the various chemical compounds hospitals deal with daily, they must consult the giant book of drugs. We wanted to put all that information at the staff’s fingertips, thereby improving the speed at which staff could access vital drug information, and reducing any chance of errors like referring to the wrong page or entry in the book. We felt that this would not only make life easier for doctors and nurses, but would also greatly improve the quality of patient care. This extensive drug database, together with the core functionality provided to the hospital staff, makes up the base of our software project.

In addition however, our software will also have the ability to read data directly from hospital equipment, allowing nurses and doctors to keep an eye on their patients’ condition and vital signs even when in another room or floor. This functionality is currently not active, but will be activated upon getting a contract and learning what equipment we need the software to integrate with. Coupled with this functionality is the software’s ability to signal the doctor and/or nurses assigned to a patient whose condition has worsened. This functionality is also disabled until vitals reader is complete and working.

On the other side of the hospital from the nurses and doctors are the pharmacists and system administrators. Our system is designed to give a lot of functionality while still being uncluttered and intuitive, and its functions for staff not involved in patient care is no exception. For system administrators, our software relies on SQLite databases for efficiency and ease of use, allowing them to both and remove entries from the databases either by working on the database directly or through our software. On the pharmacist end, our system features the ability to track what medicines the pharmacy does have in stock and provides functionality for doctors to request medicine from the pharmacy, with an alert to the nurse on duty when the prescription has been filled. Pharmacists are also capable of updating the drug database with new entries as new medicines and new recreational drugs come into existence.

All in all, our Super Hospital System will provide everything a large hospital needs in order to run efficiently, and provides the framework for the staff to grow closer as a well trained team. We believe this will greatly improve patient care, as well as making life easier for the staff. We want to make hospitals a better place, and that’s why we came up with the Super Hospital System.

2. Requirements Elicitation and Requirements Traceability Matrix

**Requirements Elicitation: Super Hospital System**

**SHS-1.0 Introduction**

The SHS is to be a software system that monitors patients' vitals and medications, and keeps track of their location according to their schedule. It is also to keep staff informed of their schedule, and issue alerts if there is a negative change in a patient's status.

**SHS-2.0 General Use**

Users (doctors, nurses, and administrators) shall log in by entering their user name and password into the GUI window. The software shall check this information against a database containing all the staff's login information. If the entered information is incorrect, the software shall return the user to the login screen and prompt for the correct user name and password. The software shall allow users to reset their passwords by using security questions to authenticate their identity. Upon successful login, the software shall bring the user to the software's home screen.

**SHS-3.0 Doctors**

Logging in as a doctor shall allow the doctor to view a list of panels, each with a single patient's name on it. When the doctor selects a panel, the software shall bring the doctor to a new window containing all patient information, including diagnosis, medications, and next appointment. The software shall have a database that contains patient name, address, current medication, reason for visit, emergency contact information, and insurance. The doctor shall be able to edit this information. If a doctor prescribes a medication that interferes with a current medication, the software shall issue a pop-up alert that the doctor must acknowledge in order to proceed. The software shall have a comprehensive database of both medicinal and recreational drugs, their uses, side affects, typical duration of affects, and other drugs that they cannot be used with. If any patient's vitals move outside of normal range the panel for that patient will flash red to alert the doctor. To dismiss the alert the doctor shall select the patient's panel and a dialogue box will pop up asking for dismissal. The software shall send digitally send medication prescriptions to the pharmacy to be filled. The software shall keep record of which doctor prescribed the prescription, the dosage, the time length of the prescription, and the time the prescription was sent to the pharmacy. The software shall alert the patient's doctor or nurse station when a prescription is ready in the pharmacy.

**SHS-4.0 Nurses**

Logging in as a nurse shall allow the nurse to view a list of room numbers in their ward and patient names for each room. When the nurse selects a panel, the software shall bring the nurse to a new window containing patient name, diagnosis, medications, and next check-up. When a prescription has been filled by the pharmacy, the panel for that patient's room shall be illuminated green on the nurse's terminal. The software shall keep record of the date and time a medication is administered to a patient. The software shall utilize the drug database to alert the patient's doctor or nurses if a medication is scheduled soon enough to cause an overdose. The software shall utilize the drug database to alert the patient's doctor or nurses if a patient's prescribed medication will conflict with any current medications. If any patient's vitals move outside of the norm, the panel for that patients room shall flash read to alert the nurse, and shall be dismissed by selecting that panel and confirming the selection via dialogue box. Nurses shall be able to page doctors on duty by clicking the “Page Doctor” button on a patient's room panel, which will illuminate the panel in green on the doctor's terminal.

**SHS-5.0 Administrators**

Administrators shall be able to create and remove users from the login database. Permission level for users shall be assigned by the administrator according to job position. The software shall contain a separate database for the login information of hospital personnel. The software shall utilize a database containing all the staff's work schedules. The software shall utilize a database containing all of the doctors' appointments. The software shall utilize a database containing all of the patient's regular checkup times. These databases shall be modifiable by the hospital administrators.

**Requirements Traceability Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entry # | Para #\* | SHS Requirements Traceability Matrix | Type | Use Case |
| 1 | 2.0 | The user shall be able to enter their username and password into a terminal. | SW, HW | 01 |
| 2 | 2.0 | System shall check the username and password to a database to allow a log in. | SW | 01 |
| 3 | 2.0 | If the logging in is unsuccessful the user shall be returned to the login screen and presented with an error message. | SW | 01 |
|
| 4 | 2.0 | If the logging in is successful the system shall bring the user to the software's home screen. | SW | 01 |
| 5 | 3.0 | Logging in as a doctor shall allow the doctor to view a list of panels each with a single patients name on it. | SW | 02 |
| 6 | 3.0 | When the doctor selects a panel the system shall bring the doctor to a new window that shall display all patient information including diagnosis, next appointment, and medications. | SW | 02 |
| 7 | 3.0 | The doctor shall be capable of editing or revising all patient care information. | SW | 03 |
| 8 | 3.0 | The software shall have a database that contains patient name, address, current medications, reason for visit, insurance, and emergency contact information. | SW | 02 |
| 9 | 3.0 | If the doctor prescribes medication that interferes with any current medication, a pop-up alert shall appear and the doctor must then confirm the choice. | SW | 05 |
| 10 | 3.0 | The software shall have a comprehensive database of both medicinal and recreational drugs, their uses, side affects, typical duration of effects, and drugs it can't be used with. | SW | 11 |
| 11 | 3.0 | If the doctor prescribes valid medication the software shall digitally send prescriptions to the pharmacy to be filled. | SW | 05 |
| 12 | 3.0 | If any of the patient’s vitals move outside of a stable window the panel for that patient will flash red to alert the doctor. | SW, HW | 06 |
| 13 | 3.0 | To dismiss the alert, the doctor shall select the panel pertaining to that patient and a dialog box shall appear asking to confirm the dismal. | SW | 06 |
| 14 | 3.0 | The software shall keep record of which doctor prescribed the prescription, the dosage, time length of the prescription, and the time it was sent to the pharmacy. | SW | 06 |
| 15 | 3.0 | The software shall alert the patient's doctor or nurse station when the prescription is ready. | SW | 09 |
| 16 | 4.0 | Logging in as a nurse shall allow the nurse to view a list of room numbers in their ward and patient names for each room. | SW | 02 |
| 17 | 4.0 | When the nurse selects a panel, the software shall bring the nurse to a new window that shall display diagnosis, patient name, medication and next check up. | SW | 02 |
| 18 | 4.0 | When a perscription has been filled by the pharmacy, the panel for that patient’s room shall be illuminated green on the nurse’s terminal. | SW | 09 |
| 19 | 4.0 | The software shall keep record of the date and time a medication is administered to a patient. | SW |  |
| 20 | 4.0 | The software shall utilize the drug database to alert doctors and nurses if a patient's next medication is scheduled soon enough to cause an overdose. | SW |  |
| 21 | 4.0 | The software shall utilize the drug database to alert doctors and nurses if a patient's prescribed medication will conflict with any current medications. | SW | 05 |
| 22 | 4.0 | If any of the patient’s vitals move outside of a stable window the panel for that patient’s room shall red flash to alert the nurse, and shall be dismissed by selecting that panel and confirming that selection via dialog box. | SW | 06 |
| 23 | 4.0 | Nurses shall be able to page doctors on duty by clicking the “Page Doctor” button on a patient's room panel, which will illuminate the panel in green on the doctor's terminal. | SW | 07 |
| 24 | 5.0 | Administrators shall have the option to create a new account with a username and password. | SW | 13 |
| 25 | 5.0 | Permission level for users shall be assigned by an administrator according to job position. | SW | 13 |
| 26 | 5.0 | The system shall contain a separate database for the login information of hospital personal. | SW | 01 |
| 27 | 5.0 | The software shall utilize a database containing all the staff's work schedules. | SW | 15 |
| 28 | 5.0 | The software shall utilize a database containing all of the doctors' appointments. | SW | 15 |
| 29 | 5.0 | The software shall utilize a database containing all of the patient's regular checkup times. | SW | 15 |
| 30 | 5.0 | These databases shall be modifiable by the hospital administrators. | SW |  |

1. System Analysis and Design
   1. **Use Cases**

**Use Case 01: User\_Login**

**Overview:**

This Use Case enables the Nurse, Doctor, or Administrator (referred to as User from here on) to login to the system. The User will be prompted for a username and password. If the information entered is incorrect, the user will be asked to reenter their information, or change their password. After three attempts they are locked out and will be told to contact an Administrator.

**Preconditions:**

1.The Login\_GUI is displayed. 3. The User has a valid username and password

2. The login database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The User enters their username. | 1. Username field is updated. |
| 2. The User enters their password. | 2. Password field is updated. |
| 3. The User clicks the Sign In button. | 3. The software checks the database for a matching username/password pair, if no pair is found then the Login\_GUI is refreshed and the User is prompted to reenter their username and password. |
| 4. The User reenters their username. | 4. Username field is updated. |
| 5. The User reenters their password. | 5. Password field is updated. |
| 6. The User clicks Sign In button. | 6. The software checks the database for a matching username/password pair, if one is found then the Login\_GUI is destroyed, the Home\_GUI is displayed, and the User is given access to patient data according to their permissions. |
| 7. User clicks Sign Out button. | 7. User is logged out, Home\_GUI is destroyed, and Login\_GUI is created. |

**Scenario Notes:**

Items 4-6 are dependent on the user failing to enter correct login information. If the login is successful on the first attempt, items 4-6 will not occur.

**Post Conditions:**

1. The User is allowed access to the patient database according to their permissions.

2. The User is brought to the Home\_GUI.

**Exceptions: Use Cases Utilized:**

1. The login database cannot be accessed. None

**Required GUI: Timing Constraints:**

1. Login\_GUI None

2. Home\_GUI

**Use Case 02: Add\_New\_Patient**

**Overview:**

This Use Case allows Doctors to add new patients to their list, and enter patient information into the Patient database. User must have Doctor Permissions.

**Preconditions:**

1. The User is in the Home\_GUI. 3. The patient records database is accessible.

2. There are no Condition Critical alerts active.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks on Patient\_Display panel. | 1. Patient\_Display\_View pop-up is created. |
| 2. User clicks on Add Patient button. | 2. Add\_Patient\_View is created. |
| 3. User clicks the Exit button. | 3. Add \_Patient \_View pop-up is destroyed and no data is added to the database. |
| 4. User enters information in Add\_Patient\_View pop-up's fields. | 4. Pop-up fields are updated. |
| 5. User clicks Submit button. | 5. A new patient entry is created in the patient records database with the given information, and the Add\_Patient\_View pop-up is destroyed. |

**Scenario Notes:**

Only name fields are required to be filled, as all others can be filled at a later time by the doctor.

**Post Conditions:**

1. User is returned to the Home\_GUI.

**Exceptions: Use Cases Utilized:**

1. Patient records database can't be accessed. None

**Required GUI: Timing Constraints:**

1. Home\_GUI None

2. Patient\_Display\_View

3. Add\_Patient\_View

**Use Case 03: View\_Patient\_Information**

**Overview:**

This Use Case allows Doctors and Nurses to view patient information, consisting of medical records, personal information, diagnosis, and condition. How much patient information is displayed depends on the whether the User has Doctor or Nurse permissions.

**Preconditions:**

1. The User is in the Home\_GUI. 3. The patient records database is accessible.

2. There are no Condition Critical alerts active.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks on Patient\_Display panel. | 1. Patient\_Display\_View is created. |
| 2. User clicks the Exit button. | 2. Diagnosis\_View is destroyed. |
| 3. User clicks Back button | 3. Patient\_Display\_View is destroyed and the user is returned to the Home\_GUI |

**Scenario Notes:**

A User with Nurse permissions will only see information relevant to patient care, and will be unable to edit the patient's records as in Items 3-7 can only be performed by a User with Doctor permissions. Changes are not required.

**Post Conditions:**

1. User is returned to the Home\_GUI.

**Exceptions: Use Cases Utilized:**

1. Patient records database can't be accessed. None

**Required GUI: Timing Constraints:**

1. Patient\_Display\_View None

4. Home\_GUI

**Use Case 04: Edit\_Patient\_Records**

**Overview:**

This Use Case allows Doctors to edit patient records, including name, age, gender, address, phone number, insurance provider, and social security number.

**Preconditions:**

1. User must have Doctor permissions. 3. There are no Condition Critical alerts active.

2. User must be in the Patient\_Patient\_Display panel. 4. The patient records database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks View Patient Records button. | 1. Patient\_Records\_View pop-up is created. |
| 2. User clicks Edit\_Records button. | 2. The fields in Patient\_Records\_View pop-up are now can now be overwritten. |
| 3. User enters new information in the Patient\_Records fields. | 3. Patient\_Records fields are updated. |
| 4. User clicks Submit button. | 4. Patient records are updated in the patient records database, and the information is updated in the Patient\_Records\_View pop-up. |
| 5. User clicks Exit button. | 5. Patient\_Records\_View pop-up is destroyed. |

**Scenario Notes:**

Both Nurses and Doctors can click the View Patient Records button. Doctors will see all of the information contained on this page, and Nurses will only see name, age, and gender. The patient's records do not have to be edited as in Items 3-4 in order to complete the Use Case. If Item 4 does not occur, the patient records database will not be updated.

**Post Conditions:**

1. User is in the Patient\_Display\_View.

2. Patient record database is updated (if changes were made to the patient's records).

**Exceptions: Use Cases Utilized:**

1. The patient records database cannot be accessed. 1. Use Case 02

**Required GUI: Timing Constraints:**

1. Patient\_Display\_View None

2. Patient\_Records\_View pop-up

**Use Case 05: Change\_Patient\_Diagnosis**

**Overview:**

This Use Case allows the User to view a patient's current diagnosis and their diagnosis history. Users with Doctor permissions can change the diagnosis.

**Preconditions:**

1. The User is in Patient\_Display\_View. 3. There are no Condition Critical alerts active.

2. Patient records database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks Current\_Diagnosis button. | 1. Diagnosis\_View pop-up is created. |
| 2. User types a new diagnosis into the New Diagnosis field. | 2. New Diagnosis field is updated. |
| 3. User clicks the Submit button. | 3. Patient records database is updated, and the Current Diagnosis panels in Display\_Patient\_View and the Diagnosis\_View pop-up are updated, all with the new diagnosis and date changed, and the Diagnosis\_View pop-up is destroyed. |
| 4. User clicks the Exit button. | 4. The Diagnosis\_View pop-up is destroyed. |

**Scenario Notes:**

Item 4 cannot happen if Item 3 happens. If the User clicks the Exit button after entering a new diagnosis, nothing will be updated. Only clicking the submit button will updated Current Diagnosis panels and the patient records database.

**Post Conditions:**

1. User is in the Patient\_Display\_View.

2. Patient record database is updated (if changes were made to the patient's diagnosis).

**Exceptions: Use Cases Utilized:**

1. Patient records database cannot be accessed. Use Case 02

**Required GUI: Timing Constraints:**

1. Patient\_Display\_View None

2. Current\_Diagnosis pop-up

**Use Case 06: New\_Prescription**

**Overview:**

This Use Case enables a Doctor to digitally prescribe medication for a patient. The prescription is automatically sent to the pharmacy to be filled. The drug database will check if this new medication cannot be used in combination with any current medication, and if the medication is incompatible the software will issue a warning to the Doctor.

**Preconditions:**

1. The Doctor is in the Patient\_Display\_View. 3. The patient record database is accessible.

2. There are no Condition Critical alerts active. 4. The pharmacy database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. Doctor clicks Medication button in Patient\_Display\_View. | 1. Medication\_History pop-up is created. |
| 2. Doctor enters a new prescription. | 2. Prescription field is updated. |
| 3. Doctor clicks on the Submit button. | 3. The new prescription is checked against the patient's previous medications in the drug database to see if it conflicts with any of those drugs, if it does then Drug\_Conflict\_Warning pop-up is created. |
| 4. Doctor clicks Yes button. | 4. Drug\_Conflict\_Warning pop-up is destroyed, the patient record database is updated with the new prescription, date and time it was issued, and the pharmacy database is updated with a new prescription, along with a record of the date, time, and the Doctor who issued it. |
| 5. Doctor clicks No button. | 5. Drug\_Conflict\_Warning is destroyed, no databases are updated. |
| 6. Doctor clicks on the Exit button. | 6. Medication\_History pop-up is destroyed. |

**Scenario Notes:**

Items 2 and 3 are optional. The Doctor does not need to fill out a new prescription in order to complete the scenario. Items 4 and 5 are mutually exclusive, if one happens then the other cannot happen for the same prescription.

**Post Conditions:**

1. Patient record database is updated.

2. Pharmacy database is updated with new prescription.

3. Doctor returns to Patient\_Display\_View.

**Exceptions: Use Cases Utilized:**

1. The patient record database cannot be accessed. 1. Use Case 02

2. The pharmacy database cannot be accessed.

3. The drug database cannot be accessed.

**Required GUI: Timing Constraints:**

1. Patient\_Display\_View None

2. Medication\_History pop-up

3. Drug\_Conflict\_Warning

**Use Case 07: Patient\_Critical\_Alarm**

**Overview:**

This Use Case alerts the User when a patient's vital signs deviate from normal range by changing the patient's panel in Home\_GUI from green to a flashing red color.

**Preconditions:**

1. User is in Home\_GUI.

2. Patient vital signs deviate from normal range.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks on the Patient Panel that issued the alert. | 1. Patient\_Display\_View is created. |
| 2. User clicks on Condition button. | 2. Dismiss\_Alert pop-up is created. |
| 3. User hits No button. | 3. Dismiss\_Alert pop-up is destroyed and the Condition panel remains flashing red. |
| 4. User hits Yes button. | 4. Dismiss Alert pop-up is destroyed, if the patient's condition is still critical then the panel will remain orange until their condition changes. |
| 5. User hits Back button. | 5. User is returned to Home\_GUI. |

**Scenario Notes:**

Item 3 cannot be performed after Item 4, but Item 4 can be performed after Item 3. The patient's panel in Home\_GUI will also remain orange until the patient's condition changes.

**Post Conditions:**

1. Alert is cleared.

2. User returns to Home\_GUI.

**Exceptions: Use Cases Utilized:**

1. Vital sign monitors not connected to network. Use Case 02

**Required GUI: Timing Constraints:**

1. Home\_GUI None

2. Patient\_Display\_View

3. Dismiss\_Alert pop-up

**Use Case 08: Doctor\_Page\_Alert**

**Overview:**

This Use Case allows a Nurse to page a Doctor. The patient panel the nurse is paging from will turn yellow, and when the User clicks it it will take them to the Patient\_Display\_View, with a pop-up box displaying the page's text.

**Preconditions:**

1. User sending the page has Nurse permissions and is 4. There are no Condition Critical alerts active.

in Patient\_Display\_View.

2. User receiving the page has Doctor permissions and

is in Patient\_Display\_View or Home\_GUI.

3. Patient database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User with Nurse permission clicks red colored panel in Home\_GUI. | 1. Patient\_Display\_View is created. |
| 2. User clicks the Page Doctor button. | 2. Page\_Send pop-up is created, with premade message in Message field and the patient's Doctor in the Address field, the Address field can be changed by using a dropdown tab. |
| 3. User clicks Address field and selects a new Doctor as recipient. | 3. Address field is updated with the new recipient. |
| 4. User edits the Message field. | 4. Message field is updated. |
| 5. User clicks Send Page button. | 5. Patient database is updated with the new page, and an alert is displayed on the screen of the chosen Doctor. |
| 6. User with Doctor permission clicks the yellow colored panel in Home\_GUI. | 6. Patient\_Display\_View and Nurse\_Page pop-up are created, if the User is already in Patient\_Display\_View then only the Nurse\_Page pop-up is created. |
| 7. User clicks Confirm button. | 7. Nurse\_Page pop-up is destroyed. |
| 8. User clicks Back button. | 8. Patient\_Display\_View is destroyed, and User is returned to Home\_GUI. |

**Scenario Notes:**

Items 1-4 can only be performed by Users with Nurse permissions. Items 5 and 6 can only be performed by Users with Doctor permissions. Item 7 is performed by Users with both permissions in order to return to Home\_GUI. Item 7 is optional and does not have to be performed.

**Post Conditions:**

1. User returns to Home\_GUI.

2. After Items 1-4 are performed, the recipient Doctor's panel for that patient will turn yellow (if the user is a Nurse).

**Exceptions: Use Cases Utilized:**

1. Patient records database cannot be accessed. Use Case 02

**Required GUI: Timing Constraints:**

1. Home\_GUI None

2. Patient\_Display\_View

3. Page\_Send pop-up

4. Nurse\_Page pop-up

**Use Case 09: Pharmacy\_Login**

**Overview:**

This Use Case enables Pharmacist (User) to login to the system. The Pharmacist will be prompted for a username and password. If the information entered is incorrect, the user will be asked to reenter their information, or change their password. After three attempts they are locked out and will be told to contact an Administrator.

**Preconditions:**

1. The Login\_GUI is displayed. 3. The User has a valid username and password

2. The login database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The User enters their username. | 1. Username field is updated. |
| 2. The User enters their password. | 2. Password field is updated. |
| 3. The User clicks the Sign In button. | 3. The software checks the database for a matching username/password pair, if no pair is found then the Login\_GUI is refreshed and the User is prompted to reenter their username and password. |
| 4. The User reenters their username. | 4. The Username field is updated. |
| 5. The User reenters their password. | 5. The password field is updated. |
| 6. The User clicks Sign In button. | 6. The software checks the database for a matching username/password pair, if one is found then the Login\_GUI is destroyed, the Pharm\_Home\_GUI is displayed, and the User is given access to data according to their permissions. |
| 7. The User clicks the Sign Out button. | 7. User is returned to the main Login Screen. |

**Scenario Notes:**

Items 4-6 are dependent on the user failing to enter correct login information. If the login is successful on the first attempt, Items 4-6 will not occur. Upon login will be able to access patient prescriptions needing to be filled and will have the ability to enter new data for a prescription to be filled.

**Post Conditions:**

1. User is in the Pharm\_Home\_GUI.

2. Patient pharmacy record database is updated (if changes were made to the patient's records).

**Exceptions Use Cases Utilized**

1. Login database cannot be accessed. None

**Required GUI Timing Contraints.**

1. Login\_GUI None

2. Pharm\_Home\_GUI

**Use Case 10: Fill\_Prescriptions**

**Overview:**

This Use Case enables the User to fill medication prescriptions sent by the Doctors. It will also show the Doctor who asked for the prescription, the drug prescribed, length of prescription, and the patient it was prescribed for.

**Preconditions:**

1. The Pharm\_Home\_GUI is displayed.

2. Patient records database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks the Fill Prescriptions button. | 1. Prescription\_View pop-up is created and shows the User the list of requested prescriptions. |
| 2. User clicks one of the Fill Prescription buttons when that particular prescription is ready. | 2. The prescription to the left of the button is filled, and an alert sent to the patient's Nurse station to notify them that the prescription is ready, and the patient records database is updated. |
| 3. User clicks the Exit button. | 3. Prescription\_View pop-up is destroyed. |

**Scenario Notes:**

Upon completion of Item 3, the User will see the Pharm\_Home\_View. From there they can either log out, or perform operations on the drug database. In Item 2, filled prescriptions will disappear from the queue once the Pharmacist hits the Fill Prescription button. Item 2 is optional, prescriptions do not have to be filled in order to log out.

**Post Conditions:**

1. The Pharm\_Home\_GUI is displayed.

2. Patient record database is updated (if any prescriptions were filled).

**Exceptions Use Cases Utilized**

1. Login database cannot be accessed. Use Case 08

**Required GUI Timing Contraints**

1. Pharm\_Home\_GUI None

2. Prescription\_View pop-up

**Use Case 11: Drug\_Database\_Query\_Modify**

**Overview:**

This Use Case enables the User to search, read, and modify current entries in the drug database.

**Preconditions:**

1. The Pharm\_Home\_GUI is displayed.

2. Drug database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks the Drug Database button. | 1. Drug\_Database\_View is created. |
| 2. User enters search terms into the Search field. | 2. Search field is updated. |
| 3. User clicks the Search button. | 3. Drug\_Results\_View is created, and displays a list of search results. |
| 4. User clicks one of the search results. | 4. Drug\_Info pop-up is created and displays all information about the drug. |
| 5. User clicks Edit Information button. | 5. The User can now edit the information in the Drug\_Info pop-up. |
| 6. User clicks Save Changes button. | 6. Changes are saved and updated in the drug database and Drug\_Info pop-up. |
| 7. User clicks Cancel button. | 7. Drug\_Info pop-up is destroyed and no changes are made in the database. |
| 8. User clicks the Exit button. | 8. Drug\_Results\_View is destroyed and the User is returned to the Drug\_Database\_View. |

**Scenario Notes:**

Items 5 and 6 do not have to be performed. Item 7 can be performed before Item 6 to cancel changes, or performed after 6 to exit the pop-up.

**Post Conditions:**

1. The Drug\_Database\_View is displayed.

2. The drug database is updated (if changes were made).

**Exceptions Use Cases Utilized**

1. Drug cannot be accessed. Use Case 08

**Required GUI Timing Constraints**

1. Pharm\_Home\_GUINone

2. Drug\_Database\_View

3. Drug\_Info pop-up

4. Drug\_Results\_View pop-up

**Use Case 12: Drug\_Database\_New\_Entry**

**Overview:**

This Use Case allows the User to add new drugs to the database.

**Preconditions:**

1. The Pharm\_Home\_GUI is displayed.

2. Drug database is accessible.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User clicks the Drug Database button. | 1. Drug\_Database\_View is created. |
| 2. User clicks New Entry button. | 2. New\_Entry\_Form pop-up is created. |
| 3. User fills out drug name, uses, sideaffects, appearance, and other information. | 3. The various fields on the form are updated. |
| 4. User clicks the Cancel button. | 4. The drug database is not updated and the User is returned to the Drug\_Database\_View. |
| 5. User clicks the Submit button. | 5. Drug database is updated with a new drug entry and User is returned to Drug\_Database\_View. |

**Scenario Notes:**

Item 4 can be performed before or after Item 3. Item 4 can be performed, OR Item 5. The User cannot perform both during the same scenario.

**Post Conditions:**

1. The Drug\_Database\_View is displayed.

2. The drug database is updated (new drugs added).

**Exceptions Use Cases Utilized**

1. Drug cannot be accessed. Use Case 08

**Required GUI Timing Constraints**

1. Pharm\_Home\_GUINone

2. Drug\_Database\_View

3. New\_Entry\_Form

**Use Case 13: Administrator\_Login**

**Overview:**

This Use Case enables Administrator (User) to login to the system. The Administrator will be prompted for a username and password. If the information entered is incorrect, the user will be asked to reenter their information, or change their password. After three attempts they are locked out and will be told to contact an Administrator.

**Preconditions:**

1. The Login\_GUI is displayed. 3. The Administrator (User) has a valid

2. The login database is accessible. username and password.

**Scenario:**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The User enters their username. | 1. Username field is updated. |
| 2. The User enters their password. | 2. Password field is updated. |
| 3. The User clicks the Sign In button. | 3. The software checks the database for a matching username/password pair, if no pair is found then the Login\_GUI is refreshed and the User is prompted to reenter their username and password. |
| 4. The User reenters their username. | 4. Username field is updated. |
| 5. The User reenters their password. | 5. Password field is updated. |
| 6. The User clicks Sign In button. | 6. The login database is searched for a matching username/password pair, if one is found then the Login\_GUI is destroyed, the Home\_GUI is displayed, and the User is given access to data according to their permissions. |

**Scenario Notes:**

Items 4 and 5 are dependent on the user failing to enter correct login information. If the login is successful on the first attempt, Item 6 will occur without Items 4 and 5 happening.

**Post Conditions:**

1. User is in the Administrator \_Display\_View.

2. User has access to the login database.

**Exceptions Use Cases Utilized**

Login database cannot be accessed. None

**Required GUI Timing Contraints**

1. Login\_GUI None

2. Admin\_View

**Use Case 14: Login\_Database\_Add\_User**

**Overview:**

This Use Case enables the User to add new Users to the database.

**Preconditions:**

1. The User is in Admin\_View.

2. The User has access to the login database.

**Scenario:**

|  |  |
| --- | --- |
| 1. The User clicks Create a new User button. | 1. New\_User pop-up is displayed showing fields to add a new user (Username, Password, User's name, and permission). |
| 2. The User enters information into the username, password, User's name, and permission fields. | 2. The fields are updated. |
| 3. User clicks the Submit button. | 3. New User is added to the login database, and the New\_User pop-up's fields are cleared. |
| 4. The User clicks the Cancel button. | 4. The software returns user to the to the Administrator to Admin\_View without updated the database. |

**Scenario Notes:**

In Item 2, all of the fields must be filled in order to submit the changes in Item 3. Item 4 can take place at any time after Item 1, and will return the User.

**Exceptions Use Cases Utilized**

1. Login database cannot be accessed. Use Case 11

**Required GUIs Timing Constraints**

1. Admin\_View None

2. New\_User pop-up

**Use Case 15: Login\_Database\_\_Query\_Modify**

**Overview:**

This Use Case enables the User to search for, modify, and remove Users in the login database.

**Preconditions:**

1. The User is in Admin\_View.

2. The User has access to the login database.

**Scenario:**

|  |  |
| --- | --- |
| 1. The User clicks Edit Current User button. | 1. User\_Search pop-up is displayed. |
| 2. The User enters a User's name into the search field. | 2. The search field is updated. |
| 3. The User hits the Search button. | 3. The software returns the User's information in the User\_Search pop-up, if the User is not found then the message “User not found” is displayed. |
| 4. The User clicks Delete User button. | 4. The deleted User is removed from the database and the current User is returned to the Admin\_View. |
| 5. The User edits the information in the Name, Username, Password, and Permission. | 5. The four fields are updated. |
| 6. The User clicks the Submit Changes button. | 6. The login database is updated with the new information. |
| 7. The User clicks the Exit button. | 7. The software returns User to the to the Admin\_View. |

**Scenario Notes:**

Item 4 does not have to be performed to complete the scenario. If Item 4 is performed, then Items 5-7 will not be performed.

**Exceptions Use Cases Utilized**

1. Login database cannot be accessed. Use Case 11

**Required GUIs Timing Constraints**

1. Admin\_View None

2. User\_Search pop-up

* 1. **Sequence Diagrams**

**Use Case 01: User\_Login**

Action Systems Boundary Update Database Home\_GUI displayed

1. The User enters their username.

2. The User enter their password

3. The User clicks the Sign In button

4. The User re-enters their username.

5. The User re-enters their password.

6. The User clicks Sign in button.

7. The.User clicks Sing Out button.

6. Destroy ()

6. DatabaseCheck () passed

5. Update ()

4. Update ()

3. DatabaseCheck () failed

1. Update ()

2. Update ()

**Use Case 02: Add\_New\_Patient**

Actions Systems Boundary Update Database

1. The User clicks Add New Patient button.

2. User enters new info. in the Patient\_Records fields.

3.User clicks submit button.

4. User clicks Exit button

1. Add\_ Patient\_View pop-up created

4. Destroy ()

2. UpdateFields()

3. UpdateDatabase ()

**Use Case 03: View\_Patient\_Information**

Actions Systems Boundary Display View Database

1. The User clicks on Patient\_Display panel.

2. The User clicks the Exit button.

3. The User clicks Back button.

3. Patient\_Display\_View destroyed (Destroy ())

2. Diagnosis\_View destroyed (Destroy ())

1. Patient\_Display\_View () (Database accessed)

**Use Case 04: Edit\_Patient\_Records**

Actions Systems Boundary Update View Update Database

1. The User clicks View Patient Records button.

2. User clicks Edit\_Records button

3. User enters new info. in the Patient\_Records fields.

4. User clicks submit button.

5. User clicks Exit button

1. Patient\_Records\_View () pop up created

6. Destroy ()

2. Edit\_Patient\_Records ()

4. UpdateDatabase ()

3. UpdateFields ()

**Use Case 05: Change\_Patient\_Diagnosis**

Actions Systems Boundary Desktop View Database

1. The User clicks Current\_Diagnosis button

2. The User types a new diagnosis into the New Diagnosis field.

3. The User clicks the Submit button

4. The User clicks the Exit button.

1. Diagnosis\_View () pop-up created

3. UpdateDatabase ()

Diagnosis\_View Updated

2. Update ()

4. Destroy ()

**Use Case 06: New\_Prescription**

Actions Systems Boundary Desktop View Update Drug Check Database

1. Doctor clicks Medication button in Patient\_Display\_View

2. Doctor enters a new prescription

3. Doctor clicks on the Submit Button

4. Doctor clicks Yes button

5. Doctor click No button

6. Doctor clicks on the Exit button

6. Medication\_History pop-up destroyed. (Destroy())

3. Drug is checked to see if it conflicts with another drug, if so Drug\_Conflict\_Warning() pop-up is created. (Drug-Check())

4. Drug\_Conflict\_Warning() pop-up destroyed.. Update()

1. Medication\_History() pop-up is created. (Pop-up())

5. Drug\_Conflict\_Warning() pop destroyed, dataase not updated

2. Prescription field is updated. (Update())

**Use Case 07: Patient\_Critical\_Alarm**

Actions Systems Boundary Desktop View Update View Update View Update View

1. User clicks on the Patient Panel that issued the alert.
2. User clicks on Condition button.

1. User hits No button.
2. User hits Yes button.
3. User hits Back button.

1. Condition button is flashing red

for critical.

2. Dismiss\_Alert()

pop-up is created.

3. Dismiss\_Alert pop-up is destroyed and the Condition panel remains flashing red.

4. Dismiss Alert pop-up is destroyed, patient's condition is still critical then the panel will remain orange until their condition changes.

5. User is returned to Home\_GUI.

**Use Case 08: Doctor\_Page\_Alert**

Systems Boundary Desktop View Update\_View Update\_View Database

1. Patient\_Display\_View

Actions is created.

3. Address field is updated with the new recipient.

1. User with Nurse permission clicks red colored panel in Home\_GUI.
2. User clicks the Page Doctor button.
3. User clicks Address field and selects a new Doctor as recipient.
4. User edits the Message field.
5. User clicks Send Page button.
6. User with Doctor permission clicks the yellow colored panel in Home\_GUI.
7. User clicks Confirm button.

5. Patient database is updated with the new page,and an alert is displayed on the screen of the chosen Doctor, and Page\_Send pop-up is destroyed.

2. Page\_Send pop-up is created, with premade message in the Message field and the patient’s Doctor in the Address field, the Address field can be changed by using a dropdown tab.

4. Message field is updated.

6. Patient\_Display\_View

and Doctor Page pop-up

are created, if the User is

already in Patient\_

Display\_View then only

the Nurse\_Page pop-up

is created.

7. Nurse\_Page destroyed.

**Use Case 09: Pharmacy\_Login**

Systems Boundary Desktop View Update View Database Update View Update View

Actions

3 The software checks the database for a matching username/password pair, if no pair is found then the Login\_GUI is refreshed and the User is prompted to reenter their username and password.

6 The software checks the database for a matching username/password pair, if one is found then the Login\_GUI is destroyed, the Pharm\_Home\_GUI is displayed, and the User is given access to data according to their permissions.

Pharm\_Home\_GUI created

7. The User clicks the Sign Out button.

6. The User clicks Sign In button.

5. The User reenters their password

4. The User reenters their username.

2. The User enters their password.

3. The User clicks the Sign In button.

1. The User enters their username.

The Login\_GUI() is displayed.

5. Password field() is updated.

4. Username field() is updated.

1 Username field() is updated.

2 Password field() is updated.

7. User is returned to the main Login Screen.

**Use Case 10: Fill\_Prescriptions**

1 Prescription\_View pop-up is created and shows the User the list of requested prescriptions.

Actions Systems Boundary Desktop View Update View Database Update View

1. User clicks the Fill Prescriptions button.

The Pharm\_Home\_GUI is displayed.

2. User clicks one of the Fill Prescription buttons when that particular prescription is ready.

3. User clicks the Exit button.

2 The prescription to the left of the button is filled, and an alert sent to the patient's Nurse station to notify them that the prescription is ready, and the patient records database is updated.

3 Prescription\_View pop-up is destroyed.

**UC 11: Drug\_Database\_Query\_Modify**

Actions System Boundary Pharm\_Home\_GUI Drug\_Database\_View Fields Updated Database

1. User clicks the Drug Database button.
2. Drug\_Database\_View is created
3. User enters the drug name into the Search field.
4. User clicks the Search button and the database is queried for the entered drug name.
5. Search results are displayed.
6. User clicks a search result.
7. User clicks Edit Information button.
8. User changes field information.
9. User clicks Submit and the new information is updated in the database.
10. User clicks Cancel button and the Drug\_Database\_View is cleared but not destroyed.
11. User clicks Exit button.

3. Update()

1. User clicks Drug Database button.

2. Display()

4. Find\_Value()

4. User clicks Search.

6. User clicks a search result.

5. Return\_Value ()

6. Display()

7. User clicks Edit Information button.

8. Update()

9. User clicks Submit button.

9. Enter\_Value()

10. Refresh()

11. Destroy()

**UC 12: Drug\_Database\_New\_Entry**

Fields updated

6. Update()

Drug Database

10. Enter\_Value()

7. User clicks Cancel button.

8. Destroy()

9. User clicks Submit button.

System Boundary

Pharm\_Home\_GUI

1. User clicks Drug Database button.

Drug\_Database\_View

2. Display()

3. User clicks New\_Entry button.

4. Display()

5. User enters information into the form fields.

New\_Entry\_Form pop-up

1. User clicks Drug Database button.

2. Drug\_Database\_View is created.

3. User clicks New Entry button.

4. New\_Entry\_Form pop-up is created.

5. User fills out drug name, appearance, uses, sideaffects, and maximum dosage.

6. Fields are updated in the New\_Entry\_Form pop-up.

7. User clicks Cancel button.

8. Nothing is updated, New\_Entry\_Form pop-up is destroyed, and User is returned to Drug\_Database\_View.

9. User clicks Submit button.

10. Drug database is updated and User is returned New\_Entry\_Form pop-up is destroyed.

**UC 13: Administrator\_Login**

6. Check\_Database()

7. Display()

8. Destroy()

9. User clicks Sign In button.

10. Check\_Database()

11. Refresh()

1. The User enters their username.

2. Username field is updated.

3. The User enters their password.

4. Password field is updated.

5. The User clicks the Sign In button.

6. The software checks the database for a matching username/password pair

7-8. Username/password pair is found in database, Login\_GUI is destroyed and Admin\_View is created.

9.User clicks Sign In button.

10-11. If the username/password pair is incorrect, then Login\_GUI is refreshed and the User is prompted to reenter their username and password.

5. User clicks Sign In button.

4. Update()

3. User enters their password.

2. Update()

1. User enters their username.

Admin\_View

Login Database

Fields Updated

Login\_GUI

System Boundary

**UC 14: Login\_Database\_Add\_User**

7. Update()

2. Display()

1. The User clicks Create New User button.

6. Update()

5. Update()

4. Update()

3. User enters new User's information.

Actions

Login Database

Fields Updated

New User pop-up

Admin\_GUI

System Boundary

1. The User clicks Create a new User button.

2. New\_User pop-up is displayed showing fields to add a new user (Username, Password, User's name, and permission).

3. The User enters information into the username, password, User's name, and permission fields.

4-7. Enter values for the login database.

8. User clicks the Submit button.

9-12. Information is added to the database.

13. New\_User pop-up is refreshed.

14. The User clicks the Cancel button.

15. The software returns user to the to the Administrator to Admin\_View without updated the database.

9. Enter\_Value(Username)

8. User clicks Submit button.

10. Enter\_Value(Password)

11. Enter\_Value(Employee\_Name)

12. Enter\_Value(Permission)

13. Refresh()

15. Destroy()

14. User clicks Cancel button

**UC 15: Login\_Database\_Query\_Modify**

System Boundary

Admin\_GUI

User Search pop-up

Fields Updated

Login Database

1. The User clicks Edit Current User button.

2. User\_Search pop-up is displayed.

3. The User enters a User's name into the search field.

4. The search field is updated.

5. The User hits the Search button.

6.Database is searched by User's name.

7. User's information is returned and displayed in the User\_Search pop-up.

8. The User clicks Delete User button.

9-10. The deleted User is removed from the database and User\_Search pop-up is destroyed.

11. The User edits the information in the Name, Username, Password, and Permission fields.

12. The four fields are updated.

13. The User clicks the Submit Changes button.

14. The login database is updated with the new information.

15. The User clicks the Exit button.

16. User\_Search pop-up is destroyed.

1. User clicks Edit Current User button.

2. Display()

4. Update()

3. User enters a User's name into the search field.

5. User clicks Search button.

6. Database is searched by User's name.

7. Display()

8. The User clicks Delete User button.

9. Delete(Selected\_User)

10. Destroy()

11. User edits information in the Name, Username, Password, and/or Permission fields.

12. Update()

13. User clicks Submit button.

14. Enter\_Value()

16. Destroy()

15. User clicks Exit button.

* 1. **Category Interaction Diagram**

**CID Use Case 01**

Doctor

Administrator

Doctor view

Administrator view

Login

Nurse

Pharmacist

Pharmacy view

Nurse view

Patient

Drug

Pharmacy

4. Object Design

**User Object**

//Each User object represents one authorized User of the software. It has four sub-classes, each with

//their own user information and schedule.

**public** **class** User {

**private** String username;

**private** String password;

**private** String firstName;

**private** String lastName;

**private** String[] schedule = **new** String[7];

**public** String getusername() {

**return** username;

}

**public** String getpassword() {

**return** password;

}

**public** String getFirstName() {

**return** firstName;

}

**public** String getLastName() {

**return** lastName;

}

//method allows an Administrator to change a User's username.

//Precondition: The new username is passed to the object as a string

//Postcondition: The object's variable username is changed to reference the new username

**public** **void** changeUsername(String newUsername) {

username = newUsername;

}

//method allows an Administrator to change a User's password.

//Precondition: The new password is passed to the object as a string

//Postcondition: The object's variable password is changed to reference the new password

**public** **void** changePassword(String newPassword) {

password = newPassword;

}

//method allows an Administrator to change a User's first name.

//Precondition: The new first name is passed to the object as a string

//Postcondition: The object's variable firstName is changed to reference the new first name

**public** **void** changeFirstName(String newFirstName) {

firstName = newFirstName;

}

//method allows an Administrator to change a User's last name.

//Precondition: The new last name is passed to the object as a string

//Postcondition: The object's variable lastName is changed to reference the new last name

**public** **void** changeLastName(String newLastName) {

lastName = newLastName;

}

//method allows the modification of the array containing the User's schedule

//Precondition: The element index and new string are passed to the method. The index

//number represents Sunday for 0, Monday for 1, and so on up to 6 for Saturday.

//Postcondition: The array is changed to reflect the User's updated schedule.

**public** **void** changeSchedule(**int** i, String str) {

schedule[i] = str;

}

}

**import** java.util.\*;

**Patient Object**

//Each Patient object represents one patient in the hospital. Each object contains all of the patient's

//information, such as their name, phone number, address, social security number, room number, the

//doctor assigned to them, the nurse station corresponding to their room number, previous medical

//diagnoses, previous and current medications, the times of their next scheduled checkup and

//medication, and so on.

**public** **class** Patient {

**private** String firstName;

**private** String lastName;

**private** **int** phoneNumber;

**private** String address;

**private** String insuranceProvider;

**private** **int** SSN;

**private** String doctor;

**private** **int** nurseStation;

**private** **int** roomNumber;

**private** String currentDiagnosis;

**private** LinkedList medications = **new** LinkedList(); //stores drug objects of each drug that the //patient is currently taking

**private** LinkedList medicalHistory = **new** LinkedList(); //stores each each diagnosis from most //recent to least

**private** **double** nextCheckup; //time of next checkup

**private** **double** nextMedication; //time next medication will be administered

**private** **double** lastMedication; //time last medication was administered

**public** String getFirstName() {

**return** firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **int** getPhoneNumber() {

**return** phoneNumber;

}

**public** String getAddress() {

**return** address;

}

**public** String getInsuranceProvider() {

**return** insuranceProvider;

}

**public** **int** getSSN() {

**return** SSN;

}

**public** String getDoctor() {

**return** doctor;

}

**public** **int** getNurseStation() {

**return** nurseStation;

}

**public** **int** getRoomNumber() {

**return** roomNumber;

}

**public** String getCurrentDiagnosis() {

currentDiagnosis = (String) medicalHistory.getFirst();

**return** currentDiagnosis;

}

**public** LinkedList getMedications() {

**return** medications;

}

**public** **double** getNextCheckup() {

**return** nextCheckup;

}

**public** **double** getNextMedication() {

**return** nextMedication;

}

**public** **void** changeFirstName(String newFirstName) {

firstName = newFirstName ;

}

**public** **void** changeLastName(String newLastName) {

lastName = newLastName;

}

**public** **void** changePhoneNumber(**int** newPhoneNumber) {

phoneNumber = newPhoneNumber;

}

**public** **void** changeAddress(String newAddress) {

address = newAddress;

}

**public** **void** changeInsuranceProvider(String newInsuranceProvider) {

insuranceProvider = newInsuranceProvider;

}

**public** **void** changeSSN(**int** newSSN) {

SSN = newSSN;

}

**public** **void** changeDoctor(String newDoctor) {

doctor = newDoctor;

}

**public** **void** changeNurseStation(**int** newNurseStation) {

nurseStation = newNurseStation;

}

**public** **void** changeRoomNumber(**int** newRoomNumber) {

roomNumber = newRoomNumber;

}

}

//This method adds a new drug to the linked list of drugs that a patient is taking

//Precondition: When the Doctor creates a new prescription and sends it to the

//pharmacy, the prescription is also passed to this method.

//Postcondition: The linked list of drug objects now has the new prescription’s drug as its first

//value.

public void newMedication(Prescription p) {

newMedication = new Drug p.getPrescribedDrug();

medications.addFirst(newMedication);

}

//This method updates a patient's current diagnosis. Before the diagnosis is changed, the //current diagnosis is added to the patient's medical history.

//Precondition: The Doctor enters a new diagnosis.

//Postcondition: Medical History is updated with the old diagnosis, and the new current //diagnosis becomes currentDiagnosis.

public void changeCurrentDiagnosis(String newCurrentDiagnosis) {

medicalHistory.addFirst(newCurrentDiagnosis);

}

//this method changes the time of the next checkup by hospital staff.

public void changeNextCheckup(double newNextCheckup) {

nextCheckup = newNextCheckup;

}

//this method changes the time when the next medication will be administered to the patient.

public void changeNextMedication(double newNextMedication) {

nextMedication = newNextMedication;

}

**Drug Object**

//Each drug object represents a drug in the database and stores the drug’s name, uses, sideaffects, //maximum safe dose, duration of the effects, whether or not the drug is a controlled substance, what //other drugs and medications it conflicts with, and how much is currently on hand at the pharmacy.

Public class Drug {

String name;

             String uses;

             String sideaffects;

             int quantity;              //amount of the drug on hand

             int maxDose;             //in milligrams

             float duration;            //in hours

             LinkedList conflictingDrugs = new LinkedList(); //linked list of incompatible medications //(strings)

boolean controlled; //whether or not it is a controlled substance (illegal or                                                        //requires prescription)

public String getName() {

                      return name;

}

             public String getUses() {

                return uses;

}

             public String getSideaffects() {

                      return sideaffects;

}

public int getQuantity() {

return quantity;

             }

public int getMaxDose() {

                         return maxDose;

}

public float getDuration() {

                       return Duration;

}

//returns a pointer to the linked list

Public getConflictingDrugs() {

LinkedList conflicts = conflictingDrugs;

Return conflicts;

}

public boolean isControlled( ) {

                        return controlled;

}

}

**Prescription Object**

//Each Prescription object represents a prescription filled out by the Doctor and sent to the Pharmacy

//to be filled. Each object contains the drug the drug the prescription is for, the issuing Doctor, the

//patient the prescription is for, the time it was prescribed, the dose per day, and the duration of the //prescription. It also provides a method to send an alert if the prescribed medication conflicts with //anything the patient is already taking.

Public class Prescription {

prescribedDrug = new Drug();

patientDoctor = new User(); //Doctor issuing the prescription

forPatient = new Patient(); //patient the prescription is for

float timePrescribed;

int dosePerDay; //in milligrams

int prescriptionDuration; //in days

boolean ready; //default is false

public getPrescribedDrug() {

return prescribedDrug;

}

public getPatientDoctor() {

return patientDoctor;

}

public getTimePrescribed() {

return timePrescribed;

}

public getDose() {

return dosePerDay;

}

public getDuration() {

return duration;

}

//This method checks if the prescribed drug is compatible with the patient’s current medication, and if //it isn’t a warning will be shown on the Doctor’s screen.

//Precondition: The Doctor selects the drug to be prescribed to the patient, and the selected drug is

//displayed on the GUI.

//Postcondition: If the drug is incompatible with any current medications, this method will tell the GUI //to display an Incompatibility Alert to the Doctor.

Public drugConflictCheck() {

LinkedList medication = forPatient.getmedication(); //list of patient meds

String newDrug = prescribedDrug.getName(); //drug being prescribed

int numMeds = medication.size();

A: for(i = 0, i < numMeds, i++) {

Drug conflict = medication.get(i); //drug to be checked

LinkedList conflicts = conflict.getConflictingDrugs(); //list of drug conflicts

numConflicts = conflicts.getSize();

B: for(j = 0, j < numConflicts, j++) {

String check = conflicts.get(j);

int conflictCheck = newDrug.compareTo(check);

C: if(conflictCheck == 0) {

//code to create alert on GUI

break A;

}

}

}

}

//This method alerts the Pharmacist that a new prescription is in the Pharmacy database

//Precondition: Doctor submits a new prescription, creating a new prescription object and adding it to //the Pharmacy database.

//Postcondition: The new prescription is displayed in the pharmacy GUI

public class prescriptionListener implements ActionListener {

public void newPrescriptionAlert(ActionEvent event) {

if(submitPrescription == true)

addPrescription(); //adds prescription to pharmacy database

}

}

//This method marks prescription as filled and ready for pickup

//Precondition: The Pharmacist clicks the filled button on the Pharmacy GUI

//Postcondition: The filled prescription is marked as filled and removed from the pharmacy GUI

public class fillListener implements ActionListener {

public void markFilled(ActionEvent event) {

if(fillClick == true)

prescriptionReadyAlert();

//remove prescription from list in Pharmacy GUI

}

}

//method updates the nurse station the prescription is ready

//precondition nurse station has no alerts

//postcondition nurse station has an alert reading prescription is ready in the pharmacy

public void prescriptionReadyAlert() {

//display Filled\_Prescription\_pop-up on Nurse GUI

}

**GUI Object**

//The GUI class creates and maintains the graphical user interface, and acts as a go-between for the //other classes, calling methods that display information from the other classes.

public class GUI {

//Each object represents something on the main login GUI screen

private Label lblScreen; // declare component Label

private TextUserF tfUser; // declare component Username Field

private TextPassF tfPass; // declare component Password Field

private Button1 btnLogin; // declare component Login Button

private Button2 btnCancel; // declare component Cancel Button

/\*\* Constructor to setup GUI components \*/

public GUIInterface () {

setLayout(new GuiLayout()); //"this" Frame sets its layout to //GuiLayout, which arranges the components

//from left-to-right, and flow to next row

//from top-to-bottom.

setTitle("Super Hospital Systems"); // "this" Frame sets title

setSize(width, height); // "this" Frame sets initial window size

setVisible(true); // "this" Frame shows

setUserName("Username"); // "this" Frame sets username field

setUserPass("Password"); // "this" Frame sets password field

}

}

//Method listens for User’s sign after entering username and

//password.

//Precondition: The User has entered the username and password

//Postcondition: The User is prompted with a message that notifies

//them if they have entered the correct or incorrect information

public class LoginListener implements ActionListener{

public void actionPerformed(ActionEvent event){

if(username.equals(txtUsername.getText())){

if(password.equals(txtPassword.getText())){

msg = "Login Granted!";

}else{

msg = "Access Denied";

}

}else{

msg = "Access Denied";

}

}

}

//Method listens for User to press cancel buttons

public class CancelListener implements ActionListener{

public void actionPerformed(ActionEvent event){

txtUsername.setText = " ";

txtPassword.setText = "";

txtUsername.requestFocus();

}

}

public String getUsername() {

return tfUser;

}

public String getUsername() {

return tfPass;

}

5. Test Cases

|  |  |
| --- | --- |
| Drug Object |  |
| Attributes | Description |
| Name | add a new drug to the system |
| Location | addNew() |
| Input | Name1 |
| Input | Things |
| Input | 50 |
| Input | 5 |
| Input | 2 |
| Oracle | New drug Name1 has been added to the database |
| Log | The drug was successfully added |
| State Based Test | Test ability to add new drugs |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | edit an existing drug |
| Location | editDrug() |
| Input | Name1 |
| Input | 1 |
| Input | Namen |
| Oracle | New drug Name1 has been found and is now Namen |
| Log | The drug was successfully added |
| State Based Test | Test ability to edit a drug |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | edit a drug not in the system |
| Location | editDrug() |
| Input | Name1 |
| Input | 1 |
| Input | Namen |
| Oracle | "Drug Not Found" error appears |
| Log | "Drug Not Found" error appears then repropts user for a drug name |
| Change | after error message user is dumped to the main screen |
| State Based Test | test ability to find if drug exists |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | View a list of all drugs |
| Location | showList() |
| Input | N/A |
| Oracle | List of Drug names is printed to the user |
| Log | The list of drug names was sucessfully printed |
| State Based Test | test ability to show all drugs |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | Remove an existing drug 'Name1' from the system |
| Location | removeDrug() |
| Input | Name1 |
| Oracle | drug Name1 is removed from the database |
| Log | The drug Name1 was successfully removed |
| State Based Test | test ability to remove drugs |
| Attributes | Description |
| Name | Remove an non-existant drug from the system |
| Location | removeDrug() |
| Input | NotHere |
| Oracle | "Drug Not Found" error displays |
| Log | "Drug Not Found" error displayed |
| State Based Test | test ability to find if drug exists before deletion |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | View a list of all drugs and their stock |
| Location | showStock() |
| Input | N/A |
| Oracle | List of Drug names is printed with drug stock |
| Log | The list of drug names with stock was sucessfully printed |
| State Based Test | tests ability to show the full drug list with stock of each drug |

|  |  |
| --- | --- |
| Perscription Object |  |
| Attributes | Description |
| Name | perscribe an existing drug |
| Location | perscribeNew() |
| Input | Name1 |
| Input | 25 |
| Input | 10 |
| Oracle | Drug has been perscribed and is added to pharmacy list and quantity is reduced |
| Log | The perscription was successfully added to pharmacy list and quantity is reduced |
| State Based Test | test ability to perscribe a drug |
| Attributes | Description |
| Name | perscribe a non-existant drug |
| Location | perscribeNew() |
| Input | NotHere |
| Oracle | "Drug Not Found" error |
| Log | "Drug Not Found" error is shown |
| State Based Test | test to see if the drug exists before continuing with percription |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | perscribe too much of an existing drug and continue |
| Location | perscribeNew() |
| Input | Name1 |
| Input | 55 |
| Input | y |
| Input | 10 |
| Oracle | Dose too high error is shown but user can override to create the perscription and quantity is reduced |
| Log | The perscription was successfully added to pharmacy list even with too high of a doseage and quantity is reduced |
| Boundry Test | tests to make sure overdoeses are detected but can still be overridden |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | perscribe too much of an existing drug and user exits |
| Location | perscribeNew() |
| Input | Name1 |
| Input | 55 |
| Input | y |
| Input | 10 |
| Oracle | Dose too high error is shown but user can override to create the perscription and quantity is reduced |
| Log | The perscription was successfully added to pharmacy list even with too high of a doseage and quandity is reduced |
| Boundry Test | test to make sure overdoses are detected |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | perscribe an existing drug that is out of stock |
| Location | perscribeNew() |
| Input | Name1 |
| Input | 25 |
| Input | 10 |
| Oracle | Out of stock error displays and perscription is cancled |
| Log | Out of stock error is displayed and perscription was cancled |
| Boundry Test | test to make sure the drug is in stock so it can be perscribed |

|  |  |
| --- | --- |
| Perscription Object |  |
| Attributes | Description |
| Name | create a new perscription valid perscription |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Input | 50 |
| Input | 50 |
| Oracle | drug1 is in patient1's perscribed list and the pharmacy list |
| Log | The drug was successfully added |
| State Based Test | Test ability to perscribe a drug |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new perscription for out of stock drug |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Oracle | perscription is canled error message is displayed |
| Log | Error message but the drug was successfully added with a resulting stock level of -1 |
| State Based Test | Test ability to catch out of stock drugs |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new perscription for out of stock drug v2 |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Oracle | perscription is canled error message is displayed |
| Log | Error message displayed and the persciption was cancled |
| State Based Test | Test ability to catch out of stock drugs |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new overdose perscription with override |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Input | 500 |
| Input | y |
| Oracle | drug1 is in patient1's perscribed list and the pharmacy list with a dose higer than max dose |
| Log | The drug was successfully added with exsessive dose |
| State Based Test | Test ability to catch overdoses and override them |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new overdose perscription without override |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Input | 500 |
| Input | n |
| Oracle | perscription is cancled and stock remains unchanged |
| Log | perscription has been cancled and stock remains unchanged |
| State Based Test | Test ability to catch perscribed overdoses |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new invalid perscription with override |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Input | 50 |
| Input | y |
| Oracle | drug1 is in patient1's perscribed list and the pharmacy list with an invalid drug |
| Log | drug1 is now in patient1's perscribed list and the pharmacy list with an invalid drug |
| State Based Test | Test ability to catch incompatible perscriptions and override the warning |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new invalid perscription without override |
| Location | perscribeNew() |
| Input | patient1 |
| Input | drug1 |
| Input | 50 |
| Input | n |
| Oracle | perscription is cancled and stock remains unchanged |
| Log | perscription has been cancled and stock remains unchanged |
| State Based Test | Test ability to catch incompatible perscriptions |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new perscription with unknown patient |
| Location | perscribeNew() |
| Input | patient1 |
| Oracle | Error message is displayed and perscription is cancled |
| Log | perscription has been cancled and error message appears |
| State Based Test | Test ability to find patient to perscribe medication |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | create a new perscription with unknown drug |
| Location | perscribeNew() |
| Input | patient1 |
| input | drugX |
| Oracle | Error message is displayed and perscription is cancled |
| Log | perscription has been cancled and error message appears |
| State Based Test | Test ability to find a drug to add a perscription |
| Attributes | Description |
| Name | fill a perscription with confirm |
| Location | fillPerscription() |
| Input | patient1 |
| input | y |
| Oracle | perscription is removed from the persription list and is placed in the filled list |
| Log | perscription has been removed from prescription list and is now in the filled list |
| State Based Test | Test ability to fill a percription |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | fill a perscription with negation |
| Location | fillPerscription() |
| Input | patient1 |
| input | n |
| Oracle | perscription is still on perscription list and user is returned |
| Log | user is returned to main screen and perscription is still on perscription list |
| State Based Test | Test ability to fill percriptions |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | show the pharmacy list |
| Location | showPharmList() |
| Oracle | list of all perscriptions in the pharmacy is shown to the user |
| Log | all perscriptions in the pharmacy list are shown to user |
| State Based Test | Test ability to show the pharmacy list |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | show the filled perscription list |
| Location | showFilledList() |
| Oracle | list the perscriptions in the filled list to the user |
| Log | all filled perscriptions are shown to the user |
| State Based Test | Test ability to show filled perscriptions |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | administer perscription |
| Location | administerPerscription() |
| Oracle | the drug is now moved into the administered list |
| Log | drug now appears in the administered list |
| State Based Test | Test ability to adminster a perscription |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | remove a perscribed perscription |
| Location | removePerscription() |
| Input | patient1 |
| input | y |
| Oracle | drug is now removed from the perscription list and is also removed from patient's perscribed med list and one is added to drug stock |
| Log | perscription is removed stock is readded and patient's med list is revised |
| State Based Test | Test ability to remove a precription |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | add a new patient to the system with current medications |
| Location | addPatient() |
| Input | patient1 |
| input | Last |
| input | tel |
| input | Add |
| Input | inure |
| Input | 123 |
| Input | Dr. Test |
| Input | 100 |
| Input | y |
| Input | Drug2 |
| Input | n |
| Oracle | patient1 is added to patient list with values given for variables including a list of current drugs perscribed |
| Log | patient 1 has been added with the entered variables and has drug 2 in its current medications list |
| State Based Test | Test ability to add new patients |

|  |  |
| --- | --- |
| Attributes | Description |
| Name | add a new patient to the system without current medications |
| Location | addPatient() |
| Input | patient1 |
| input | Last |
| input | tel |
| input | Add |
| Input | inure |
| Input | 123 |
| Input | Dr. Test |
| Input | 100 |
| Input | n |
| Oracle | patient1 is added to patient list with values given for variables |
| Log | patient 1 has been added with the entered variables and its current medications list is empty |
| State Based Test | Test ability to add new patients |

6. Rationale for Project and Implementation

**Rational for Topic**

We have chosen this topic because keeping and storing information in a hospital is very important, but presenting that information in a clear easy to understand and use way is even more important. So we want to create an application that is capable of tracking all a patients’ important information, and making that information usable to both the doctors and nurses treating the patient. This combined with a method of tracking prescribed drugs and their interference with other drugs should help the doctors and nurses to better diagnose, treat, and provide care for the patients. This system is currently unavailable in the medical field and we believe it could be a valuable resource. We want to make hospitals a better place.

**Use Case Rationale**

(UC1 & UC8 & UC12) This use case is required so the program can give the proper functionality to the proper user, creating a much more efficient experience. The doctors can access information critical for them, as well as nurses and pharmacists seeing information relevant to their particular jobs.

(UC2) This use case is required so that doctors can add new patients into their list, and allows doctors to verify all patient information on written records is correct with the patient.

(UC3) This use case is required so that doctors and nurses are able to access general patient information that might be relevant for particular forms, such as insurance and emergency contact information.

(UC4) This use case is required so that the doctor may be able to revise the current status of a patient’s information, such as weight or allergies the patient may have not listed be for upon admittance.

(UC5) This use case is required so that a doctor may be able to diagnose a patient and store that diagnosis for later use, as well as revise a previous diagnosis that may have been incorrect. Then the entered diagnosis can been easily seen by both doctors and nurses so that the patient can be more effectively treated.

(UC6) This use case is required so that the doctor can send a new prescription directly to the pharmacy. Additionally, this use case will help prevent accidental overdoses as well as improper mixing of medications by checking the drug to be prescribed against the current drug database.

(UC7) This use case is required so that both doctor and nursing staff can quickly be alerted to a patient who is in critical need or urgent care. This will allow for quicker treatment of a patient who is in urgent need.

(UC9) This use case is required so that the nursing staff can quickly communicate with the doctor. However, this method is also better than the traditional pager method as doctors will see a full message on screen to give the doctor more information as to why the page was sent.

(UC11)This use case is required so that the pharmacy staff can see a list of the prescriptions ordered, the dosage, and what doctor ordered the prescription. By having this use case we are enabling prescriptions to be filled more quickly as they will populate in a list as soon as the doctor makes the order, and informing the nursing staff as soon as the prescription is filled.

(UC12 & UC13) This use case is required so that the doctor cannot prescribe medication that could be harmful if prescribed in the improper amount or if it were combined with any other medication that might cause adverse side effects. This use case also allows for drugs to be modified, and for new drugs to be prescribed as soon as they are entered into the database by the pharmacy. The drug database is the most important part of our program as there is currently no such database being used in hospitals today.

(UC14) & (UC15) This use case is required so that new users may be added to the user database, and current users can have their information changed. Only the administrator will have access to this use case as it provides a greater level of security to the system. Thus new hires that will have access to this system will have to be manually be entered into or edited by the system administrator.

(UC16) This use case is required so that hospital employee’s will be able to track which doctors, or nurses are working during a particular shift so that they can more efficiently share information pertaining to patient care.

**Rational for Software Databases**

For our program we will make use of SQLite, as this was one of the project specifications.

**Rational For Software Architecture Used**

This architecture will be especially useful for us as our program will be interactive and use 4 different views depending on the user’s permissions.

**Object Rationale**

Patient Object - We are using a Patient object that has the patient’s information stored in the database. By using this object the user can access and change the patients object information in the database.

Prescription Object - We are using a Prescription object which will access current medications for each patient. It will have the ability to access the drug object to determine the quantity on hand and determine if it conflicts with any drug the patient is currently receiving.

Drug Object - The drug object will hold all of the drugs information such as, conflicting drugs, quantity, uses, side effects, max dosage, etc. This object will be accessed by the prescription object and communicate back and forth.

User Object - The user object will hold the information for each user, doctors, nurses, administrators. It will have their login information, the user’s current permissions, and personal information. This object will have 4 subclasses for each user.

Alarm Object - The alarm object will only be used when a patient needs immediate assistance. The color of the patient’s panel will turn red signaling the doctors and nurses.

GUI Object - The GUI object is basically the interface where everything appears to the user. It has buttons and windows that the user sees and interacts with. The buttons will call a method when they are pushed.

7. Project Cost Analysis

**a. Function Point Analysis**

**Weighting Factor Estimation**

**Measurement Parameter count average**

Number of user inputs 12 x 4 = 48

Number of users outputs 12 x 5 = 60

Number of user inquiries 8 x 4 = 32

Number of files 5 x 10 = 50

Number of external interfaces 1 x 7 = 7

Total 197

**Adjustment Factor**

**Factor Value**

1. Backup & recovery 4

2. Data communications 2

3. Distributed Processing 0

4. Performance critical 4

5. Existing operating environment 3

6. On-line data entry 4

7. Input transaction over multiple screens 5

8. Master files updated on-line 3

9. Information domain values complex 5

10. Internal processing complex 5

11. Code designed for reuse 4

12. Conversion/installation in design 3

13. Multiple installations 5

14. Application designed for change 5

15. Complexity adjustment factor 1.17

F15 = 53.17

**Calculations**

FP = Total x (0.65 + 0.01 x F15)

FP = 197 x (0.65 + 0.01 x 53.17) = 197 x (1.1817) = 232.6949

Labor = $8000 per month (given

FP = 7 per month (given)

Cost per FP = $8000/7 = $1142.86 per FP

$1142.86 x 256.4289 FP = $266,051.98

**b. COCOMO**

**Multiplier** **Rationale** **Value**

Reliability High Risk to Human Life 1.4

Database Relatively Large Database / LOC ratio 1.16

Complexity Simple DBMS getting and checking methods 0.85

Timing Nominal use of processing time 1.0

Storage Program not memory intensive 1.0

Machine Indented for commercial application; stable 0.87

Turnaround Time expected <4hrs 1.0

Analysts Average Level Analysts 1.0

Programmers Programmers are within the 55th percentile 1.0

Experience Team has at least 1 yr experience 1.13

Experience Team has at least 1 yr experience 1.0

Experience Team has at least 1 yr experience with java 1.0

Practice We will be using routine programming practices 1.0

Tools We will be using Basic (midi/maxi) programming tools 1.0

Schedule We will be using the nominal schedule with no stretch out or acceleration 1.0

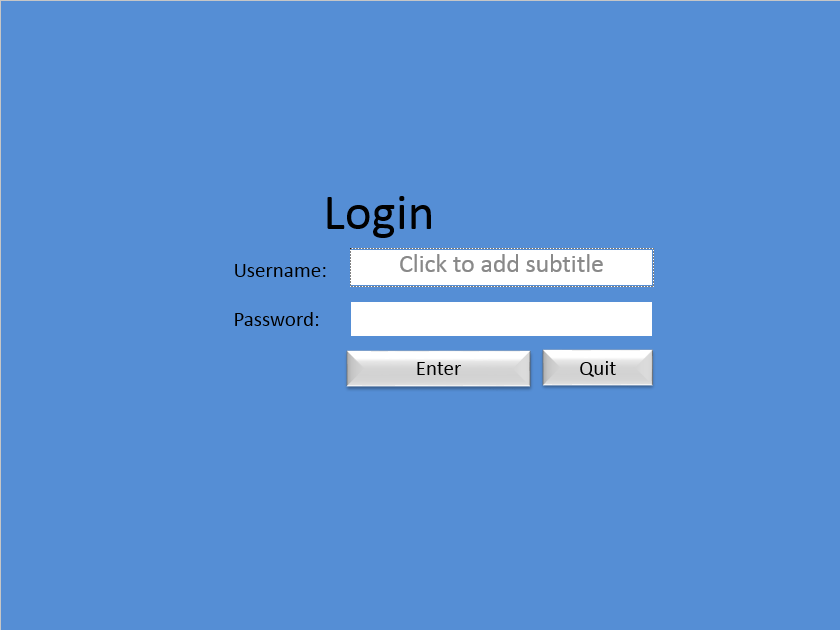
EAF = **1.36**

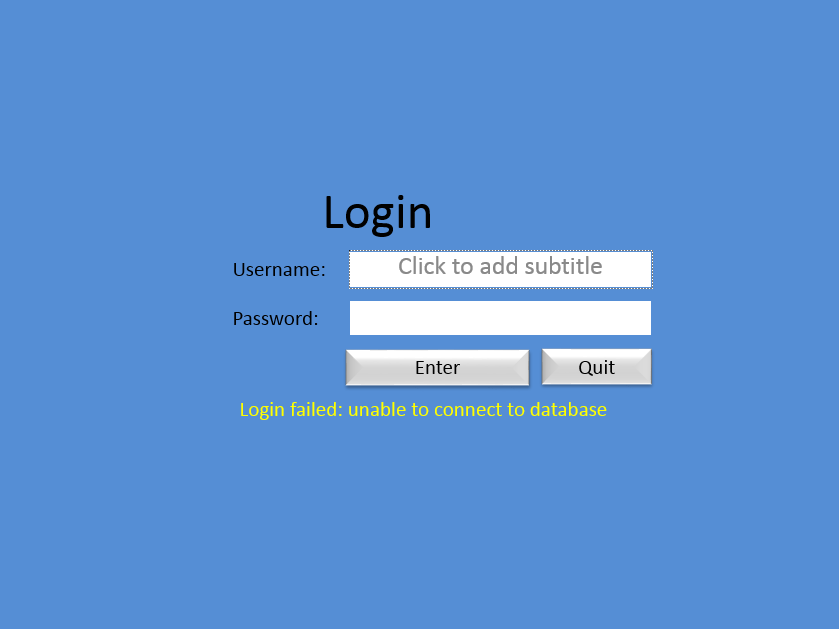
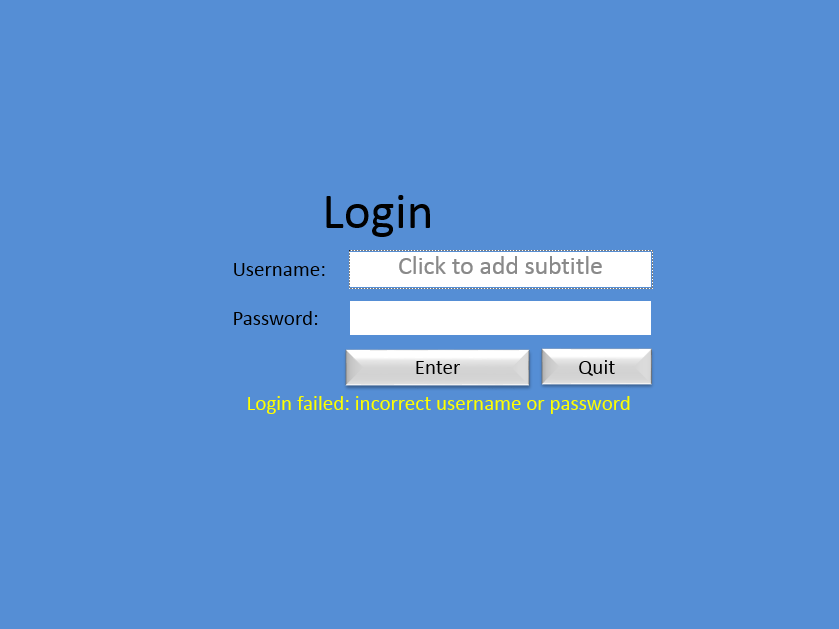
E = **28.56** Programmer/Months

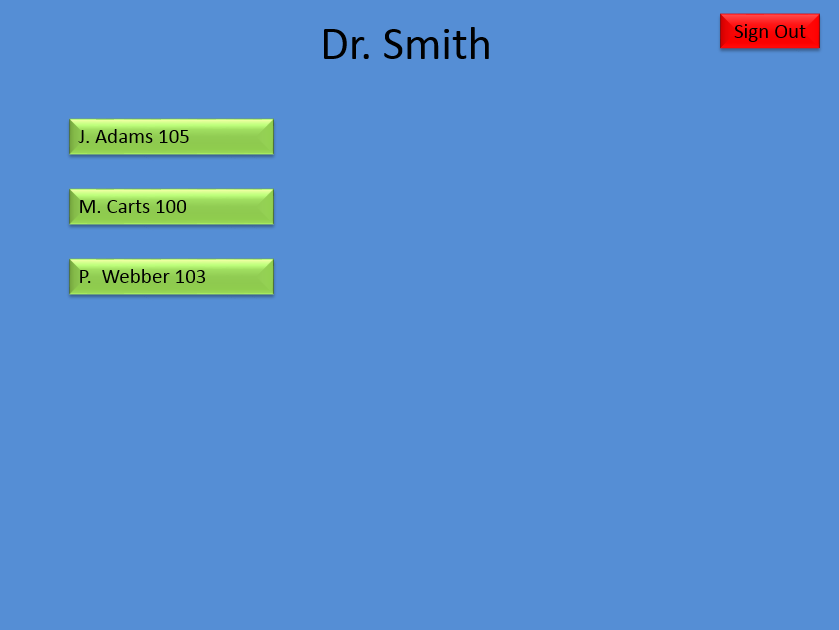
D = **8.94** Months of Development Time

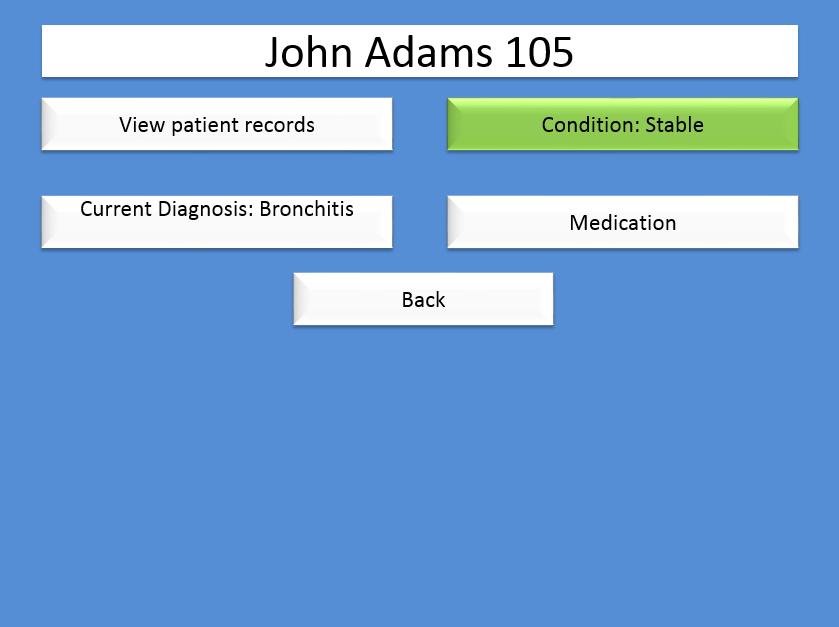
Cost($) = **$71,485.53**

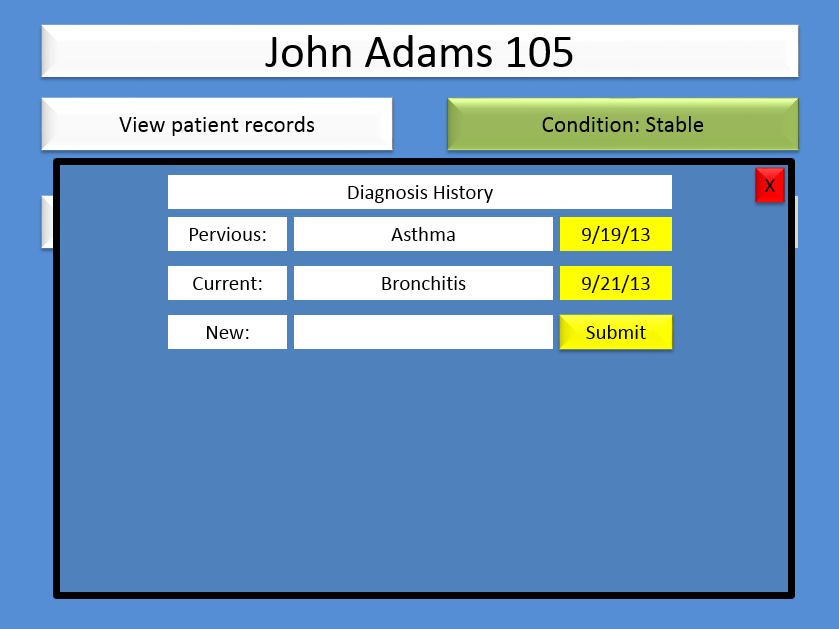
8. Horizontal Prototype

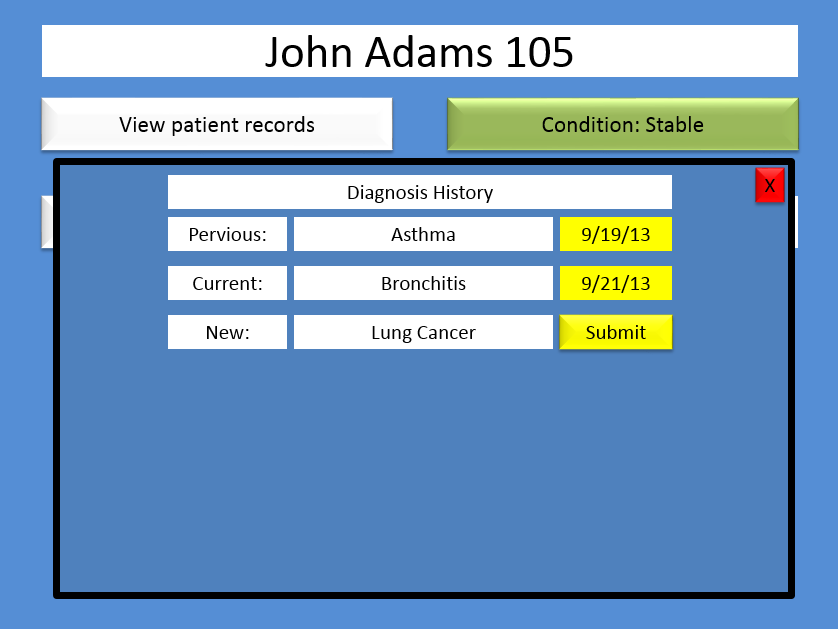


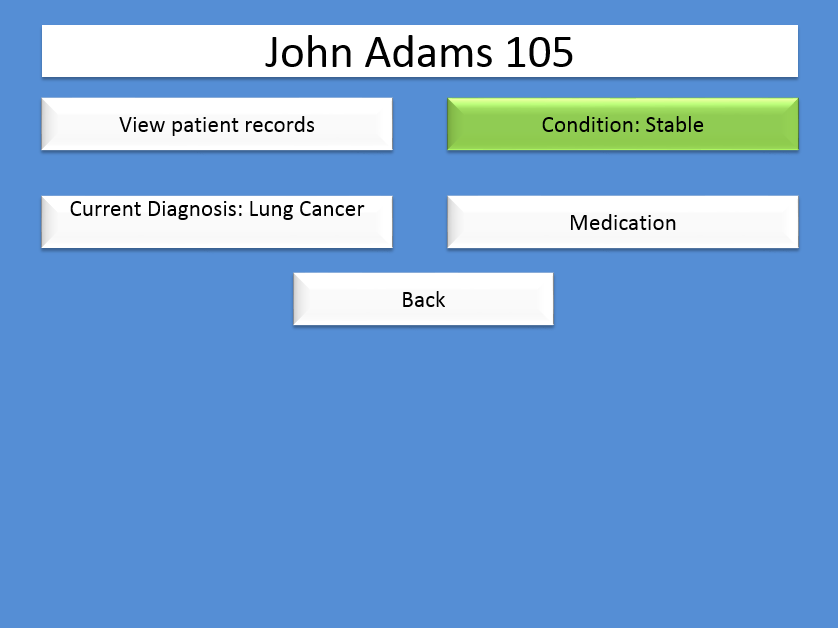


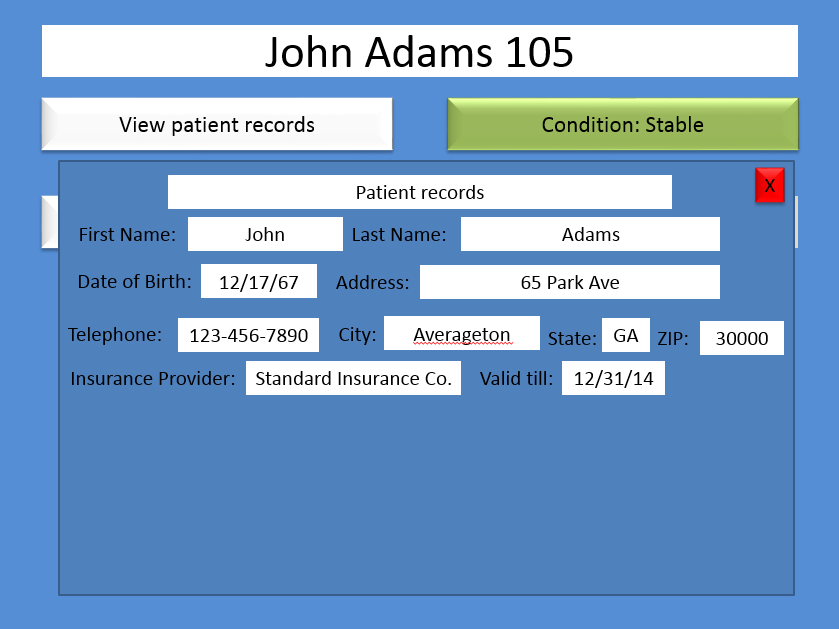


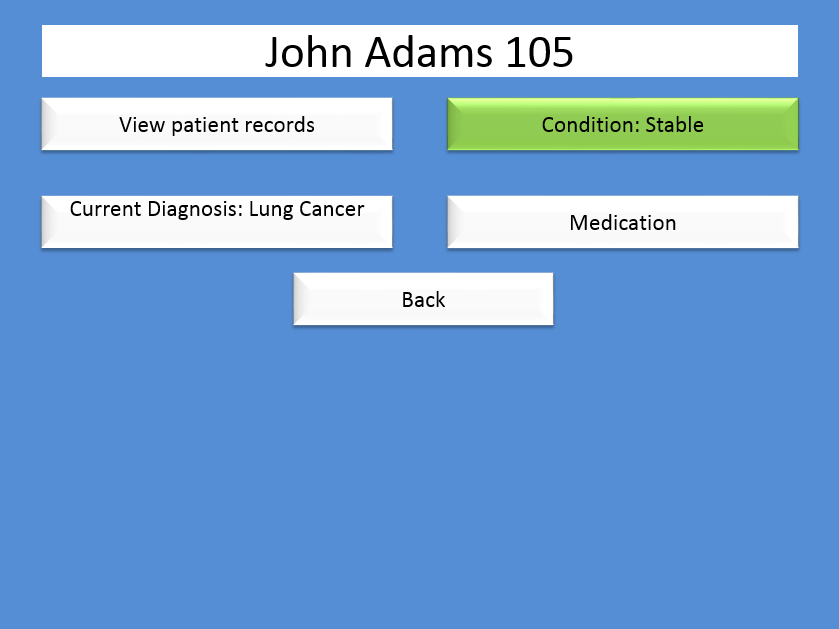


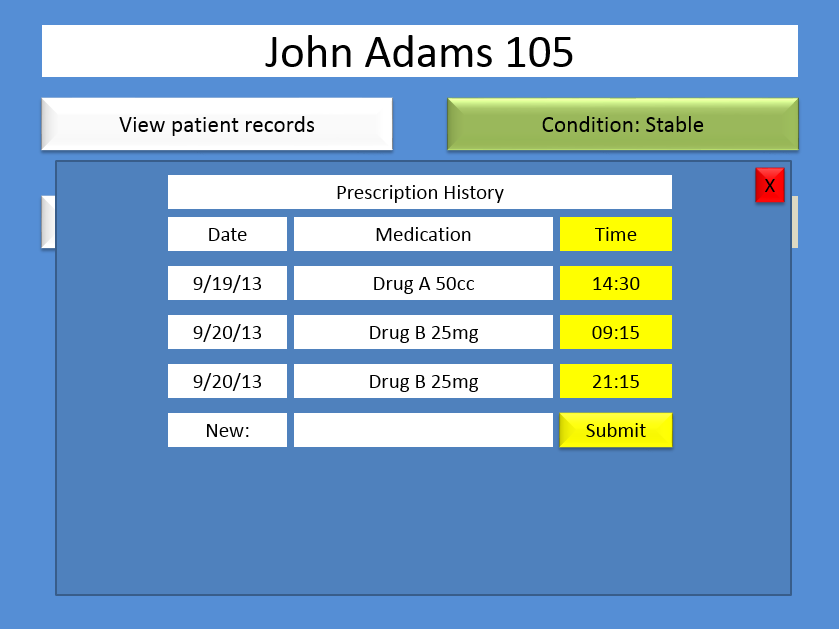


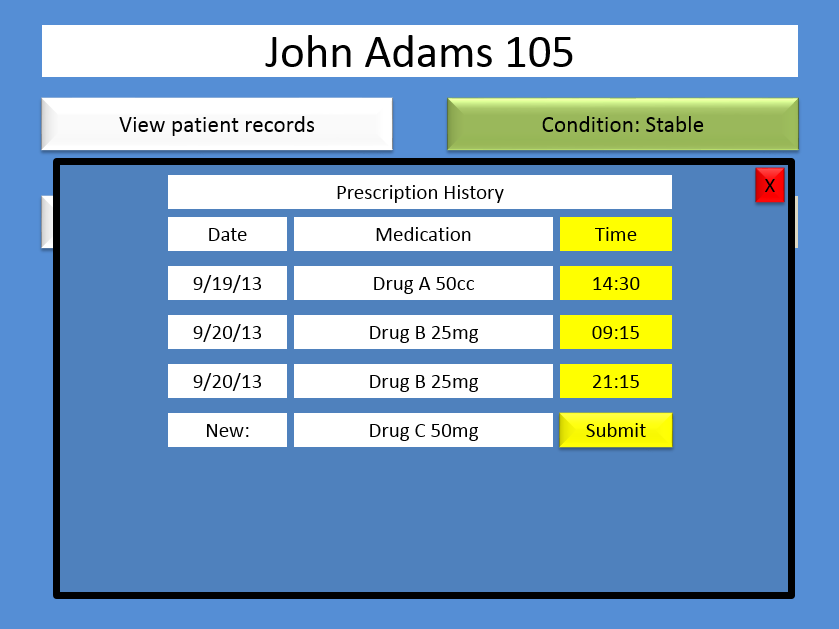


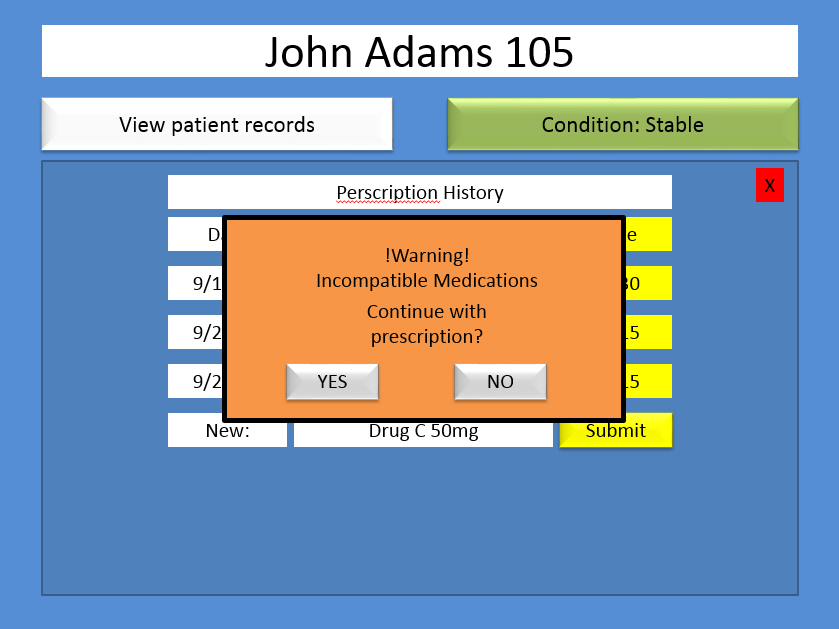


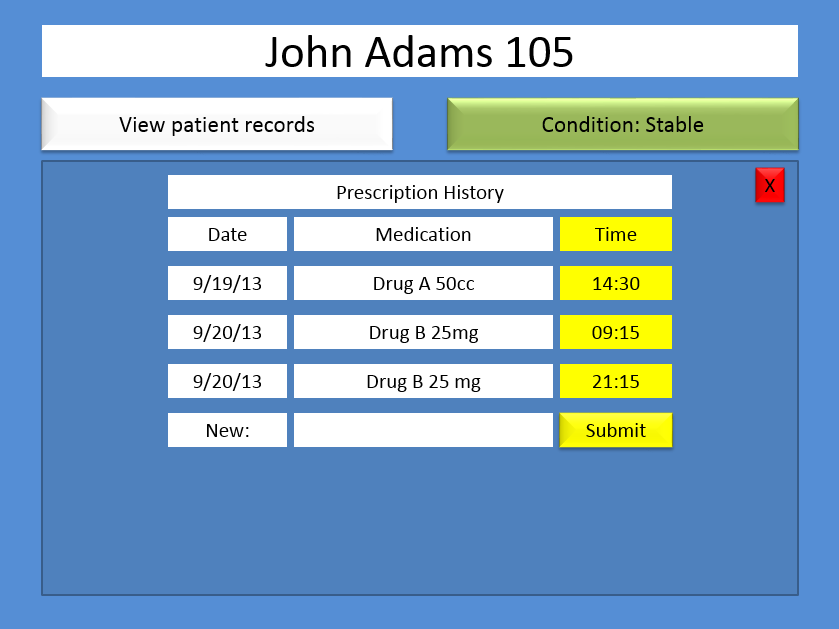


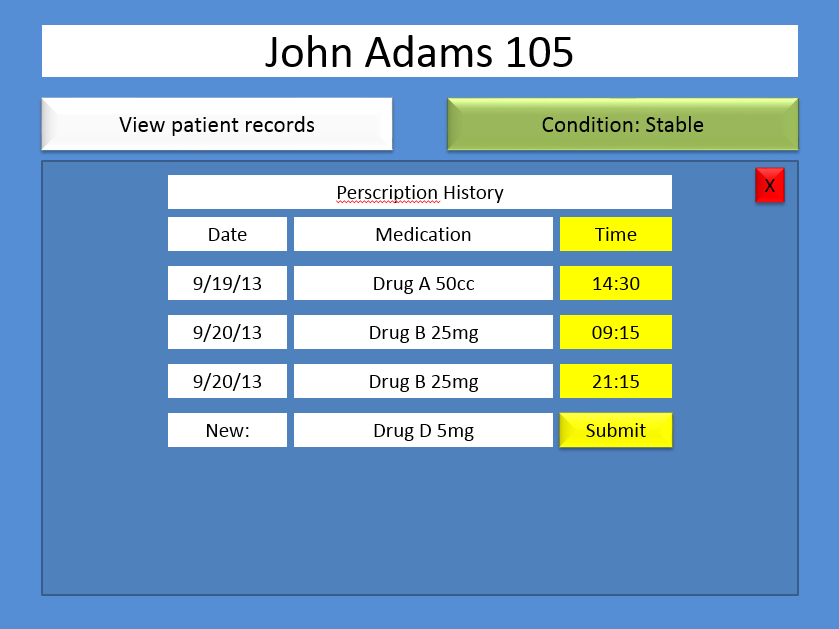


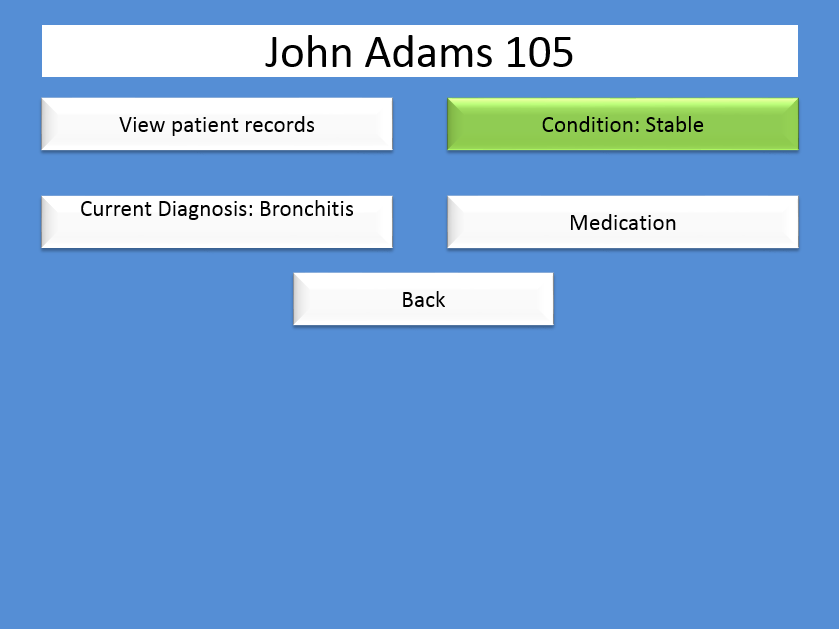


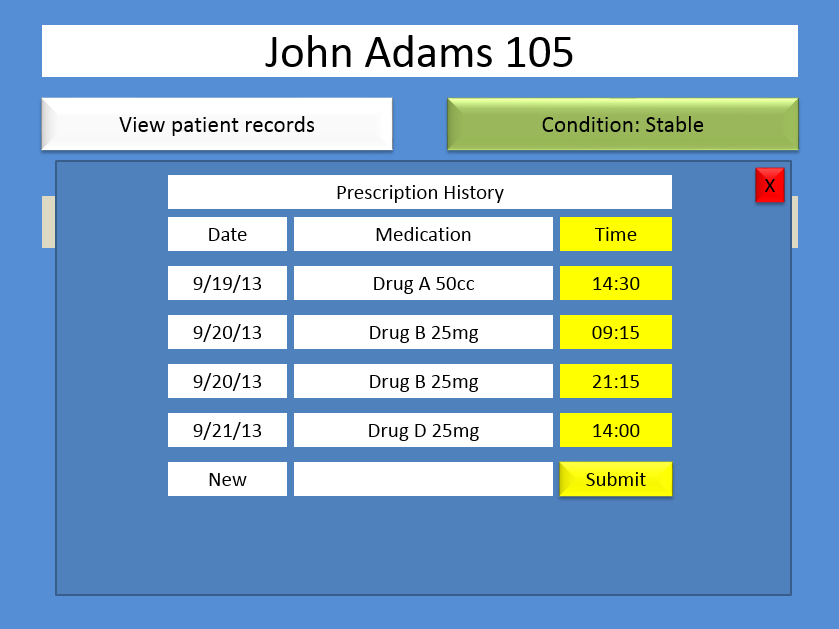


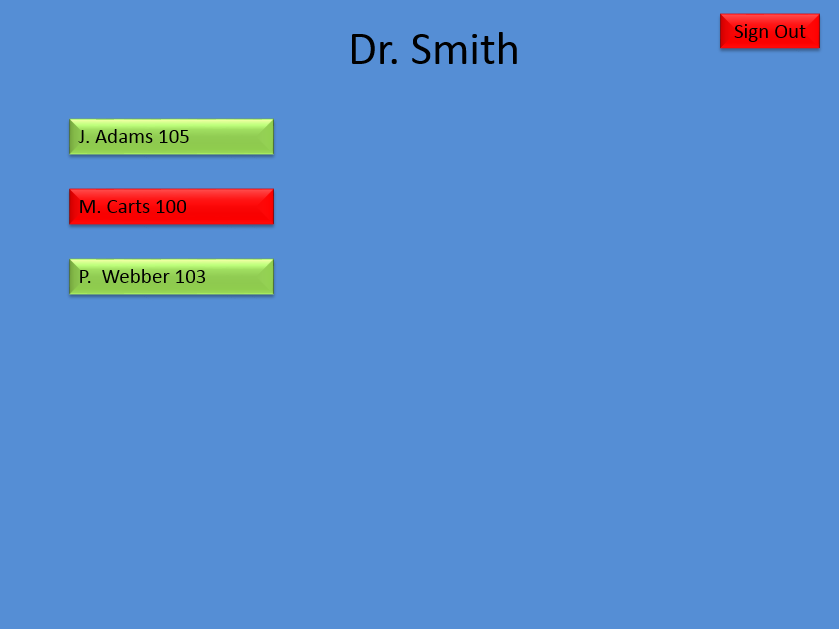


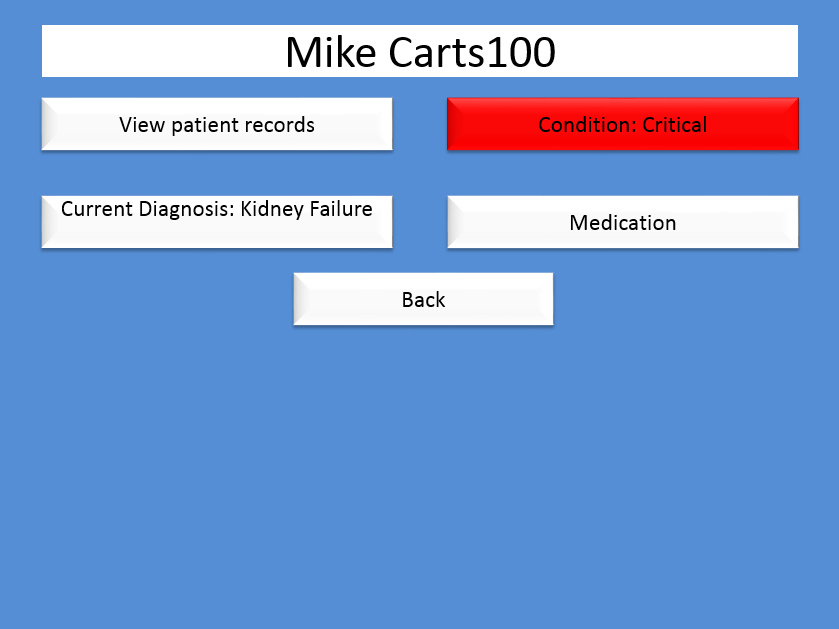


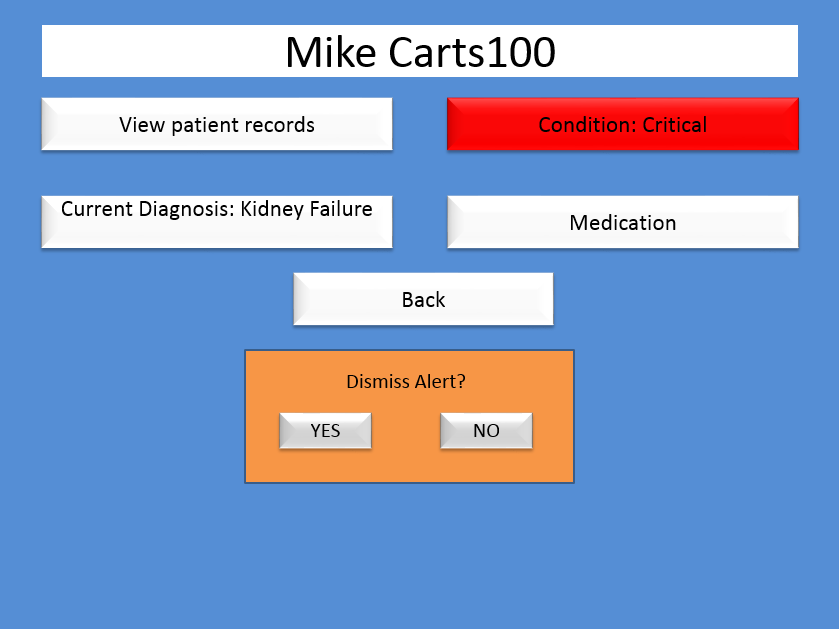


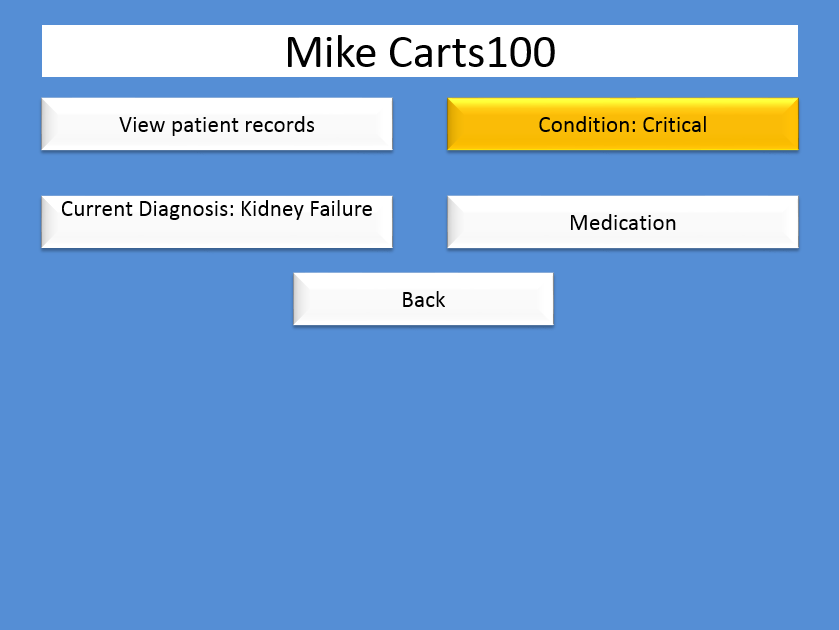


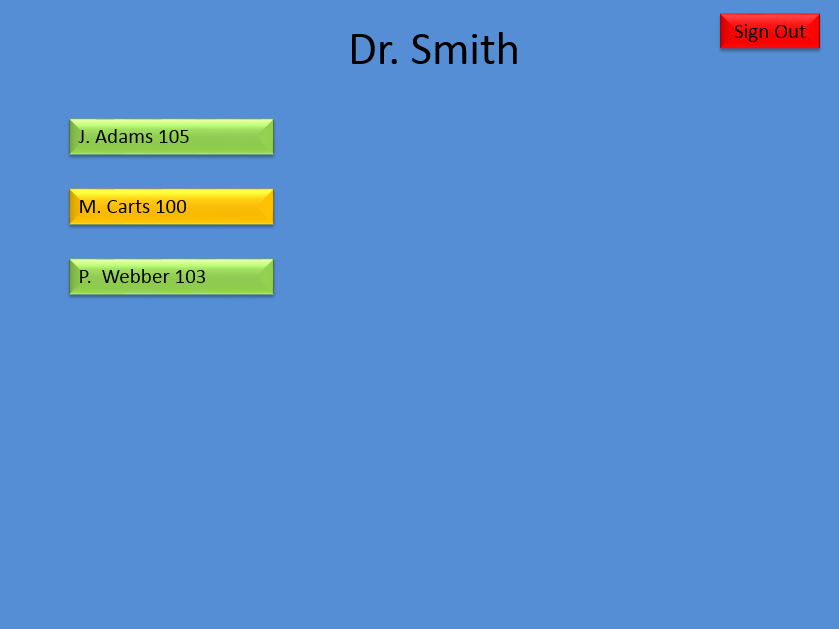


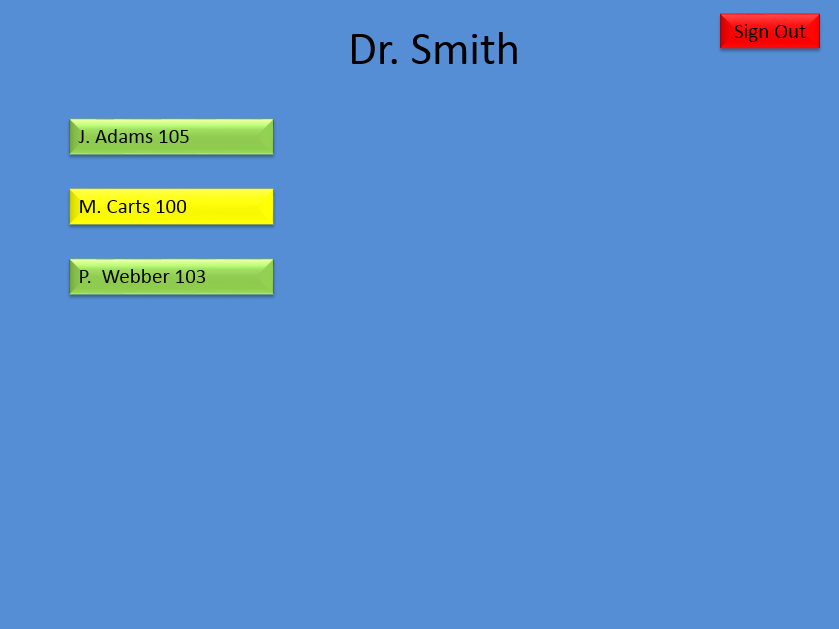




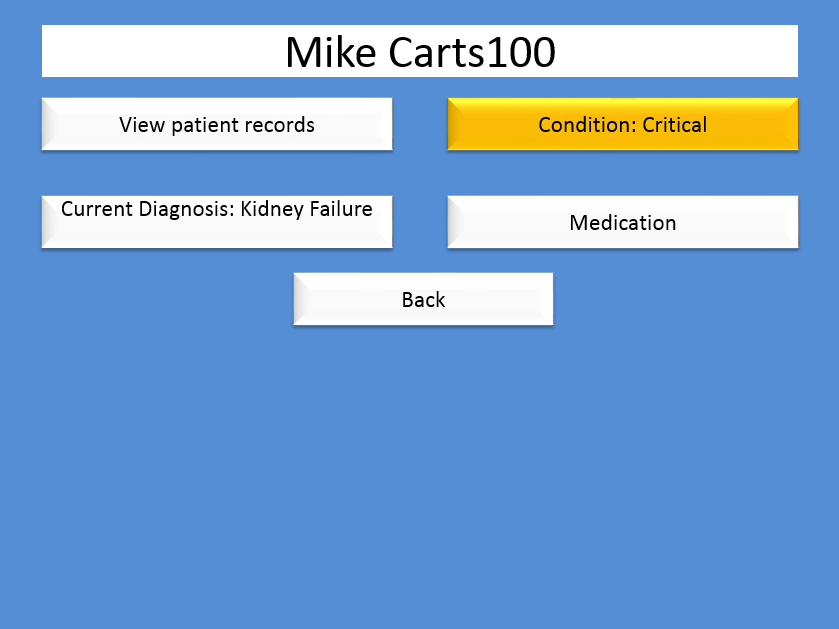




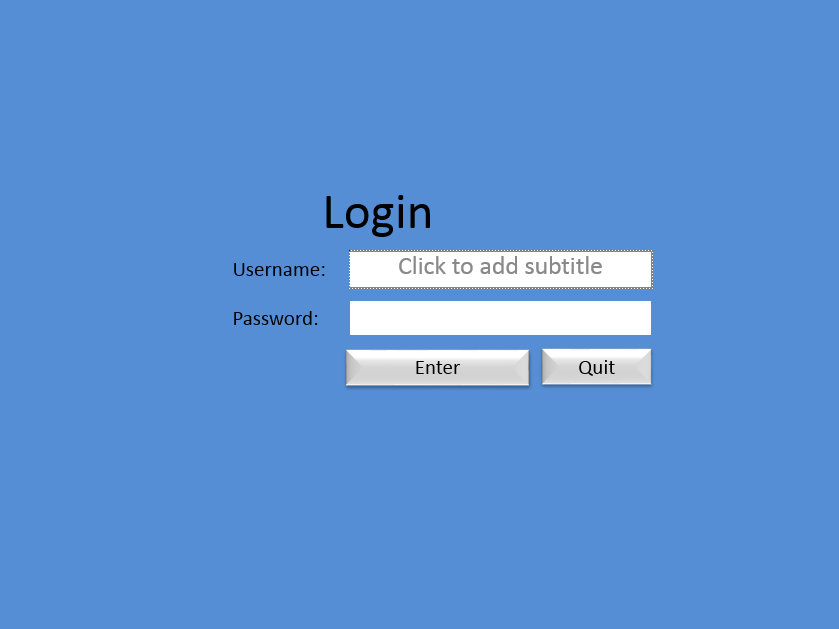


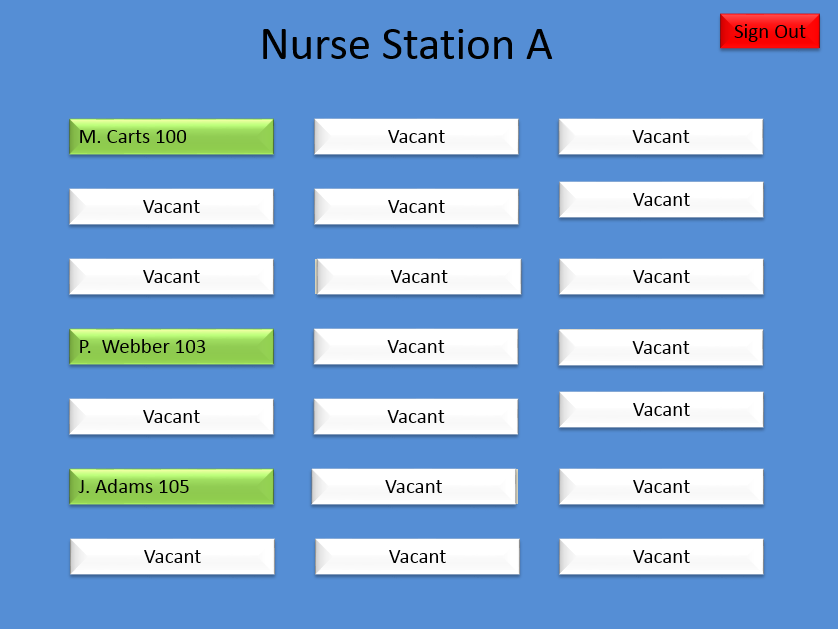


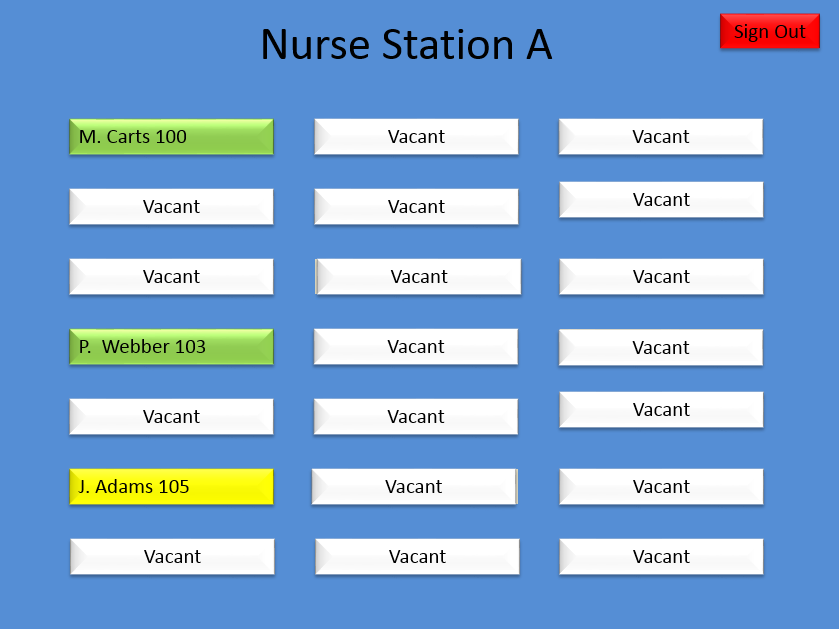


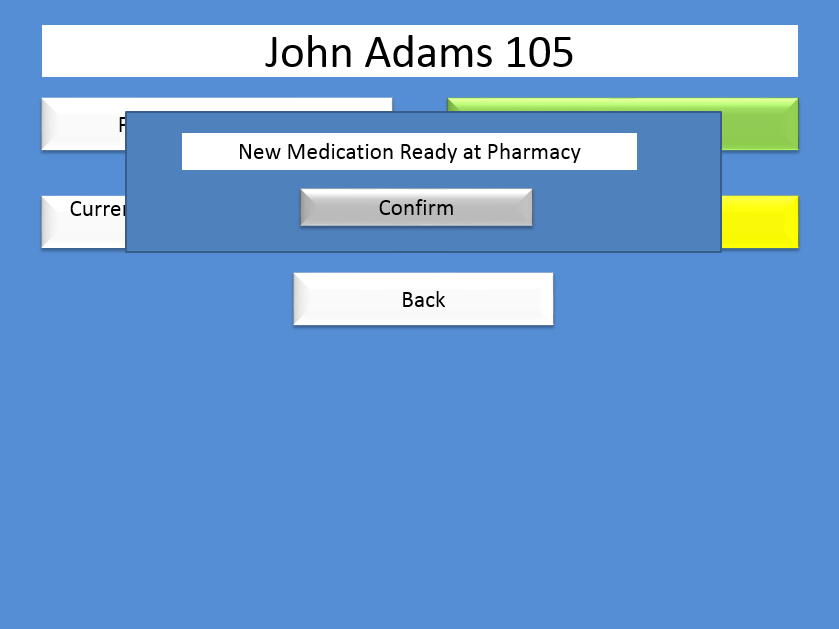


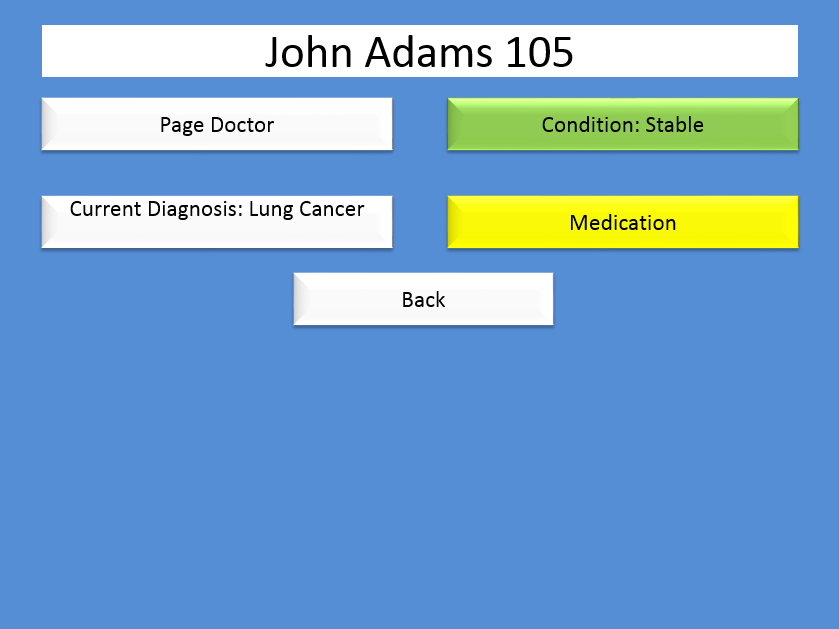


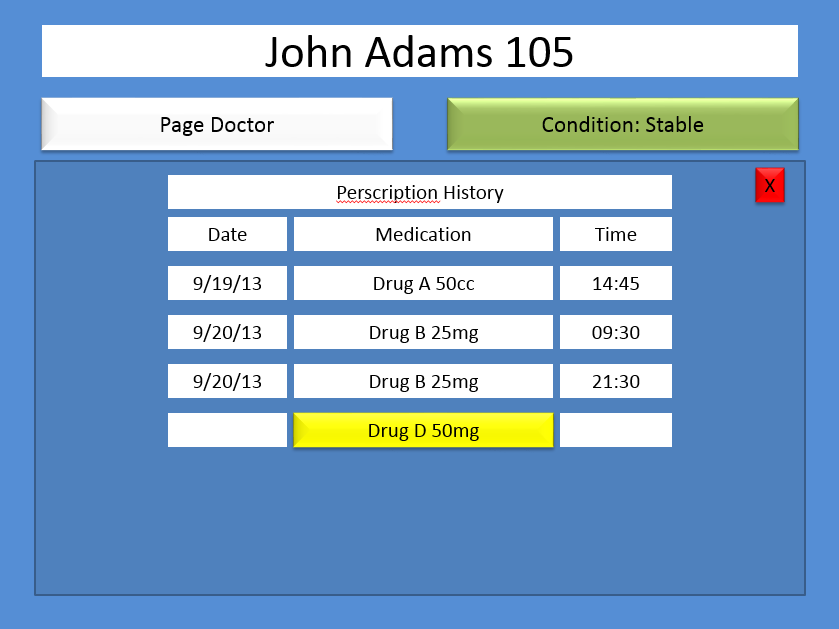


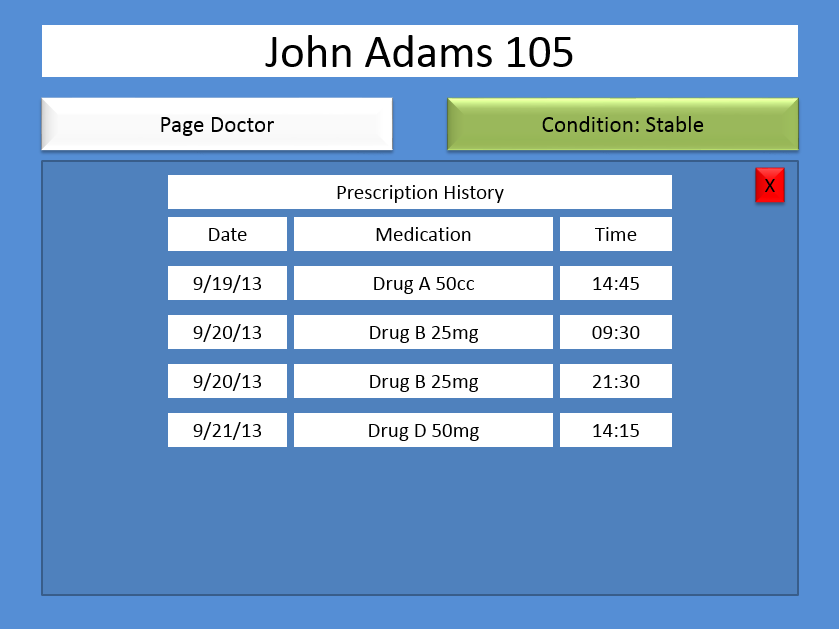


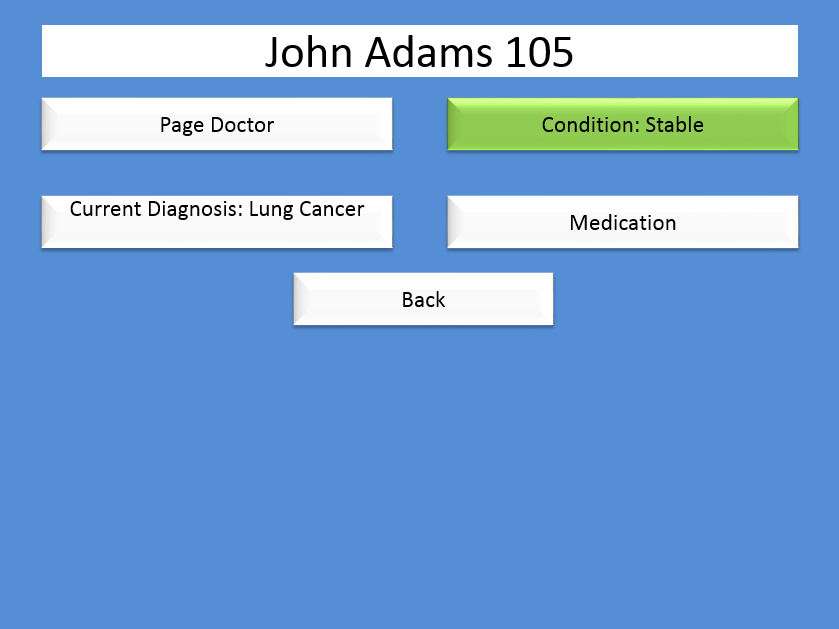


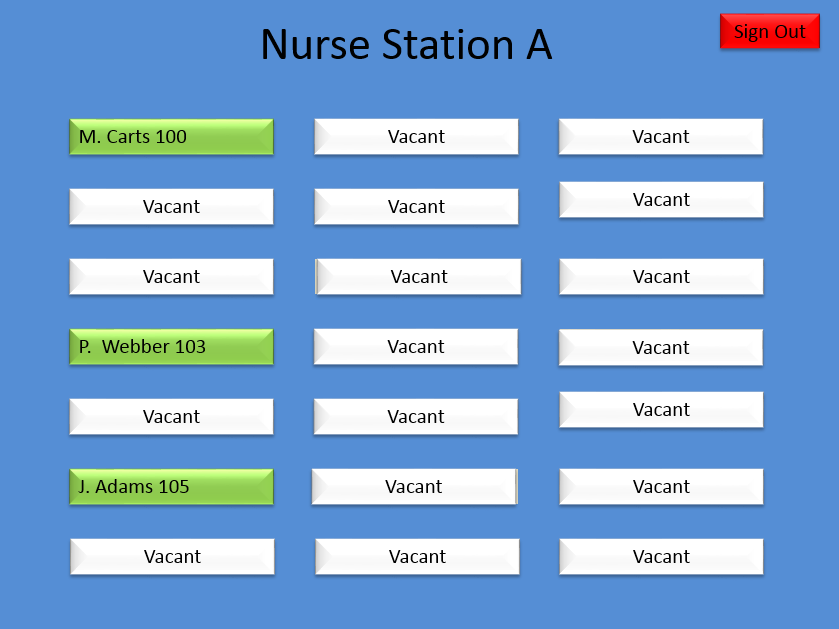


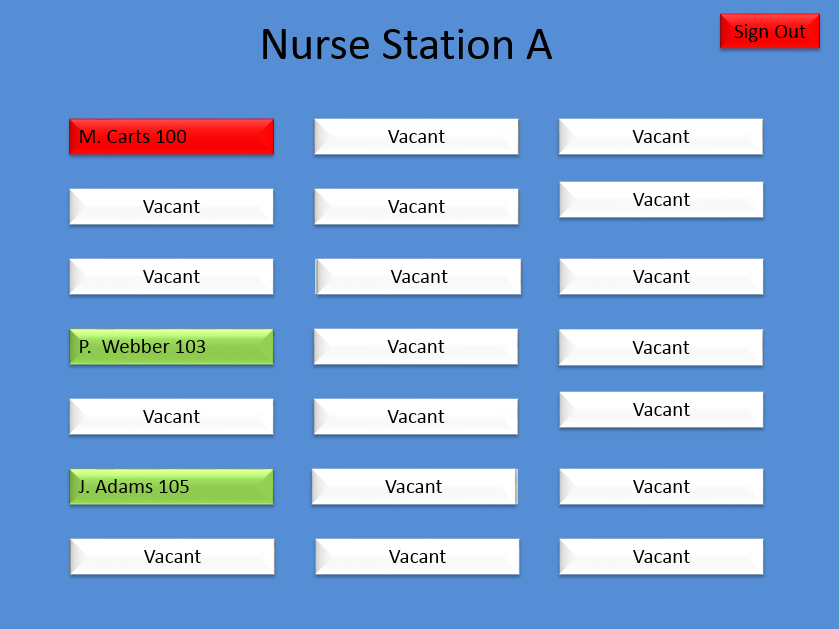


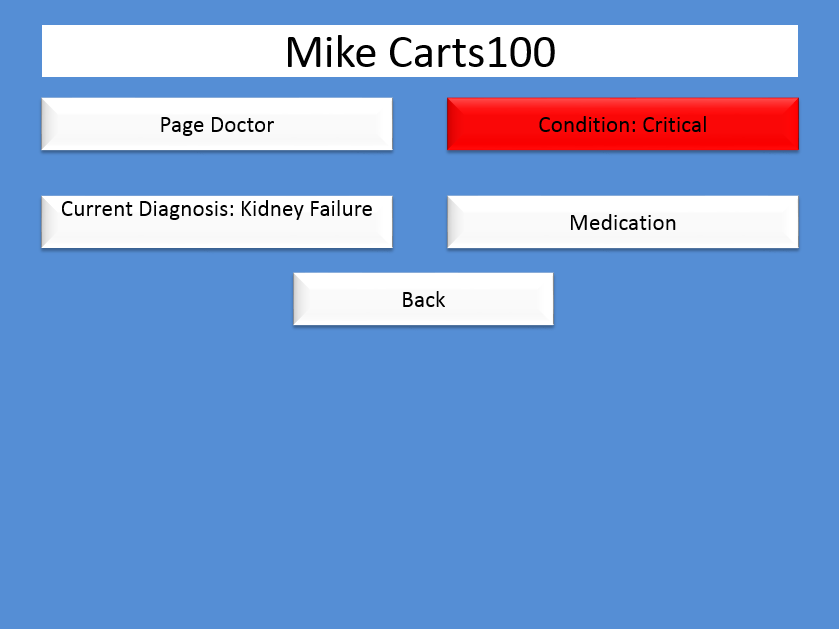


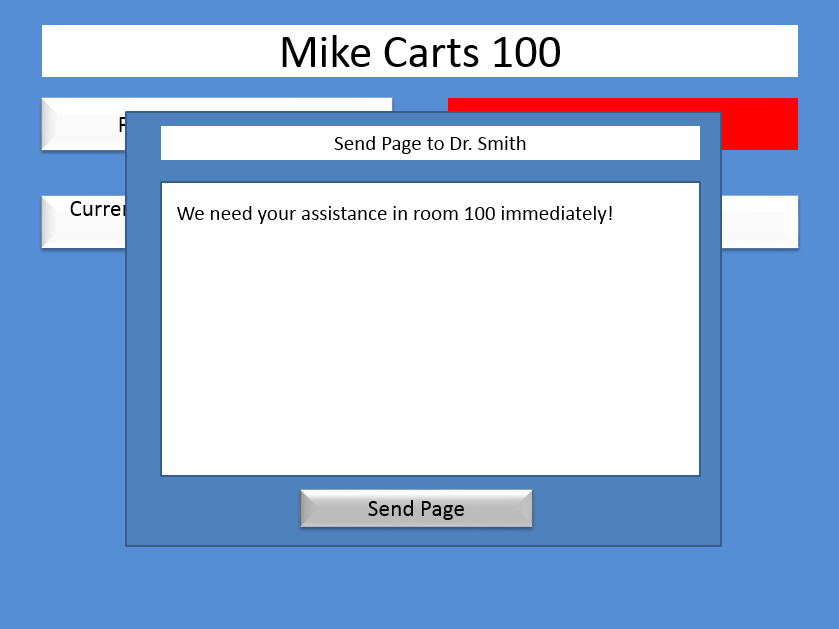


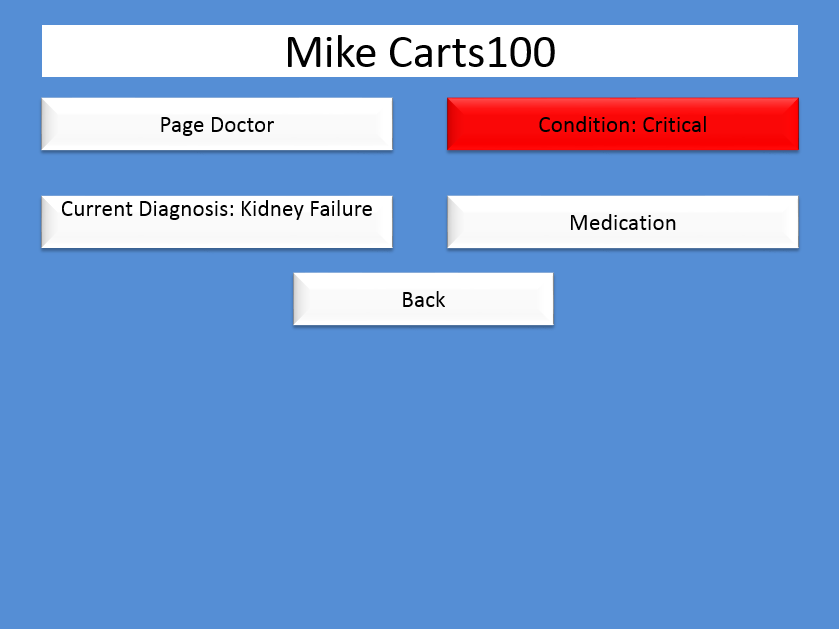


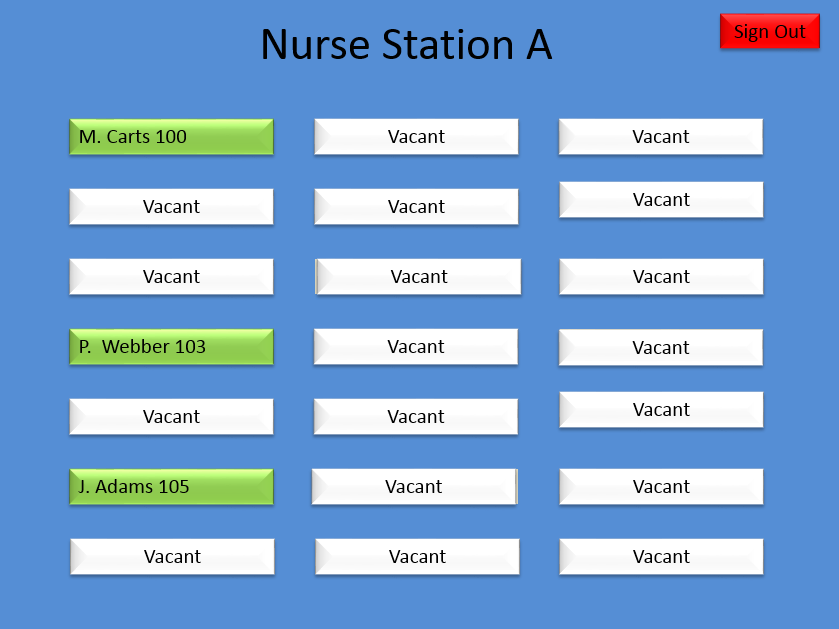


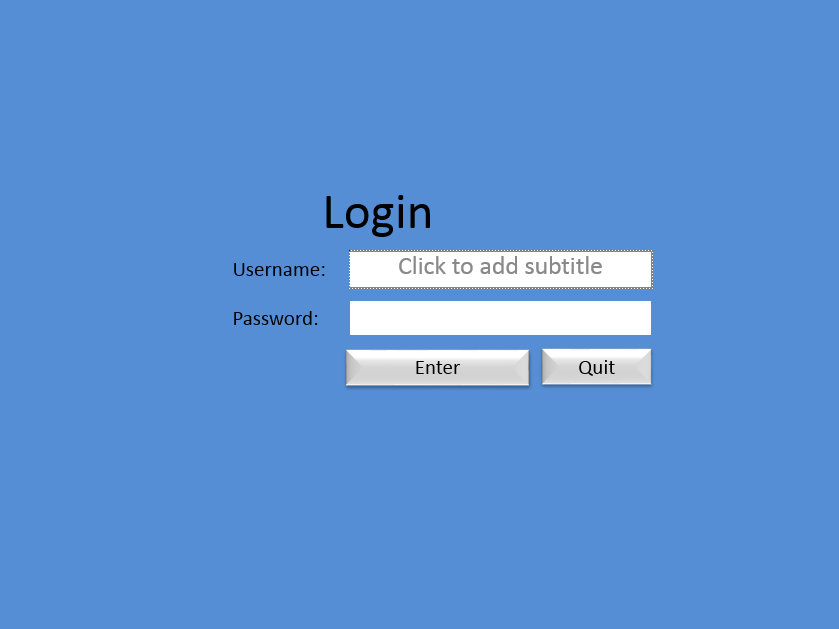




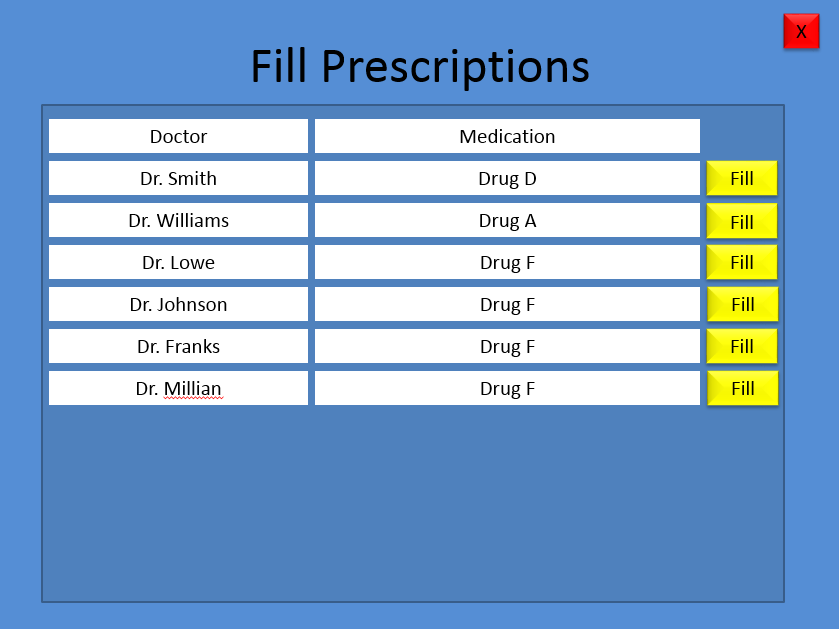


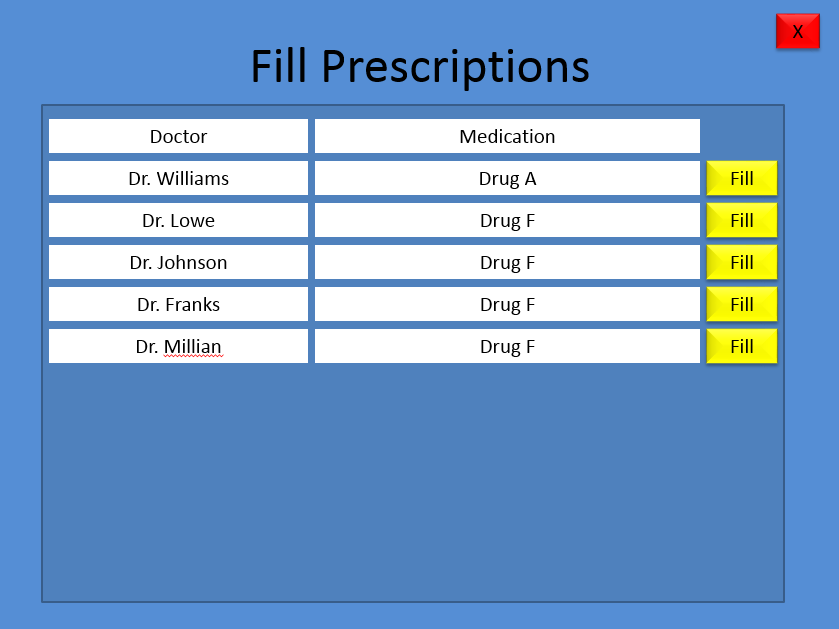




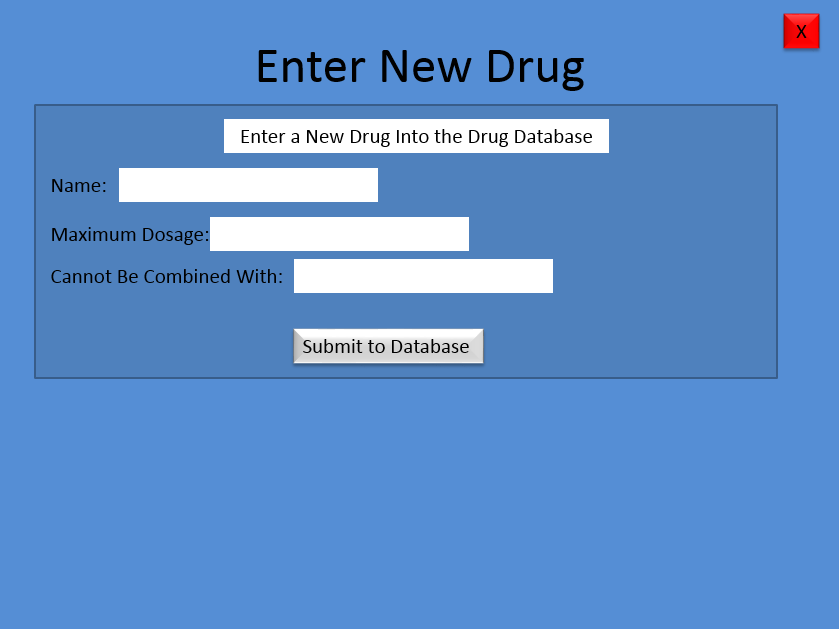


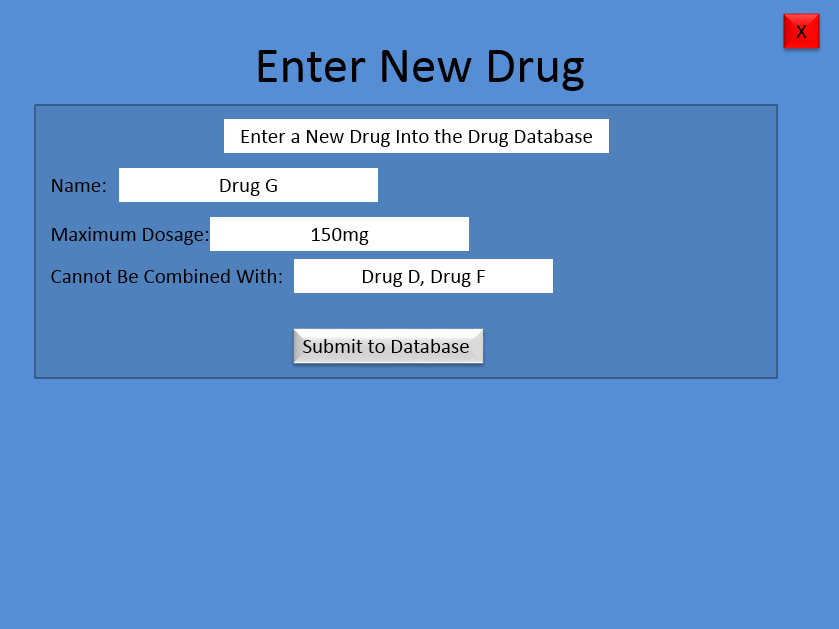




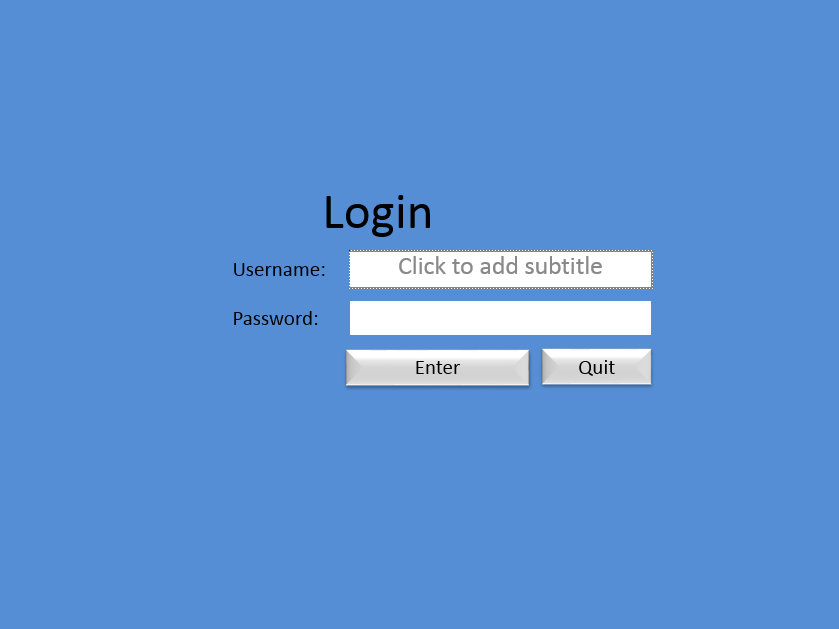




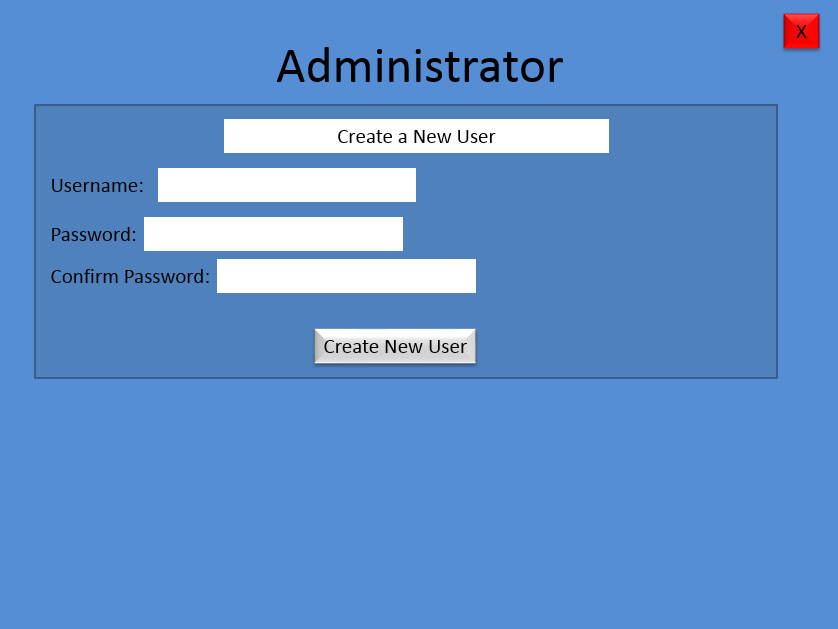


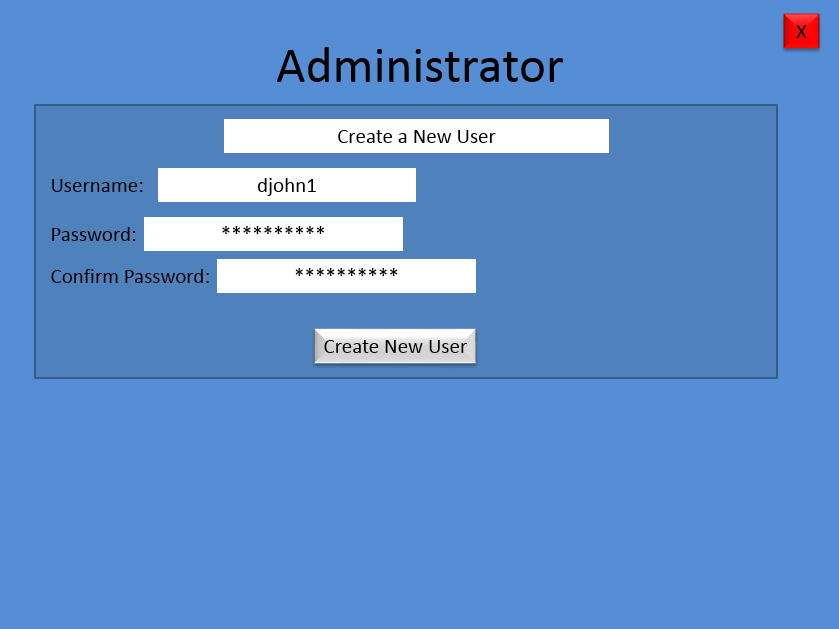




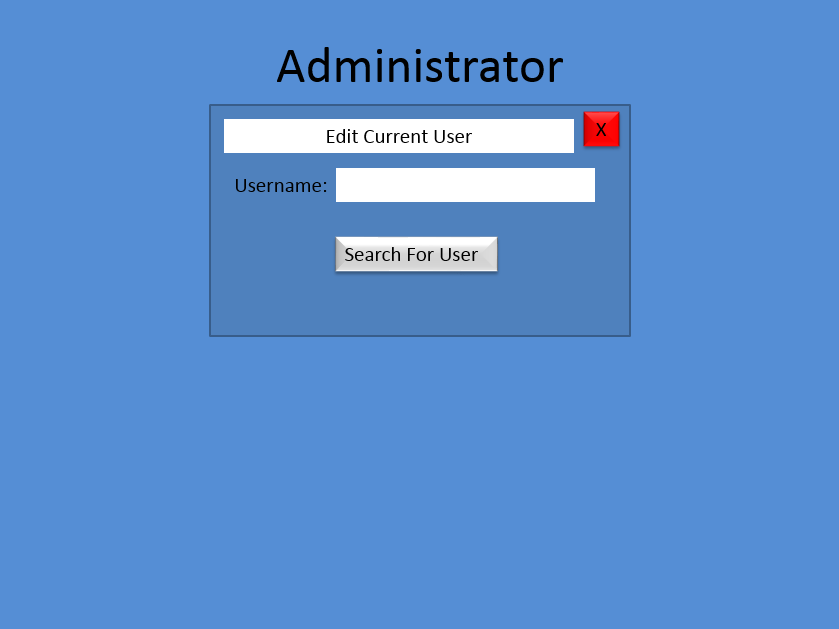


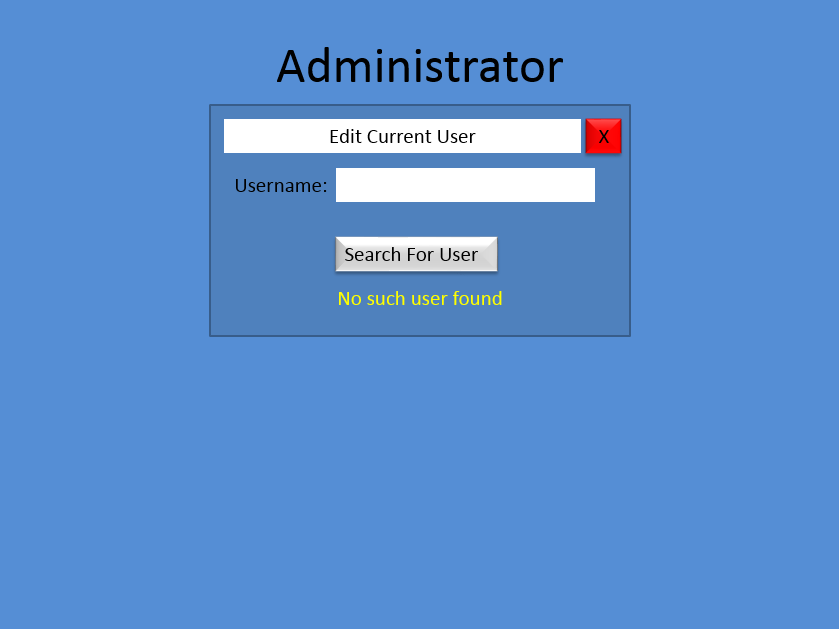


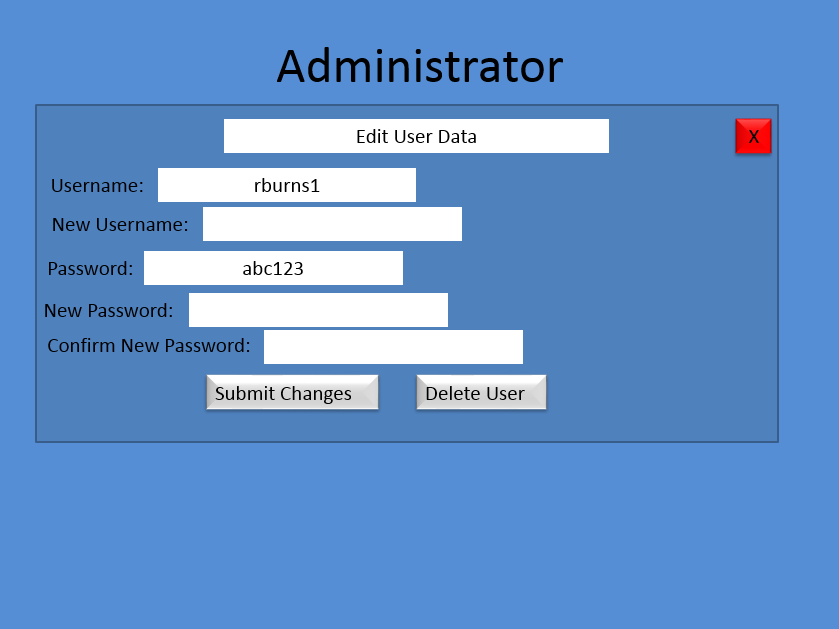


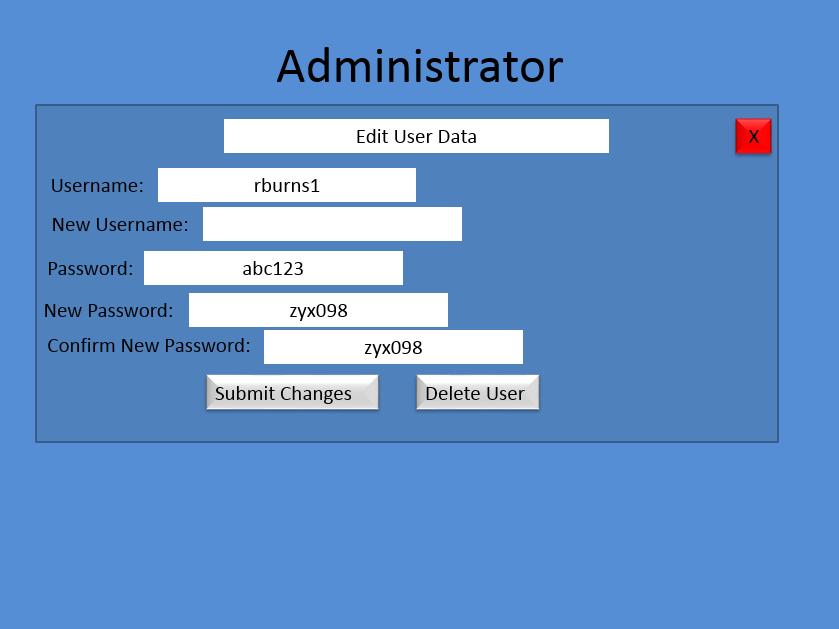




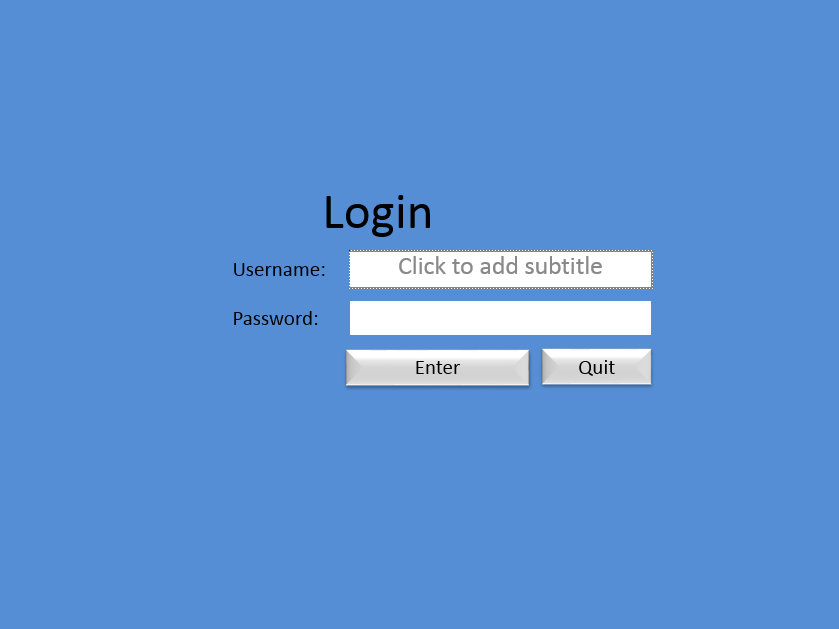












9. Project Legacy

The Good

* The project has almost all of the functionality we originally wanted in both the text based version (which was originally to be the back end of the software that performed the operations on the database) and the GUI based version (where the code that displays and updates the GUI also performs all the database operations, eliminating the need for the back end portion of the software).
* The GUI looks exactly as shown in the horizontal prototype.
* The Bad
* Communication problems plagued the development phase to the point where we essentially had two groups, each designing a different piece of software to do the same thing.
* Resulting from the above, we have a text based version of the software with all of our intended functionality, and a version that does have a GUI but also has the database operations code built in as part of the GUI itself.

What We Could Have Done Better

* Communication. Specifically, spending less time divvying up assignments for the next document, and spending more time making sure that everybody’s mental image of how the software will be put together is identical.

For Future Releases

* Staff will have their schedules encased in each instance of the User object and they will be viewable by other Users.
* Software will be capable of reading information from hospital equipment to allow nurses and doctors to view a patient’s pulse, breathing rate, and other vital medical information from their terminal.

10. Work Schedule Diagram

Always Be on Time

11. Gantt Chart

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Completed By: | Week 1 (9/01-9/07) | | | | | | |
| **Document #1** |  | S | M | T | W | T | F | S |
| Brainstorming | All |  |  |  |  |  |  |  |
| Project Topic Selection | All |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 2 (9/08-9/14) | | | | | | | Week 3 (9/15-9/21) | | | | | | | | | |
| **Document #2** |  | S | M | T | W | T | F | S | S | M | T | W | | T | | F | | S |
| Program Requirements for Problem Statements | All |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| Problem Statement | Dustin Green |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| Update WSD | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| RTM | Dustin Green, Aadil Allidina |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| Dictionary | John Rucker |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| Gnatt Chart | Dustin Green, Ketra King, Aadil Allidina |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |
| Rationale Statement | Don Squirek |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 3 (9/15-9/21) | | | | | | | Week 4 (9/22-9/28) | | | | | | | | Week 4 (9/29-10/05) | |
| **Document #3** |  | S | M | T | W | T | F | S | S | M | T | W | T | | F | S | S | M |
| Update WSD | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Use Cases | Dustin Green, Aadil Allidina |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Interaction Diagrams | Ketra King |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Use Case Rationale | Don Squirek |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Cost Analysis | Aadil Allidina |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Update Dictionary | John Rucker |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Update RTM | Dustin Green |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Update Gantt Chart | Ketra King |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |
| Database | Don Squirek |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 4 (9/29-10/05) | | | | Week 5 (10/06-10/12) | | | | | | | Week 6 (10/13-10/19) | | |
| **Document #4** |  | W | T | F | S | S | M | T | W | T | F | S | S | M | T |
| Software Architecture | Don Squirek |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RTM | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Category Interation Diagram | All |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update WSD | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update Gantt Chart | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dictionary | John Rucker |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Objects Rationale | Don Squirek |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Object Design | Dustin Green |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 8 (10/20-10/27) | | | | | | | Week 9 (10/28-11/02) | | | | | | |
| **Document #5** |  | S | M | T | W | T | F | S | S | M | T | W | T | F | S |
| RTM | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update WSD | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update Gantt Chart | Aadill Allidina |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update Dictionary | John Rucker |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Rationale | Don Squirek |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Week 9 (10/28-11/02) | | | | | | | Week 10 (11/03-11/13) | | | | | | |
| **Document #6** |  | S | M | T | W | T | F | S | S | M | T | W | T | F | S |
| Update RTM | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Source Code | All |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Test Cases | All |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update WSD | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Update Gantt Chart | Ketra King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dictionary | John Rucker |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Rationale for test cases | All |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

12. Dictionary

1. **Administrators** – Computer specialist who is responsible for maintaining the system, setting up user accounts, resetting passwords, profile permission access, and database access.
2. **Alarm** – An alert or warning.
3. **Condition -** current health situation of a patient – Stable or Critical.
4. **Diagnosis** –identification status of ailment from its symptoms (Diagnosis\_View).
5. **Database** – medical information system where patients information is stored.
6. **Dialog box** – a prompt from the system for confirmation for a user action.
7. **Drug** - a chemical substance used in the treatment, cure, prevention, or diagnosis of disease or used to otherwise enhance physical or mental well-being; also, any habit forming medicinal or illicit substance, IE narcotics, recreational drugs, caffeine, etc.
8. **Login screen** – User interface where the user is prompted to enter a username and password. Also referred to the Login\_GUI.
9. **Medications** - a medicinal substance; medicament.
10. **Password** - a password is a set of secret characters or words utilized to gain access to a computer, web page, network resource, or data. Passwords help ensure that computers or data can only be accessed by those who have been granted the right to view or access them.
11. **Patient\_Display -** interface designed to display patient information (Patient\_Record).
12. **Patient records** –patient information/database (Patient\_Records\_View).
13. **Permission level** – a User’s information access level.
14. **Pop-up alert** – a notification to the user from the database or another user in the system.
15. **Prescription** - A written order, by a physician, for the preparation and administration of a medicine or other treatment.
16. **Profile** – A user account setup by an administrator where users are able to view and manipulate patient information in the medical database.
17. **Sign Out button** – Button used to sign out of system.
18. **System** - User interface that gives the user access to patient information in real-time.
19. **Terminal** - A device that enables the user to communicate with a computer. Generally, a terminal is a combination of keyboard and display screen.
20. **User** – Doctors, Nurses, Administrators, and Pharmacists; alternatively referred to as an end-user, a user is any individual who is not involved with supporting or developing a computer or service. The user is any individual who has access granted by an administrator to access the medical system. For example in this case the doctors and nurses are users.
21. **Username** - referred to as an account name, login ID, and user ID, a username**.** Usernames allow multiple users to use the same computer and also commonly allow users to have access to their own personal profile and patient view.

13. Team Member Resumes

**Dustin Green**

**1015 Pebble Creek Trail ~ Suwanee, GA 30024**

**(770) 608-3731 dgreen7230@yahoo.com**

**Education**

South Forsyth High School, Graduated 2009

Georgia State University, Undergraduate 2009-present, Computer Science Major and History Minor

CSc Courses Taken (3000 level or higher)

Computer Organization and Programming (CSc 3210)

System Level Programming (CSc 3320)

Data Structures (CSc 3410)

Embedded Systems (CSc 4110)

Computer Architecture (CSc 4210)

Computer Languages

Java, C, some Assembly experience

**Extracurricular**

Boy Scouts of America 2002-2009, Eagle Scout rank

Boy Scouts of America 2009-2011, Assistant Scoutmaster

**Employment**

Starbucks Barista for Kroger, 2008-2009

Starbucks Barista for Target, 2009-present

Ketra King

225 Birch St. (404) 671-6891

Hapeville, GA 30354 ketra\_king@yahoo.com

**OBJECTIVE:** To obtain an intern position.

**EDUCATION**

**Georgia State University**, Atlanta, GA

Bachelors of Art and Sciences in Computer Science May 2014

GPA 3.12

**WORK EXPERIENCE**

**Kroger Deli**, Atlanta, GA June 2010 - Present

Bakery Assistant

* Provide Superior Customer Service
* Accurately prepare approximately 5 food orders per day
* Proficient use of Deli equipment such as deli meat cutter, bakery bread proffer
* Stocked service areas with supplies
* Completed training for food safety and deli guidelines

**Atlanta Spirit, LLC/ Philips Arena**, Atlanta, GA June 2008 – June 2009

Assistant to Assistant Director of Security

* Basic Office Clerical Duties such as filled papers, made copies, made and answered phone calls
* Participated with Special Event preparation
* Attended and took minutes during event operations meetings
* Handled security training preparations by building portfolios, creating build boards and training packets

**Notable Projects:**

<https://www.dropbox.com/sh/2g1e8d4pdfd60u4/sG6NBBVhGR>

* Includes school projects written in various programming languages such as, Python, Java, C, and Android.

**Technologies:**

* Proficient: Microsoft Office Excel, PowerPoint, Word, Java
* Familiar with: Python, C, Linux, GitHub, Android Programming, HTML, PHP, MySQL, Javascript

14. User Guide

15. Source Code

**a. Text Based Version**

package drug;

import static drug.DrugMain.drugList;

import java.util.\*;

import java.io.\*;

import java.sql.Timestamp;

public class drug {

String uses;

String dName;

public LinkedList<String> incompatible = new LinkedList<String>();

double quantity;

double maxDose;

double durration;

public static int addNew(String command){

drug a = new drug();

System.out.println("Enter A Name For New Drug: ");

Scanner in = new Scanner(System.in);

a.dName = in.next();

System.out.println("Enter The Use For New Drug: ");

a.uses = in.next();

System.out.println("Are Any Medications Incompatible With New Drug?(y/n");

int c = 0;

command = in.next();

if(command.equals("y")){

while (c<100){

System.out.println("Enter An Incompatible Medication");

drug current = new drug();

current.dName = in.next();

a.incompatible.add(current.dName);

System.out.println("Any Other Incompatible Medications?(y/n)");

command = in.next();

if(command.equals("n")){

c=500;

}

}

}

System.out.println("Enter The Max Doseage For New Drug(in ml): ");

a.maxDose = in.nextDouble();

System.out.println("Enter The Durration For New Drug: ");

a.durration = in.nextDouble();

System.out.println("Enter The Quantity Of New Durg On Hand: ");

a.quantity = in.nextDouble();

drugList.add(a);

System.out.println(drugList.size());

return 0;

}

public static int editDrug(){

Scanner in = new Scanner(System.in);

System.out.println("Enter The Drug You Would Like To Edit: ");

drug select = new drug();

select.dName = in.next();

for(int i=0;i<drugList.size();i++){

drug current = drugList.get(i);

System.out.println(current.dName);

if(current.dName.equals(select.dName)){

System.out.println("Match found");

System.out.println("Enter a Feild You Would Like To Edit");

System.out.println("1 For Name");

System.out.println("2 For Incompatible");

System.out.println("3 For Max Dose");

System.out.println("4 For Durration");

System.out.println("5 For Quantity");

System.out.println("E for Escape");

String command = in.next();

if(command.equals("1")){

System.out.println("Enter The Drug's New Name: ");

current.dName = in.next();

}

if(command.equals("2")){

System.out.println("Enter Revised List of Incompatible Drugs: ");

int c = 0;

command = in.next();

while (c<100){

System.out.println("Enter An Incompatible Medication");

drug edit = new drug();

edit.dName = in.next();

current.incompatible.add(current.dName);

System.out.println("Any Other Incompatible Medications?(y/n)");

command = in.next();

if(command.equals("n")){

c=500;

}

}

}

if(command.equals("3")){

System.out.println("Enter The Max Dose: ");

current.maxDose = in.nextDouble();

}

if(command.equals("4")){

System.out.println("Enter The Durration ");

current.durration = in.nextDouble();

}

if(command.equals("5")){

System.out.println("Enter The Quantity: ");

current.quantity = in.nextDouble();

}

if(command.equals("E")){

return 0;

}

return 0;

}

}

System.out.println("No Match Found Please Try Again");

editDrug();

return 0;

}

public static int showList(){

for(int i=0;i<drugList.size();i++){

drug current = drugList.get(i);

System.out.println(current.dName);

}

return 0;

}

public static int removeDrug(){

Scanner in = new Scanner(System.in);

String confirm;

System.out.println("Enter The Drug You Would Like To Delete: ");

drug select = new drug();

select.dName = in.next();

for(int i=0;i<drugList.size();i++){

drug current = drugList.get(i);

System.out.println(current.dName);

if(current.dName.equals(select.dName)){

System.out.println("Match found");

System.out.println("Are You Sure You Want To Delete "+current.dName+"?(y/n)");

confirm = in.next();

if(confirm.equals("y")){

drugList.remove(i);

return 0;

}

}

}

System.out.println("Drug Not Found");

return 0;

}

public static int showStock(){

for(int i=0;i<drugList.size();i++){

drug current = drugList.get(i);

System.out.println("Currently "+current.quantity+" of "+current.dName+" on hand");

}

return 0;

}

}

package drug;

import java.util.\*;

import java.io.\*;

import java.sql.Timestamp;

import java.io.File;

import java.io.FileNotFoundException;

public class loader {

public static int loadDrugDB(){

try {

File file = new File("C:\\Documents and Settings\\Administrator\\Desktop\\drugdb.txt");

Scanner scanner = new Scanner(file);

while (scanner.hasNext()) {

if(scanner.next().equals("[")){

drug a = new drug();

String incoming = scanner.next();

while(!incoming.equals("]")){

a.dName = incoming;

if(scanner.next().equals("\*")){

String marker = scanner.next();

while(!marker.equals("$")){

a.incompatible.add(marker);

marker = scanner.next();

}

}

a.uses = scanner.next();

String quantity = scanner.next();

double quant = Double.parseDouble(quantity);

a.quantity = quant;

String maxDose = scanner.next();

double max = Double.parseDouble(maxDose);

a.maxDose = max;

String durration = scanner.next();

double durr = Double.parseDouble(durration);

a.durration = durr;

incoming = scanner.next();

}

DrugMain.drugList.add(a);

//System.out.println("---------");

}

}

scanner.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return 0;

}

public static int loadPatientDB(){

try {

File file = new File("C:\\Documents and Settings\\Administrator\\Desktop\\patientdb.txt");

Scanner scanner = new Scanner(file);

while (scanner.hasNext()) {

if(scanner.next().equals("[")){

patient a = new patient();

String incoming = scanner.next();

while(!incoming.equals("]")){

a.fName = incoming;

a.lName = scanner.next();

a.tel = scanner.next();

a.address = scanner.next();

a.insurance = scanner.next();

String social = scanner.next();

int str = Integer.parseInt(social);

a.SSN = str;

a.doctor = scanner.next();

String room = scanner.next();

str = Integer.parseInt(room);

a.roomNumber = str;

a.currentDiagnosis = scanner.next();

String iD = scanner.next();

str = Integer.parseInt(iD);

a.patientId = str;

incoming = scanner.next();

}

DrugMain.patientList.add(a);

//System.out.println("---------");

}

}

scanner.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return 0;

}

public static int loadUserDB(){

try {

File file = new File("C:\\Documents and Settings\\Administrator\\Desktop\\userdb.txt");

Scanner scanner = new Scanner(file);

while (scanner.hasNext()) {

if(scanner.next().equals("[")){

user a = new user();

String incoming = scanner.next();

while(!incoming.equals("]")){

a.username = incoming;

a.password = scanner.next();

a.lastName = scanner.next();

a.permission = scanner.next();

a.page = scanner.next();

incoming = scanner.next();

}

DrugMain.userList.add(a);

//System.out.println("---------");

}

}

scanner.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return 0;

}

}

package drug;

import java.util.\*;

import static drug.DrugMain.drugList;

import static drug.DrugMain.filledList;

import static drug.DrugMain.pharmList;

import static drug.DrugMain.patientList;

//Patient Object

//Each Patient object represents one patient in the hospital. Each object contains all of the patient's

//information, such as their name, phone number, address, social security number, room number, the

//doctor assigned to them, the nurse station corresponding to their room number, previous medical

//diagnoses, previous and current medications, the times of their next scheduled checkup and

//medication, and so on.

public class patient {

public int patientId;

public String fName;

public String lName;

public String tel;

public String address;

public String insurance;

public int SSN;

public String doctor;

private int nurseStation;

public int roomNumber;

public String currentDiagnosis;

public LinkedList<drug> medications = new LinkedList<drug>(); //stores drug objects of each drug that the //patient is currently taking

public LinkedList<String> diagnosisHistory = new LinkedList<String>(); //stores each each diagnosis from most //recent to least

private double nextCheckup; //time of next checkup

private double nextMedication; //time next medication will be administered

private double lastMedication; //time last medication was administered

/\*public Patient(String fN, String lN, int pN, String adrs, String iP, int social, String doc, int nS, int rN, String cD) {

firstName = fN;

lastName = lN;

phoneNumber = pN;

address = adrs;

insuranceProvider = iP;

SSN = social;

doctor = doc;

nurseStation = nS;

roomNumber = rN;

currentDiagnosis = cD;

}\*/

public static int addPatient(){

patient a = new patient();

System.out.println("Enter Patient's First Name: ");

Scanner in = new Scanner(System.in);

a.fName = in.next();

System.out.println("Enter Patient's Last Name: ");

a.lName = in.next();

System.out.println("Enter Patient's Telephone Number: ");

a.tel = in.next();

System.out.println("Enter Patient's Address: ");

a.address = in.next();

System.out.println("Enter Patient's Insureance Provider: ");

a.insurance = in.next();

System.out.println("Enter Patient's SSN: ");

a.SSN = in.nextInt();

System.out.println("Enter Patient's Doctor: ");

a.doctor = in.next();

System.out.println("Enter Patient's Room Number: ");

a.roomNumber = in.nextInt();

//adds current medication the patient is on

System.out.println("Is Patient On Current Medications?(y/n");

int c = 0;

String command = in.next();

if(command.equals("y")){

while (c<100){

System.out.println("Enter A Medication");

drug current = new drug();

current.dName = in.next();

a.medications.add(current);

System.out.println("Any Other Medications?(y/n)");

command = in.next();

if(command.equals("n")){

c=500;

}

}

}

a.patientId = DrugMain.patientList.size()+1;

System.out.println("Patient ID: "+a.patientId);

patientList.add(a);

return 0;

}

public static int showPatientInfo(){

showPatientIds();

//patient find = new patient();

System.out.println("Enter The Patient ID For Which You Wish To View: ");

Scanner in = new Scanner(System.in);

//find.patientId = in.nextInt();

int find = in.nextInt();

for(int i=0; i<patientList.size();i++){

patient check = patientList.get(i);

if(find == check.patientId){

System.out.println("Patient Found");

System.out.println("Patient ID: "+check.patientId);

System.out.println("Name: "+check.fName+" "+ check.lName);

System.out.println("Doctor: "+check.doctor);

System.out.println("Room Number: "+check.roomNumber);

System.out.println("Current Diagnosis: "+check.currentDiagnosis);

System.out.println("Current Meds: ");

int c = 0;

while(c<check.medications.size()){

drug med = check.medications.get(c);

System.out.println(med.dName);

c++;

}

return 0;

}

}

System.out.println("Patient Not Found");

return 0;

}

public static int editPatientInfo(){

showPatientIds();

System.out.println("Enter The Patient ID For Which You Wish To View: ");

Scanner in = new Scanner(System.in);

int find = in.nextInt();

for(int i = 0; i<patientList.size(); i++){

patient check = patientList.get(i);

if(check.patientId == find){

System.out.println("patient Found");

System.out.println("Enter a property to edit: ");

System.out.println("1 For First Name");

System.out.println("2 For Last Name");

System.out.println("3 For Phone Number");

System.out.println("4 For Address");

System.out.println("5 For Insurance Provider");

System.out.println("6 For SSN");

System.out.println("7 For Doctor");

System.out.println("8 For Room Number");

System.out.println("9 For current Diagnosis");

System.out.println("10 For Current Medications");

System.out.println("11 For diagnosis History");

String command = in.next();

if(command.equals("1")){

System.out.println("Enter New Patient First Name: ");

String update = in.next();

check.fName = update;

return 0;

}

if(command.equals("2")){

System.out.println("Enter New Patient Last Name: ");

String update = in.next();

check.lName = update;

return 0;

}

if(command.equals("3")){

System.out.println("Enter New Patient Phone Number: ");

String update = in.next();

check.tel = update;

return 0;

}

if(command.equals("4")){

System.out.println("Enter New Patient Address: ");

String update = in.next();

check.address = update;

return 0;

}

if(command.equals("5")){

System.out.println("Enter New Patient Insureance Provider: ");

String update = in.next();

check.insurance = update;

return 0;

}

if(command.equals("6")){

System.out.println("Enter New Patient SSN: ");

int update = in.nextInt();

check.SSN = update;

return 0;

}

if(command.equals("7")){

System.out.println("Enter New Patient Doctor: ");

String update = in.next();

check.doctor = update;

return 0;

}

if(command.equals("8")){

System.out.println("Enter New Patient Room Number: ");

int update = in.nextInt();

check.roomNumber = update;

return 0;

}

if(command.equals("9")){

System.out.println("Enter New Patient Diagnosis: ");

String update = in.next();

check.currentDiagnosis = update;

check.diagnosisHistory.add(update);

return 0;

}

if(command.equals("10")){

System.out.println("Are You Sure You Wish To Re-Write All Current Medications? (y/n)");

command = in.next();

if(command.equals("n")){

return 0;

}

check.medications.clear();

System.out.println("Enter All Current Medications: ");

while(command.equals("y")){

drug update = new drug();

update.dName = in.next();

int c = 0;

while(c<drugList.size()){

drug lookup = new drug();

lookup = drugList.get(c);

if(update.dName.equals(lookup.dName)){

System.out.println("Match Found");

check.medications.add(lookup);

System.out.println("Enter Another Medication? (y/n)");

command = in.next();

System.out.println("Enter Next Medication: ");

}

c++;

}

}

return 0;

}

if(command.equals("11")){

System.out.println("Are Your Sure You Wish To RE-WRITE This Patient's Diagnosis History? (y/n): ");

command = in.next();

if(command.equals("n")){

return 0;

}

check.diagnosisHistory.clear();

while(command.equals("y")){

System.out.println("Enter a Diagnosis: ");

String update = in.next();

check.diagnosisHistory.add(update);

System.out.println("Add Another Diagnosis?(y/n)");

command = in.next();

}

return 0;

}

}

}

return 0;

}

public static int showDiagnosisHistory(){

showPatientIds();

Scanner in = new Scanner(System.in);

System.out.println("Enter The Patient ID For Which You Wish To View: ");

int find = in.nextInt();

int c = 0;

while(c < patientList.size()){

patient check = patientList.get(c);

if(find == check.patientId){

System.out.println("Diagnosis History for "+check.fName+" "+check.lName+": ");

int d = 0;

while(d < check.diagnosisHistory.size()){

String diag = check.diagnosisHistory.get(d);

System.out.println(diag);

d++;

}

}

c++;

}

return 0;

}

public static int addDiagnosis(){

showPatientIds();

Scanner in = new Scanner(System.in);

System.out.println("Enter The Patient ID For Which You Wish To View: ");

int find = in.nextInt();

int c = 0;

while(c < patientList.size()){

patient check = patientList.get(c);

if(find == check.patientId){

System.out.println("Patient Found");

System.out.println("Enter new Diagnosis for "+check.fName+" "+check.lName+": ");

String overwrite = in.next();

check.currentDiagnosis = overwrite;

check.diagnosisHistory.add(overwrite);

}

c++;

}

return 0;

}

public static int showPatientIds(){

for(int i=0;i<patientList.size();i++){

patient look = patientList.get(i);

System.out.println(look.patientId+" "+look.fName+" "+look.lName);

}

return 0;

}

public String getFirstName() {

return fName;

}

public String getLastName() {

return lName;

}

public String getPhoneNumber() {

return tel;

}

public String getAddress() {

return address;

}

public String getInsuranceProvider() {

return insurance;

}

public int getSSN() {

return SSN;

}

public String getDoctor() {

return doctor;

}

public int getNurseStation() {

return nurseStation;

}

public int getRoomNumber() {

return roomNumber;

}

public String getCurrentDiagnosis() {

return currentDiagnosis;

}

public LinkedList getMedications() {

return medications;

}

public double getNextCheckup() {

return nextCheckup;

}

public double getNextMedication() {

return nextMedication;

}

public void changeFirstName(String newFirstName) {

fName = newFirstName ;

}

public void changeLastName(String newLastName) {

lName = newLastName;

}

public void changePhoneNumber(String newPhoneNumber) {

tel = newPhoneNumber;

}

public void changeAddress(String newAddress) {

address = newAddress;

}

public void changeInsuranceProvider(String newInsuranceProvider) {

insurance = newInsuranceProvider;

}

public void changeSSN(int newSSN) {

SSN = newSSN;

}

public void changeDoctor(String newDoctor) {

doctor = newDoctor;

}

public void changeNurseStation(int newNurseStation) {

nurseStation = newNurseStation;

}

public void changeRoomNumber(int newRoomNumber) {

roomNumber = newRoomNumber;

}

public void setNextCheckup(int newCheckupTime) {

nextCheckup = newCheckupTime;

}

public void setNextMed(int newMedTime) {

lastMedication = nextMedication;

nextMedication = newMedTime;

}

public void newCurrentDiagnosis(String cD) {

currentDiagnosis = cD;

diagnosisHistory.addFirst(currentDiagnosis);

}

public void newMedication(drug d) {

medications.addFirst(d);

}

}

package drug;

import static drug.DrugMain.drugList;

import static drug.DrugMain.filledList;

import static drug.DrugMain.pharmList;

import static drug.DrugMain.patientList;

import static drug.DrugMain.administeredList;

import java.util.\*;

import java.io.\*;

import java.sql.Timestamp;

public class perscription {

static List<drug> perscribedDrugs = new ArrayList<drug>();

public LinkedList<drug> currentMeds = new LinkedList<drug>();

String perscribedDrug;//

String PatientDoctor;//

String patientName;

int Id;

Timestamp timePerscribed;//

Timestamp timeAdministered;

Timestamp timeFilled;

boolean isCompatible;//

double dosePerDay;//

double perscriptionDuration;//

public static int perscribeNew(){

perscription a = new perscription();

Scanner in = new Scanner(System.in);

drug perscribe = new drug();

patient lookup = new patient();

int z = 0;

System.out.println("Enter The Patient ID For Which You Wish To Make A Percription For: ");

a.Id = in.nextInt();

//looks up patient and gets patient doctor and other info

for(int q = 0; q < patientList.size();q++){

lookup = patientList.get(q);

if(a.Id == lookup.patientId){

System.out.println("patient found");

z = q;

a.PatientDoctor = lookup.getDoctor();

a.currentMeds = lookup.medications;

//get list of incompatible drugs for the patient

/\*int c = 0;

while(c<lookup.medications.size()){

System.out.println("size of med list is"+lookup.medications.size());

System.out.println(c);

drug bad = a.currentMeds.get(c);

System.out.println("current med list: "+bad.dName);

a.currentMeds.add(bad);

c++;

}\*/

}

}

//Gets the drug user wants to add

System.out.println("Enter The Drug You Would Like To Perscribe: ");

a.perscribedDrug = in.next();

//check the database to find the drug

for(int i=0;i<drugList.size();i++){

drug current = drugList.get(i);

if(a.perscribedDrug.equals(current.dName)){

System.out.println("Drug match Found");

//check and update drug stock

if(current.quantity < 1){

System.out.println("Drug OUT OF STOCK");

System.out.println("Perscription cancled!");

return 0;

}

current.quantity = current.quantity - 1;

//get doseage and check

System.out.println("Enter The Dose To Perscribe: (dose/day)");

a.dosePerDay = in.nextDouble();

if(current.maxDose < a.dosePerDay){

System.out.println("The dose perscribed is too high continue? (y/n)");

String confirm = in.next();

if(confirm.equals("n")){

current.quantity = current.quantity + 1;

return 0;

}

}

//check for incompatibility

int c = 0;

while(c < current.incompatible.size()){

System.out.println("Made it!");

String test = current.incompatible.get(c);

System.out.println("checking the incompatibility of "+test);

int d = 0;

while( d < a.currentMeds.size()){

drug incomp = a.currentMeds.get(d);

System.out.println("comparing with "+incomp.dName);

if(incomp.dName.equals(test)){

System.out.println("Incompatibility Found!");

System.out.println("Continue With Perscription?(y/n)");

String command = in.next();

if(command.equals("n")){

current.quantity = current.quantity + 1;

return 0;

}

}

d++;

}

c++;

}

//adds perscribed drug to the current meds list and patient medication

a.currentMeds.add(current);

patientList.get(z).medications = a.currentMeds;

//get the rest of the attributes for the perscription

System.out.println("Enter The Durration For This Perscription: ");

a.perscriptionDuration = in.nextDouble();

//get time stamp

java.util.Date date= new java.util.Date();

a.timePerscribed = new Timestamp(date.getTime());

//System.out.println(a.timePerscribed);

//perscription.perscribedDrugs.add(current);

pharmList.add(a);

return 0;

}

}

System.out.println("Drug Not Found");

return 0;

}

public static int fillPerscription(){

Scanner in = new Scanner(System.in);

perscription toFill = new perscription();

//checks to see if the perscription is valid

System.out.println("Enter The Perscription You Wish To Fill: ");

toFill.patientName = in.next();

for(int i=0;i<pharmList.size();i++){

perscription check = pharmList.get(i);

if(toFill.patientName.equals(check.patientName)){

System.out.println("Perscription Found");

System.out.println("Do You Wish to Fill?(y/n)");

String confirm = in.next();

if(confirm.equals("n")){

return 0;

}

java.util.Date date= new java.util.Date();

check.timeFilled = new Timestamp(date.getTime());

filledList.add(check);

pharmList.remove(i);

System.out.println("Perscription is Ready for pickup in the pharmacy");

return 0;

}

}

System.out.println("Perscription Not Found");

return 0;

}

public static int showPharmList(){

for(int i=0;i<pharmList.size();i++){

System.out.println(pharmList.get(i).patientName+" | "+pharmList.get(i).timePerscribed+" | "+pharmList.get(i).dosePerDay+"mg of "+pharmList.get(i).perscribedDrug);

}

return 0;

}

public static int showFilledList(){

for(int i=0;i<filledList.size();i++){

System.out.println(filledList.get(i).patientName+" | "+filledList.get(i).timeFilled);

}

return 0;

}

public static int removePerscription(){

Scanner in = new Scanner(System.in);

perscription d = new perscription();

System.out.println("Enter The Perscription You Would Like To Delete");

d.patientName = in.next();

for(int i=0;i<pharmList.size();i++){

perscription check = pharmList.get(i);

if(d.patientName.equals(check.patientName)){

System.out.println("Sure You Wish To Delete Perscription For "+check.patientName+"? (y/n)");

String confirm = in.next();

if(confirm.equals("n")){

return 0;

}

//remove drug from current meds

//pat.fName = d.patientName;

int c = 0;

while(c<patientList.size()){

patient pat = patientList.get(c);

if(pat.fName.equals(d.patientName)){

int e = 0;

while(e < pat.medications.size()){

drug old = pat.medications.get(e);

if(old.dName.equals(check.perscribedDrug)){

pat.medications.remove(check.perscribedDrug);

}

e++;

}

}

c++;

}

check.currentMeds.remove();

//remove perscription from list

pharmList.remove(i);

return 0;

}

}

System.out.println("Perscription Not Found");

return 0;

}

public static int administerPerscription(){

Scanner in = new Scanner(System.in);

System.out.println("Enter the patient name: ");

String find = in.next();

int c = 0;

while(c<filledList.size()){

perscription check = filledList.get(c);

if(check.patientName.equals(find)){

java.util.Date date= new java.util.Date();

check.timeAdministered = new Timestamp(date.getTime());

System.out.println(check.dosePerDay+ " of "+check.perscribedDrug+" has been perscribed to "+check.patientName+ " on "+check.timeAdministered);

filledList.remove(check);

administeredList.add(check);

return 0;

}

c++;

}

return 0;

}

public static int showAdministered(){

for(int i=0;i<DrugMain.administeredList.size();i++){

perscription show = DrugMain.administeredList.get(i);

System.out.println(show.dosePerDay+" of "+show.perscribedDrug+ " For "+show.patientName+" at "+show.timeAdministered);

}

return 0;

}

}

package drug;

import java.io.\*;

import java.util.List;

import java.util.Scanner;

//Each User object represents one authorized User of the software. It has four sub-classes, each with

//their own user information and schedule.

public class user {

String username;

String password;

String firstName;

String lastName;

String permission;

String[] schedule;

String page;

user u;

user currentUser;

//static List<User> userDB;

int i = 1;

int command;

int listIndex;

/\*private User(String un, String pw, String fN, String lN, String prmsn) {

username = un;

password = pw;

firstName = fN;

lastName = lN;

permission = permission;

} \*/

/\*

public static String getUsername(User u) {

return u.username;

}

public static String getpassword(User u) {

return u.password;

}

public static String getFirstName(User u) {

return u.firstName;

}

public static String getLastName(User u) {

return u.lastName;

}

\*/

//Method allows the modification of the array containing the User's schedule.

//Precondition: The element index and new string are passed to the Method. The index

//number represents Sunday for 0, Monday for 1, and so on up to 6 for Saturday.

//Postcondition: The array is changed to reflect the User's updated schedule.

public void changeSchedule(int i, String str) {

u.schedule[i] = str;

}

//Method checks the database for the User's username and login pair. If found, it

//displays the home screen that matches the User's permission. If no matching

//username or password is found, it prints "bad username/password" and refreshes

//the login screen.

//Precondition: The entered username and password pair are passed to the method.

//Postcondition: If a matching username and password are found, the User's permission

//is returned. Else, an empty string is returned.

public void login(String un, String pw) {

username = un;

password = pw;

for(int i=0;i<DrugMain.userList.size();i++){

user check = DrugMain.userList.get(i);

if(check.username.equals(username)){

if(password.equals(check.password)){

if(check.permission.equals("doctor")){

DrugMain.doctorView();

}

if(check.permission.equals("nurse")){

DrugMain.nurseView();

}

if(check.permission.equals("admin")){

DrugMain.adminView();

}

if(check.permission.equals("pharmacy")){

DrugMain.pharmView();

}

}

else{

System.out.println("Invalid Password");

}

System.out.println("Invalid Username");

}

}

/\*

if(username.equals("doctor")){

if(password.equals("123")){

DrugMain.doctorView();

return;

}

else{

System.out.println("Incorrect Password");

}

}

if(username.equals("nurse")){

if(password.equals("123")){

DrugMain.nurseView();

return;

}

else{

System.out.println("Incorrect Password");

}

}

if(username.equals("pharmacy")){

if(password.equals("123")){

DrugMain.pharmView();

return;

}

else{

System.out.println("Incorrect Password");

}

}

if(username.equals("admin")){

if(password.equals("123")){

DrugMain.adminView();

return;

}

else{

System.out.println("Incorrect Password");

}

}

if(username.equals("super")){

if(password.equals("123")){

DrugMain.superUser();

return;

}

else{

System.out.println("Incorrect Password");

}

}

\*/

/\*

//listIndex = userList.size();

i = 0;

while(i < listIndex) {

u = userList.get(i);

if(u.username.equals(username)) {

if(u.password.equals(password)) {

permission = u.permission;

if(permission.equals("doctor")) {

DrugMain.doctorView();

} else if(permission.equals("nurse")) {

DrugMain.nurseView();

} else if(permission.equals("administrator")) {

DrugMain.adminView();

} else if(permission.equals("pharmacist")) {

DrugMain.pharmView();

}

}

}

}

\*/

System.out.println("Username/Password incorrect, please try again.");

//display(LoginGUI);

}

//Method scans the entered first name, last name, username, password and permission

//and passes it to the User() method to create a new User object.

//Precondition: The method is called and scans the entered first and last names,

//username, password, and permission.

//Postcondition: The method creates a new User object with those arguments and adds

//it to the database, but does not return anything to the method that called it.

public static int newUser() {

Scanner in = new Scanner(System.in);

user a = new user();

System.out.println("Enter Username");

a.username = in.next();

System.out.println("Enter Password");

a.password = in.next();

System.out.println("Enter Last Name");

a.lastName = in.next();

System.out.println("Enter Permision");

a.permission = in.next();

DrugMain.userList.add(a);

return 0;

}

public static int editUser() {

Scanner in = new Scanner(System.in);

System.out.println("Enter Username To Edit: ");

String un = in.next();

System.out.println("Enter Password To Edit: ");

String pw = in.next();

for(int i=0; i<DrugMain.userList.size();i++){

user check = DrugMain.userList.get(i);

if(un.equals(check.username)){

if(pw.equals(check.password)){

System.out.println("You Have Selected "+check.permission+" "+check.lastName);

System.out.println("What Information Do You Wish To Edit?");

System.out.println("1 For Username");

System.out.println("2 For Password");

System.out.println("3 For Last Name");

System.out.println("4 For Permision");

String command = in.next();

if(command.equals("1")){

System.out.println("Enter New Username: ");

check.username = in.next();

return 0;

}

if(command.equals("2")){

System.out.println("Enter New Password: ");

check.password = in.next();

return 0;

}

if(command.equals("3")){

System.out.println("Enter New Last Name: ");

check.lastName = in.next();

return 0;

}

if(command.equals("4")){

System.out.println("Enter New Permission: ");

check.permission = in.next();

return 0;

}

}

else{

System.out.println("Invalid Password");

}

}

else{

System.out.println("Username Not Found");

}

}

return 0;

}

//Method scans the entered first and last name, then searches the database for the

//matching pair, and then sets that User as the current User.

//Precondition: The method is called and scans the entered first and last name.

//Postcondition: The method sets the User object with matching first and last

//names to the current User variable or prints "User not found."

/\*

public void setCurrentUser() {

Scanner in = new Scanner(System.in);

System.out.println("Enter User's first name");

firstName = in.next();

System.out.println("Enter User's last name");

lastName = in.next();

listIndex = userList.size();

i = 0;

while(i < listIndex) {

u = userList.get(i);

if(u.firstName.equals(firstName)) {

if(u.lastName.equals(lastName)) {

currentUser = u;

System.out.println("Current User is " + currentUser.firstName + " " + currentUser.lastName);

}

} else {

i++;

}

}

System.out.println("Error: User not found");

}

\*/

//Method changes the username of the current User.

//Precondition: The method is called and scans in the new username.

//Postcondition: The current User's username is set to the new username.

public void changeUsername() {

Scanner in = new Scanner(System.in);

System.out.println("Enter new username for current User");

username = in.next();

currentUser.username = username;

System.out.println("Current User's username has been changed");

}

//Method changes the password of the current User.

//Precondition: The method is called and scans in the new password.

//Postcondition: The current User's password is set to the new password.

public void changePassword() {

Scanner in = new Scanner(System.in);

System.out.println("Enter new password for current User");

password = in.next();

currentUser.password = password;

System.out.println("Current User's password has been changed");

}

//Method changes the first name of the current User.

//Precondition: The method is called and scans in the new first name.

//Postcondition: The current User's first name is set to the new first name.

public void changeFirstName() {

Scanner in = new Scanner(System.in);

System.out.println("Enter new first name for current User");

firstName = in.next();

currentUser.firstName = firstName;

System.out.println("Current User's first name has been changed");

}

//Method changes the last name of the current User.

//Precondition: The method is called and scans in the new last name.

//Postcondition: The current User's last name is set to the new last name.

public void changeLastName() {

Scanner in = new Scanner(System.in);

System.out.println("Enter new last name for current User");

lastName = in.next();

currentUser.lastName = lastName;

System.out.println("Current User's last name has been changed");

}

public static int sendPage(){

Scanner in = new Scanner(System.in);

System.out.println("Enter the username you would like to page: ");

String find = in.next();

int i = 0;

while(i<DrugMain.userList.size()){

user check = DrugMain.userList.get(i);

if(find.equals(check.username)){

System.out.println("Enter Page: ");

check.page = in.next();

return 0;

}

else{

System.out.println("User Not Found");

}

}

return 0;

}

public static int readPage(){

Scanner in = new Scanner(System.in);

System.out.println("For Security Please Re-enter your Username");

String un = in.next();

for(int i=0;i<DrugMain.userList.size();i++){

user check = DrugMain.userList.get(i);

if(un.equals(check.username)){

System.out.println(check.page);

}

}

return 0;

}

}

[

drugA

\*

drugB

drugX

drugY

$

sicknessA

5.0

50.0

15.0

]

[

drugB

\*

drugA

$

sicknessB

7.0

100.0

5.0

]

[

drugC

\*

drugX

$

cancer

20.0

30.0

60.0

]

[

drugX

\*

drugA

drugC

$

yellow\_fever

20.0

30.0

20.0

]

[

drugY

\*

drugA

$

pain

20.0

90.0

4.0

]

[

drugZ

\*

$

antibiotic

25.0

30.0

7.0

]

[

Johnson

Smith

\*

drugA

drugB

drugZ

$

404-123-4567

abc\_rd

insure\_usa

123-45-6789

docman

100

none

1

]

[

Jane

tompson

\*

drugB

$

678-123-4567

xyz\_ln

insur\_me

111-11-1111

docman

101

birdflu

2

]

[

Lewis

Ronald

\*

$

404-444-4444

my\_dr

insure\_usa

222-22-2222

docman

102

diabeties

3

]

[

Mark

Johnson

\*

$

688-111-2222

this\_rd

insure\_usa

666119990

docman

105

none

4

]

[

Jim

Elson

\*

$

678-657-6781

my\_rd

insure\_me

098-76-5432

docman

103

none

5

]

[

doc1

111

docman

doctor

-

]

[

nurse1

222

station1

nurse

-

]

[

admin1

333

Bossman

admin

-

]

[

pharmacy1

444

pharmacy1

pharmacy

-

]

**b. GUI Version Source Code**