Project Title: Mednet
Group: Alpha Medical Systems

Team Members:
Christopher Luu
Jakub Pietrasik
Justin Phung
Daniel Sepe
Zijie Qiu(Eric)

Term: Spring 2018
Class: CSC 4350: Software Engineering

1. Planning and Scheduling

1.1. Outline plan

Hardware requirements:

- Computer

Software resource requirements:

- Java/mySQL compiler

The single most serious challenge in developing the project on schedule is whether the project may need to add more features in order to be unique.

Some possible risks are:

- The application may not function as originally desired
- The connection between the database and the interface may not be compatible
- New skills may need to be learned when implementing
- The project topic may need to be slightly altered
- Tasks may need more than the set amount of people

Ways to minimize risk:

- Plan all parts before it is done
- Organize each task onto every team member
- Have good communication

1.2.Scheduling

Tasks	Effort(person-days)	Duration(hours/days)	Dependencies
Login page	1	1 day	
Front-end registration	10	3 days	
Back-end registration	14	10 days	Front-end registration
Doctors home page	10	7 days	
Patients home page	10	7 days	
Patient's ticket submission page	6	1 day	
Front-end doctors open tickets page	9	5 days	
Front-end patients open tickets page	9	5 days	
Back-end open tickets page	14	7 days	Front-end doctors/patients open tickets page
Front-end doctors taken tickets page	9	5 days	
Front-end patients taken tickets page	9	5 days	
Back-end taken tickets	14	7 days	Front-end doctors/patients taken tickets page

2. Problem Modeling

2.1. Problem Statement (User Requirements)

On a high level, our product is an interconnected database for both doctors and patients. It solves the problem of delay when trying to pull up a patient's records, and allows for more convenient methods of information transfer between a patient and doctor. Alternatives to our product is directly going to a hospital and transferring information, or calling the doctors at the hospital. It is compelling and worth developing because it will be really helpful, and there are no other applications like it. The objective of the product is to help out doctors and hospitals along with the patients. Our approach is novel because it includes a more modern approach to healthcare. Possible competitors to this app would be those such as insight optics, medici, digital pharmacist, aetna, and teladoc. This system is possible because our group members are capable at using databases and designing front-end. From a technical point of view, this project is interesting because it allows for both front-end and back-end developers to show all their skills while also understanding the fields of healthcare.

2.2. Use Cases

Use Case: *Register*

Summary: User creates an account that allows them to access Mednet

Actors: Doctor, Patient, and Database

Basic Course of Events

1. User starts application
2. User clicks on register button
3. User is directed to an information entering page
4. User enters information into the specified text fields
5. User clicks register profile
6. If the information entered is correct for a patient, the user will be directed to the patient homepage

Alternative Paths: In step 6, if the information entered is correct for a doctor, the user will be directed to the doctor homepage

Exception Paths: In step 6, if the information entered is incorrect, an error message will be displayed.

Extension Points: Create ticket use case, edit ticket use case, choose ticket use case, edit personal information use case, view doctor's information use case, delete account use case, logout use case.

Assumption: The user does not currently have an account

Precondition: The application must start up

Postcondition: The user obtains an account and is directed to the homepage.

Author: Christopher Luu

Date: 2/19/18

Use Case: *Forgot Password - If the user forgot their password, the should be able to click a button, type in their email address, and be emailed a copy of their password.*

Summary: User is trying to login, but can't remember their password. They should be able to click on a "forgot password" button and provide the email address of the account. From there, an email will be sent to them that contains their password.

Actor: Patient

Basic Course of Events:

1. Completion of registration and login page.
2. User tries to login into the page with their email address as username.
3. They enter in the wrong password and are prompted with a message telling them they're entered a wrong password.
4. After several attempts the user gives up and clicks on the forgot password button.
5. A copy of their password will be copied from the database and emailed to the email address they have provided

Exception Path: The email address the user entered is not registered to an account then an error message will pop up telling them that the email is not registered to an account.

Extension Path: If the user remembers their password, they can attempt to login again.

Preconditions: User must have a registered account with Mednet.

Postconditions: User will reset their password with the link that is sent to their email.

Author: Justin Phung

Date: 2/19/18

Use Case: *Login*

Actors: Doctor and Patient, Database

Summary: The user gains access to the program.

Basic Course of Events:

1. Completion of register use case.
2. User inputs their username into the username text field.
3. User inputs their password into the password text field on the application.
4. User clicks the “login” button.
5. The application compares the values of the username and password fields to ones in the database.
6. If the values correspond to each other then grant the user access.

Alternative Paths: In step 2, if the user forgets their password than an email would be sent to them with the attached password.

Exception Paths: In step 5, if the value does not correspond then go to step 2.

Extension Points: *Edit Personal Information, Create Ticket, Edit Ticket, Choose Ticket, View Doctors Information, Delete Account, Logout.*

Trigger: The user wants to communicate with another user.

Assumptions: The user wants to use the application.

Precondition: The user already completed the *Register* use case.

Postcondition: The user is taken to a home screen.

Author: Jakub Pietrasik

Date: 2/19/2018

Use Case: *Edit Account Information*

Actors: Doctor, patient and database

Summary: Doctor and patient can edit their personal information after they finished the register. The new information will be added to the database.

Basic Course of Events:

1. *Completion of use case Login.*
2. User click on edit information button.
3. User is directed to an information page and their information should show on the page. Base on the data from database.
4. User edit their information in this page.
5. User clicks on submit button.
6. If the information entered is correct, the user will be directed to the homepage and the new information will be add to the database.

Exception Paths: In step 6, if the information entered is incorrect, an error message will be displayed.

Extension Points:
Create Ticket, Edit Ticket, Choose Ticket, View Doctors Information, Delete Account, Log Out.

Trigger: User want to change their personal information.

Assumption: User must have an account.

Precondition: User must have complete use case *"Login"*.

Postcondition: The user changes their personal information and is directed to the homepage.

Author: Eric

Date: 2/18/18

Use Case: *Create Ticket - The user should be able to create a ticket and submit it and after submitting, the ticket should be sent to the database.*

Actors: Patient and Database

Summary: Information that the patients provides the website so that the doctor will be able to help them.

Description:

1. Completion of userID login.
2. Otherwise completion of registration page.
3. On homepage, patient clicks the “create ticket” button.
4. User is asked a series of general questions that they will answer. Questions will include how long they have been experiencing this, when did they first notice, etc. When they answer the question, they will click on the next button it will take them to the next question. The last part will be for them to describe their problem briefly in a text box.
5. When the patient is done they will click on “Submit Ticket” button.
6. If the patient is satisfied with everything they can logout.

Alternative Path: In steps 3 or 4 if the patients wants to delete the ticket mid-way, they can click on a “X” in the top right corner. After step 5, if the patient wants to edit their ticket, they can click the edit ticket button.

Extension Points: Edit ticket, Choose ticket, Edit personal information, delete account, logout.

Trigger: There is something wrong with the user and they want to submit a ticket so that they can be seen by a doctor.

Precondition: Completion of login page or registration page.

Postcondition: The ticket is sent to the database and doctors are now able to see the ticket in queue and choose which patient they want to help.

Author: Justin Phung

Date: 2/19/18

Use Case: *Edit ticket*

Summary: User desires to change, or delete, their previously submitted ticket

Actors: Patient and Database

Basic Course of Events:
1. Completion of use case register
2. Completion of use case login
3. Patient is directed to homepage
4. The patient clicks on edit ticket button
5. The patient changes the information from the previous ticket
6. Patient resubmits ticket
7. The User is redirected to the homepage

Alternative Paths: In step 5, the patient is able to delete the ticket.

Exception Paths: In step 5, if the user submits an empty ticket, an error message will popup.

Extension Points: Use-case logout

Trigger: The patient must have completed create ticket use case.

Assumption: The patient's needs must have changed since the last ticket submission.

Precondition: The register use case, the login use case, and the create ticket use case must be completed successfully.

Postcondition: The ticket inside the database is changed and the user is directed to the homepage.

Author: Christopher Luu

Date: 2/19/18

Use Case: *Choose ticket*

Actors: Doctor

Summary: The doctor request to get a ticket from different patients and the app add the ticket to the doctor's ticket list.

Basic Course of Events:

1. Completion of use case *Log in.*
2. The doctor selects the Choose *Ticket menu option.*
3. Open the *tickets pool* page.
4. Doctor can swap up or down to browse the tickets from different patients.
5. Doctor can click one of these tickets then go to the *ticket detail* page.
6. If the doctor wants to choose this ticket, then they can click the *add* button and this ticket will be add to the doctor's *ticket list*. After being chosen the ticket will be removed from the *tickets pool*.
7. If the doctor doesn't want to choose this ticket, they can click the *back* button to go back to the tickets pool page.
8. Doctor can repeat step 4 – 7 to choose another ticket or click the *back* button to go back to the home page.
9. Doctor can log out by use case *Log out.*

Exception Paths: If the ticket has been chosen before the doctor user click the *add* button, an error message will be displayed, and go back to the *tickets pool page.*

Extension Points: *Edit Personal Information, Choose Ticket, Delete Account, Log Out.*

Trigger: Doctor want to choose a ticket.

Assumption: Patient have already created tickets..

Preconditions: User must have complete use case *"Login"*.

Postcondition: The doctor choose a ticket and go back to the homepage.

Author: Eric

Date: 2/18/18

Use Case: *View Doctors Information*

Actors: Patient and Database

Summary: The patient want to see which doctor chose their tickets and the information of the doctor.

Basic Course of Events:

1. Completion of use case *Login.*
2. The patient click on the *View Doctor* button.
3. Open the *View Doctor* page.
4. Patient can see which doctor chose their tickets and the necessary information of the doctor will show on the page.
5. Patient can go back to homepage by click the *back* button.
6. Patient can log out by use case *Log out.*

Exception Paths: If the ticket has not been chosen by any doctor then the patient can not see a message said ”Your ticket has not been chosen by any doctor”.

Extension Points: *Edit ticket, Choose ticket, Edit personal information, delete account, logout.*

Trigger: Patient want to check which doctor choose their tickets.

Assumption: Patient have already created tickets and any doctor have chosen the ticket.

Preconditions: User must have complete use case *“Login” and “Create Ticket”.*

Postcondition: The patient contact the doctor who have chosen the ticket.

Author: Eric

Date: 2/18/18

Use Case: *View Chosen Ticket*

Summary: The doctor chooses to view his/her chosen ticket and Mednet shows the doctor's chosen tickets

Actors: Doctor and ticket database

Basic Course of Events:

1. Completion of use case register.
2. Completion of use case login.
3. User is directed to homepage.
4. Completion of use case choose ticket.
5. If the doctor has not met the needs of a ticket, the chosen ticket is viewable by the user.

Alternative Paths: In steps 3, the doctor can choose to go directly to step 5 to view chosen tickets.

Exception Paths: If the doctor has met the needs of the ticket and removed it, the ticket is unable to be viewed.

Extension Points: *Logout* use case

Trigger: Use case create ticket completed.

Assumptions: User wants to view the chosen tickets

Precondition: Completion of register use case, login use case, and choose ticket use case.

Postcondition: The information of the ticket is shown to the doctor

Author: Christopher Luu

Date: 2/18/18

Use Case: *Resolve Ticket*

Summary: The user has decided they do not need the ticket or the problem has been addressed.

Actors: Patient, Doctor and ticket database

Basic Course of Events:

1. The ticket has been opened.
2. The user decides that the ticket is no longer needed.
3. The ticket is closed.
4. The user is taken back to their ticket selection page.

Alternative Paths: In step 2, the patient can chose to remove the ticket at any time, even before interacting with the doctor.

Exception Paths: A ticket can only be resolved once.

Extension Points: *Create ticket use case.*

Trigger: Use case create ticket completed.

Assumptions: User wants to remove the chosen tickets

Precondition: Completion of register use case, login use case, and choose ticket use case.

Postcondition: The ticket is displayed as resolved.

Author: Jakub Pietrasik

Date: 2/23/18

Use Case: *Delete Account*

Actors: Doctor and Patient, Database

Summary: The user does not want to continue using the program.

Basic Course of Events:

1. The user is at the account information page.
2. The user clicks the Delete Account button.
3. The user is prompted with a confirmation message.
4. The user is taken to the Login page.

Alternative Paths: None

Exception Paths: In step 3, If the user changes their mind they can chose not to confirm and will be taken back to step 1.

Extension Points: *Login* use case.

Trigger: The user does not want to continue using the program..

Assumptions: The user does not want to use the application any more or ever again.

Precondition: The user is on the account information page.

Postcondition: The user is taken to the login page.

Author: Jakub Pietrasik

Date: 2/19/2018

Use Case: *Logout*

Actors: Doctor and Patient

Summary: The user no longer wishes to use the application or wants to change accounts.

Basic Course of Events:

5. The user is at the home page.
6. The user clicks the Logout button.
7. The user is taken to the Login page.

Alternative Paths: None

Exception Paths: None

Extension Points: *Login* use case.

Trigger: The user wants to change accounts or has completed using the program.

Assumptions: The user does not want to use the application any more.

Precondition: The user is on the home page.

Postcondition: The user is taken to the login page.

Author: Jakub Pietrasik

Date: 2/19/2018

2.3. Test Cases

Test case: **Register**

Description: Check if the information entered is correct and to check if the account is entered into the database.

Test Inputs: Must input string name, int age, string address, string gender, string ethnicity, string email, string password, string previous illnesses, string occupation, string insurance, string times of availability, string security question, and string security answer in the specified text fields.

Expected Results: The system must accept the inputs, and store the user's information into the database.

Dependencies: None

Initialization: Application starts up

Test Steps:

1. Enter the specified information in the specified text fields with no extra information.
    a. The email address will be used as the username.
    b. Password should be 6-10 characters with uppercase and lowercase letters and 1 number.
    c. User should select a predefined security question and provide an answer to that question.
    d. They should submit all other personal information such as name, age, height, etc.
2. Click on the submit button and send the registration form to the database.
3. The database verifies if the information entered is in correct form.
4. The account is saved into the database.
5. The user should be redirected to the homepage if everything is correct. Otherwise, an error message will show that some information was entered incorrectly.

Test case: **Forgot Password**

Description: The purpose of this test case is to help the user change the password in the instance that he/she forgets the password to the account and is unable to access any information.

Test Inputs: The user must click the forgot password button, and input the string email and string security answer when prompted.

Expected Results: New string password will replace the old string password in the database.

Dependencies: This test case is dependent on the use case register account for a valid email address.

Initialization: The application starts up while the user has a registered account in the database.

Test Steps:

1. User clicks on forgot password button.
2. The email address should be entered.
3. If the email is entered correctly, the database should search for the email address.
4. If the email address is found, the system should display the security question that the user chose when they first registered, and provide a text box for the user to enter the security answer.
5. If the answer to the security question is the same format as entered during register use case, a new textbox would appear that would allow the user to alter the current password, in the database, to a nonempty string.

Test case: **Login**

Description: The purpose of this test case is to verify that the user attempting to login has an associated password and username that are registered with the server.

Test Inputs: A username input as a string in the first field and a password input as a string in the second field.

Expected Results: The user should be directed to their designated homepage.

Dependencies: The user must have values entered into each of the fields in the window with the strings in the correct field. This case depends on the user already having a registered account.

Initialization: The user must open the program in order for them to see the login page.

Test Steps:

1. Open the program.
2. Type in the username.
3. Type in the password.
4. Click the login button.
    a. The information will be compared to the information in the database and system will check if the information matches.
5. Either the user will be taken to the homepage or will be given an error message that they have entered wrong information.

Test case: **Edit account information**

Description: Changes the account of the user to match real life changes to the tangible user.

Test Inputs: The informations, string name, int age, string address, string gender, string ethnicity, string email, string password, string previous illnesses, string occupation, string insurance, and string times of availability in the specified text fields.

Expected Results: The information which has been changed should be stored in database and replace the old data.

Dependencies: The user must input data in the specified fields. The user must also have information already saved in the database.

Initialization: The database is loaded with the user's information. User must submit the new information.

Test Steps:

1. The database finds the user information linked with the current user.
2. The user is given a form with his/her current information.
3. The user changes the account information.
4. The user submits the new form to the database.
5. The database overwrites the currently stored information with the new information.

Test case: **Create ticket**

Description: Creates a ticket with the user's medical needs and symptoms, that will be stored in a ticket database

Test Inputs: Must input string patient's medical needs, string symptoms and string medical history.

Expected Results: The ticket will be stored in a database where it can be accessed by the creator of the ticket and multiple doctors.

Dependencies: Register and login use cases completed.

Initialization: The user must submit the ticket to the database.

Test Steps:

1. The user creates a ticket.
2. The user submits the ticket to the database.
3. The database makes the user anonymous.
4. The database stores the ticket in a ticket database.

Test case: **Edit ticket**

Description: Allows the user to edit an already submitted ticket.

Test Inputs: The user submits a new ticket with a different string description than the previously submitted ticket.

Expected Results: The old ticket is deleted from the database and the new ticket is added. The doctors should be able to see the new ticket, not the old ticket.

Dependencies: This test case relies on a patient having an account that has already submitted a ticket otherwise there will be nothing to edit.

Initialization: All database are loaded, and the ticket information is already set in the system..

Test Steps:

1. Patient should login or register an account.
2. After that they should create a ticket and submit it.
3. If they want to make any changes to the ticket they can click on the edit ticket button and it should bring up the ticket they selected with the previously filled out information. From there they should be able to alter the ticket and resubmit it.
4. The old ticket should be deleted from the database and replaced with the new one.
5. Now the doctor should be able to see the new ticket as well.

Test case: **Choose ticket**

Description: The purpose of this test case is to check if doctor can choose patient's ticket correctly.

Test Inputs: Doctor open the *ticket pool* page, choose one ticket and clicks the *add* button.

Expected Results: The ticket which has been chosen will be added to the doctor's ticket list.

Dependencies: Register, login and create ticket use cases completed

Initialization: Doctor must have an account and log in to choose a ticket.

Test Steps:

1. Doctor click on the *Choose Ticket* button.
2. Doctor click one ticket.
3. Check if the ticket has been removed from the ticket pool.
4. Check if the ticket has been added to the doctor's list so that they are able to view it.

Test case: **View doctor's information**

Description: The purpose of this test case is to check if patient can view the doctor's information who has chosen the patient's ticket.

Test Inputs: Patient's ticket has been chosen by a doctor. Patient clicks on the View Doctor button.

Expected Results: Patient can see which doctor chose their tickets and the string contact information and string specialization of this doctor.

Dependencies: Create ticket and choose ticket use cases are completed.

Initialization: Patient must have an account and log in to view doctor's information.

Test Steps:

1. The patient click on the *View Doctor* button.
2. Open the *View Doctor* page.
3. Patient can see which doctor chose their tickets and the necessary information of the doctor will show on the page.
4. Check if the doctor is the one who chose this patient's ticket.
5. Check if the doctor's information are correct, include Name, .

Test case: **View chosen ticket**

Description: Checks the ticket that has been chosen by the doctor

Test Inputs: Doctor have chosen any ticket and clicks on view chosen ticket.

Expected Results: Information on the previously chosen patient's ticket is shown to the doctor.

Dependencies: A ticket must already be chosen for it to be viewed.

Initialization: The chosen ticket is stored in the database and the doctor logs in to view it.

Test Steps:

1. The doctor clicks on view chosen ticket.
2. The database checks on which ticket was chosen by the doctor.
3. The chosen ticket list is displayed to the doctor.
4. The doctor is able to view any chosen ticket out of the many chosen tickets.

Test case: **Resolve Ticket**

Description: If the issue has been resolved the doctor can close the ticket making it unavailable for all other doctors to see.

Test Inputs: Doctor will view a ticket and click the resolve button.

Expected Results: The ticket attribute for availability is changed in view ticket and in the database.

Dependencies: The doctors must have already chosen the ticket.

Initialization: The doctor must choose to view the chosen ticket use case.

Test Steps:

1. The doctor selects the ticket.
2. When and if the doctor feels that the patient's problem has been solved, they can click on the close ticket button.
    a. Doing this should remove the ticket from the choose ticket menus so that other doctors can not open an already solved file. This means that there will be an attribute associated with the ticket that determines if it is solved or not.
    b. The patient should be able to see that the ticket has been resolved as well.
3. The doctor will be taken back to the ticket selection screen.

Test case: **Delete account**

Description: If patient or physician wants to be removed from the database so that their account is no longer active.

Test Inputs: The user will go to edit account information and click on the delete account button.

Expected Results: All user info removed from the database once all necessary “transactions” have been verified to be completed

Dependencies: Register account, login, and edit account information use cases are completed.

Initialization: The user has logged into Mednet, and chooses to edit the account.

Test Steps:

1. The user will login from the homepage.
2. They will click on the edit account information section.
3. The user will click on the delete account button.
    a. Database administrator needs to verify that both patient has completed their part(s) before they are removed from the database.
    For example, if the patient has a pending payment, they will not be removed from the database until their payment has been fully processed (similar to non-repudiation).
        i. Patient must have some form of digital signature once they have agreed to being treated by the physician.
4. The user should be taken back to the front page and should not be able to login again (after administrator reviews all legal transactions).
    a. Patient will have 72 hrs to log back in before their account is permanently deleted.
        i. Administrator(s) should not look at the accounts pending for deletion until after 72 hrs.
5. On the backend, all the information about the user and tickets they have submitted should be deleted.

Test case: **Logout**

Description: If the user no longer wishes to use the application or is satisfied with their time on the application, they have the ability to logout of the application.

Test Inputs: The user clicks the Logout button.

Expected Results: The user is redirected to the login page, and will be unable to use Mednet until the user completes the login use case again.

Dependencies: Register and login use cases must be completed.

Initialization: The user is already logged in and is at the homepage.

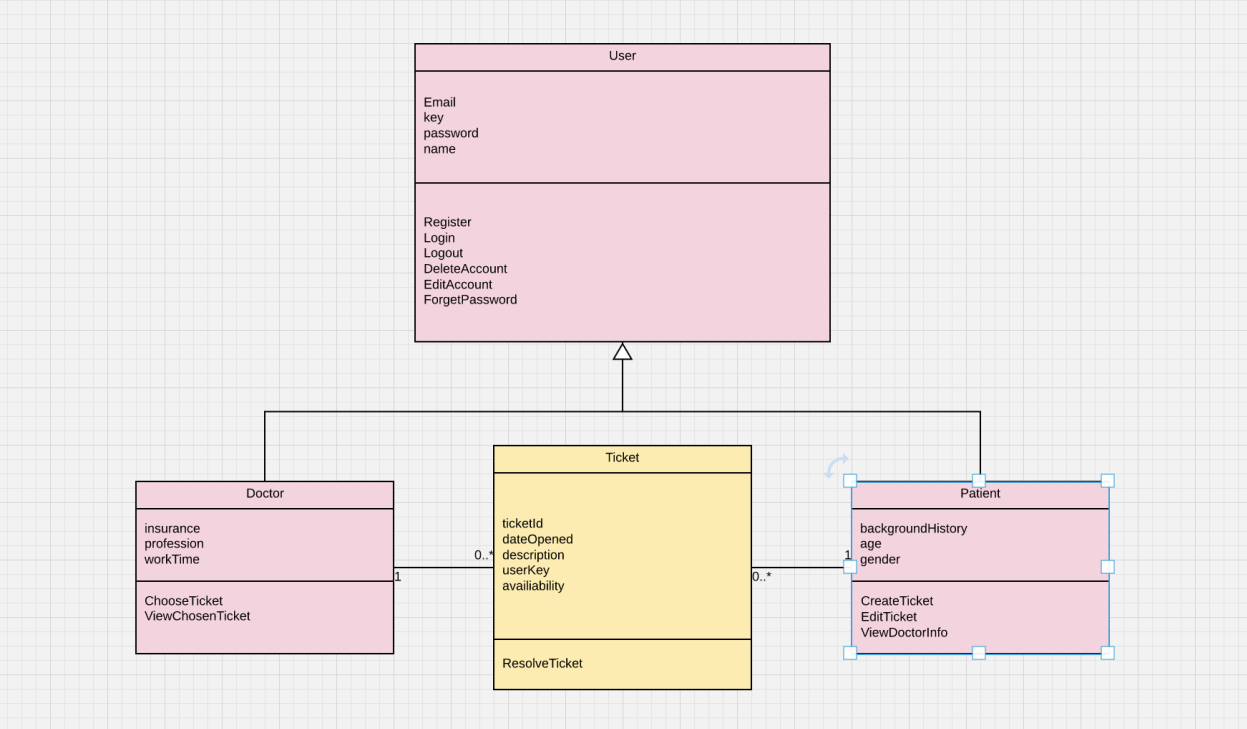
Test Steps:

1. The user is at the home page.
2. The user clicks the Logout button.
3. The user should to taken back to the homepage and be prompted to login again.
    a. Text fields for the username and password should be empty.
    b. Any unsaved work will be lost.

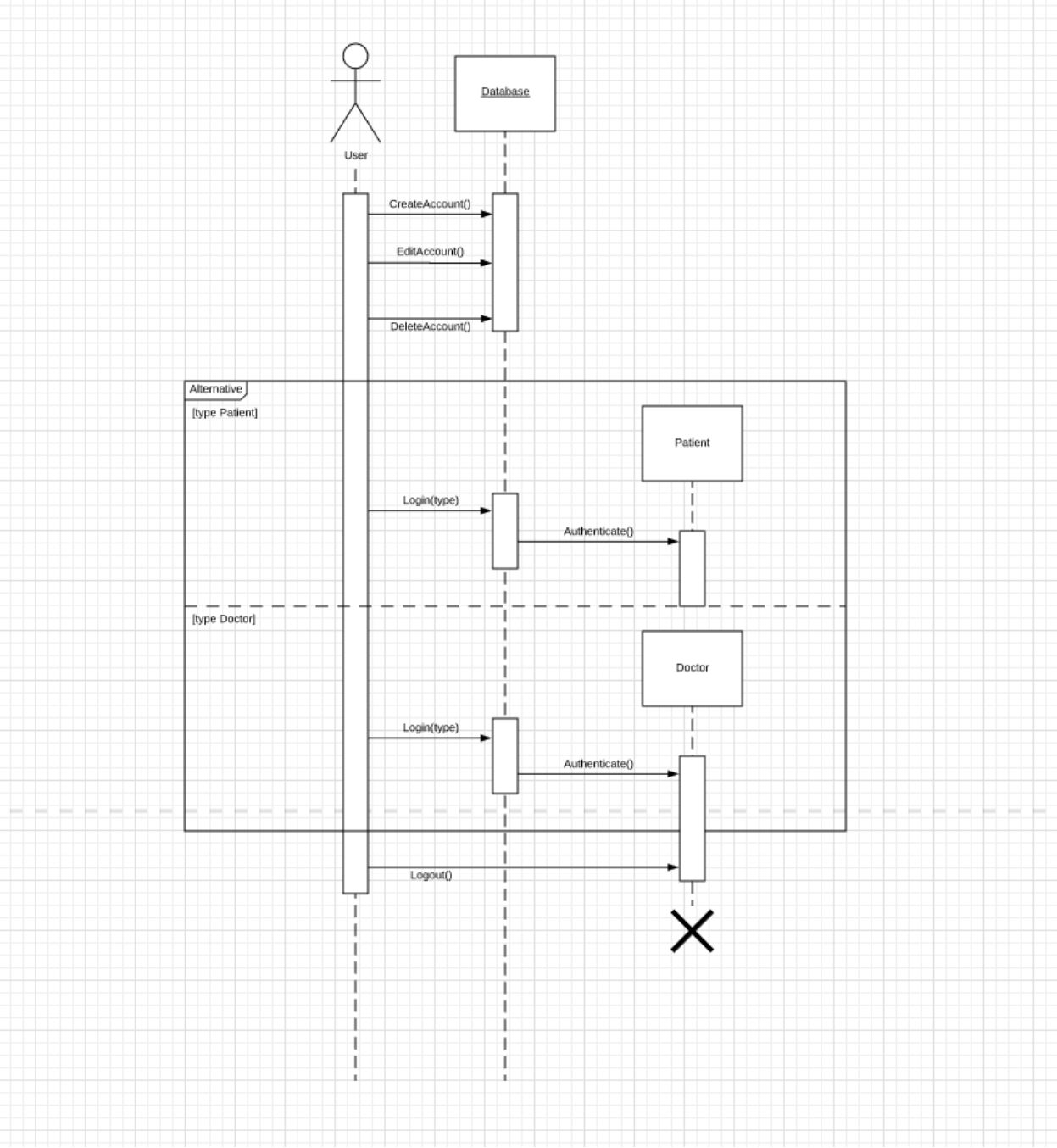
2.4. System Modeling

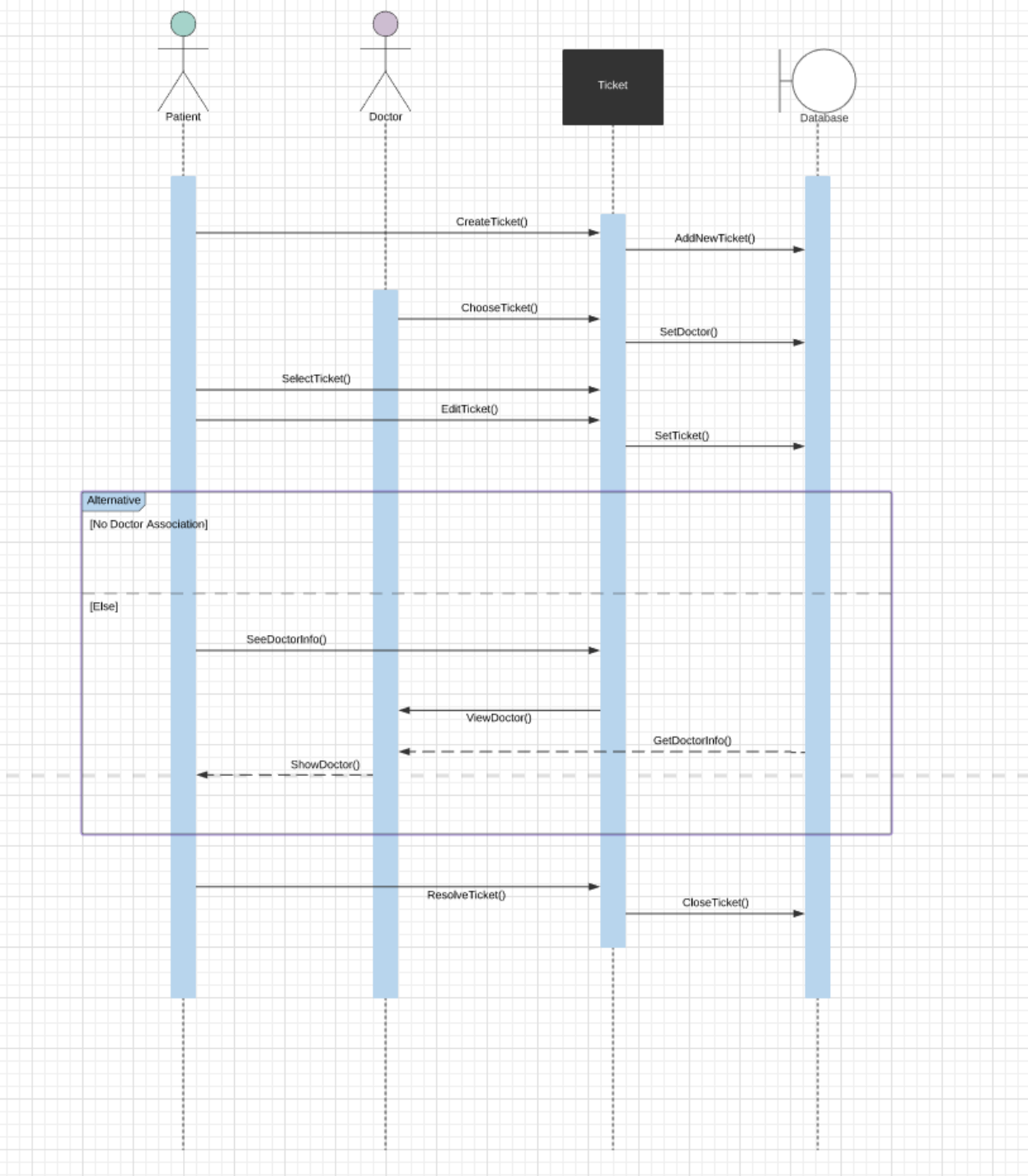
2.4.1. Class Diagram

The objects are the user, which can be a patient or doctor, and a ticket. The doctor object and patient object inherit from the user object, and a ticket object is created from a patient object. A doctor, or patient, is only one user. One patient can have multiple tickets, and one doctor can choose multiple tickets. A user object has the attributes of email, key, password, name, age, and gender. A user has the operation of registering, logging in, logging out, deleting account, editing account, and forgetting a password. A doctor object has the attributes of insurance, profession, and work time. A doctor has the operation of choosing a ticket, and viewing a chosen ticket. A patient object has the attributes of background history. A patient has the operation of creating a ticket, editing a ticket, and viewing a doctor's information. A ticket object has the attributes of ticket ID, date opened, description, user key, and availability. A ticket has the operation of resolving a ticket.

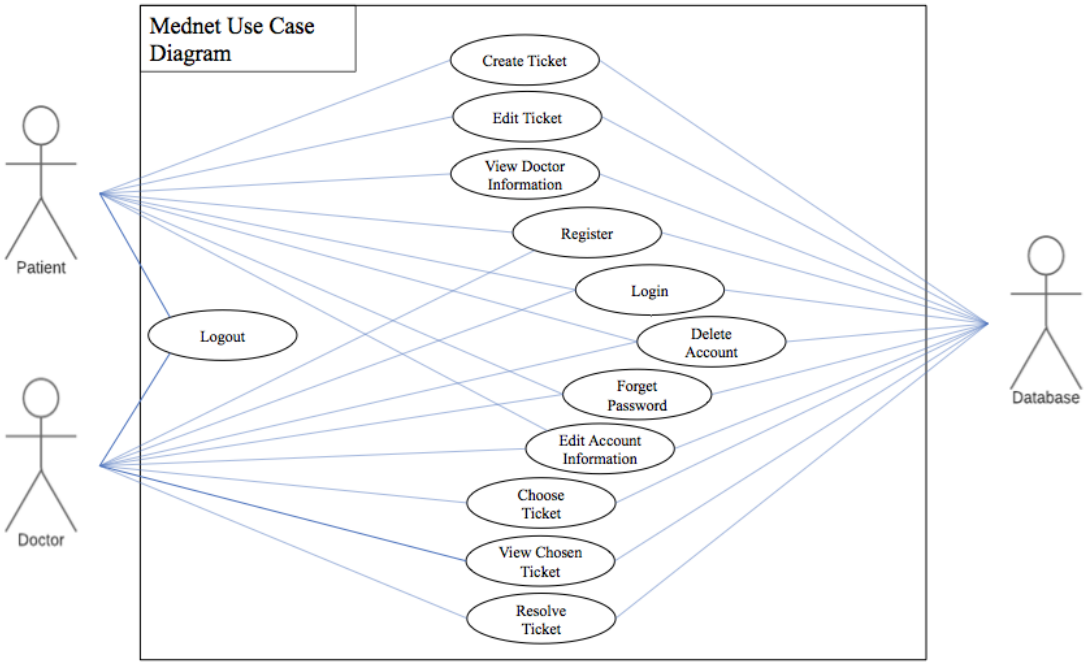


2.4.2. Sequence Diagram




Patient
Doctor
Ticket
Database
CreateTicket()
AddNewTicket()
ChooseTicket()
SetDoctor()
SelectTicket()
EditTicket()
SetTicket()
Alternative
[No Doctor Association]
[Else]
SeeDoctorInfo()
ViewDoctor()
GetDoctorInfo()
ShowDoctor()
ResolveTicket()
CloseTicket()

2.4.3. Use Case Diagram

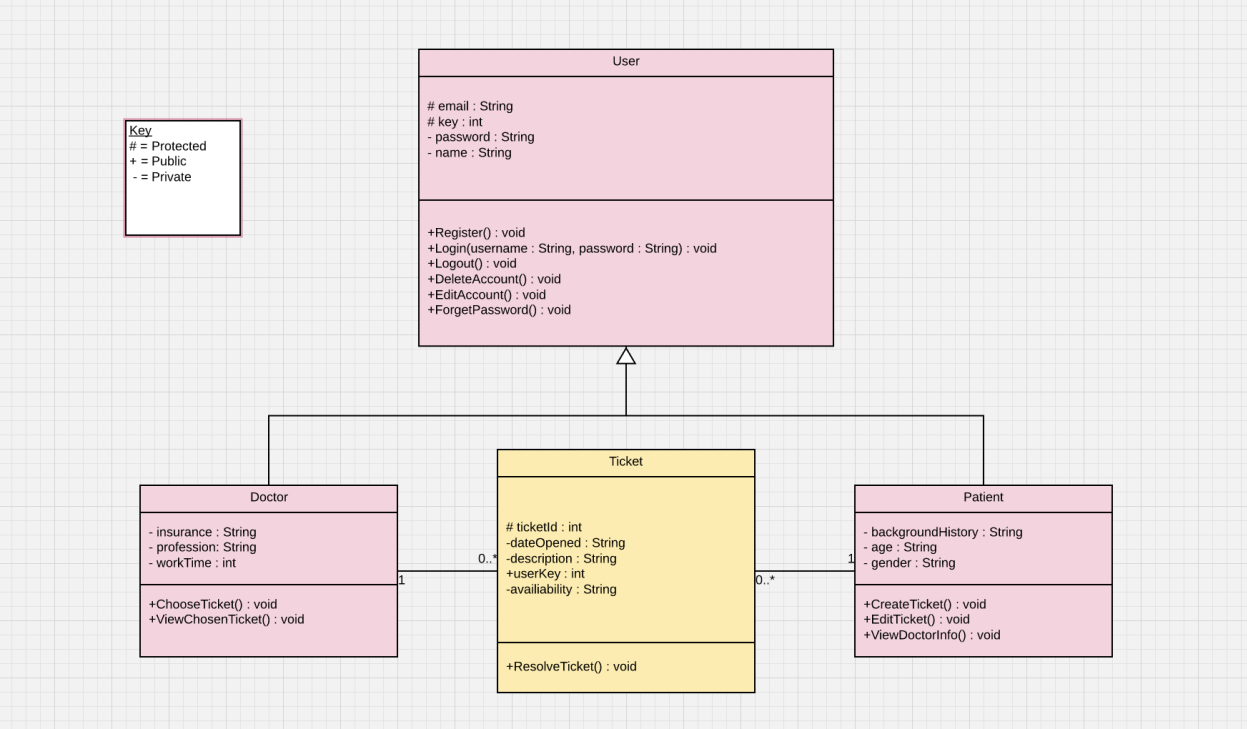


3. System Design

3.1. Architectural Design

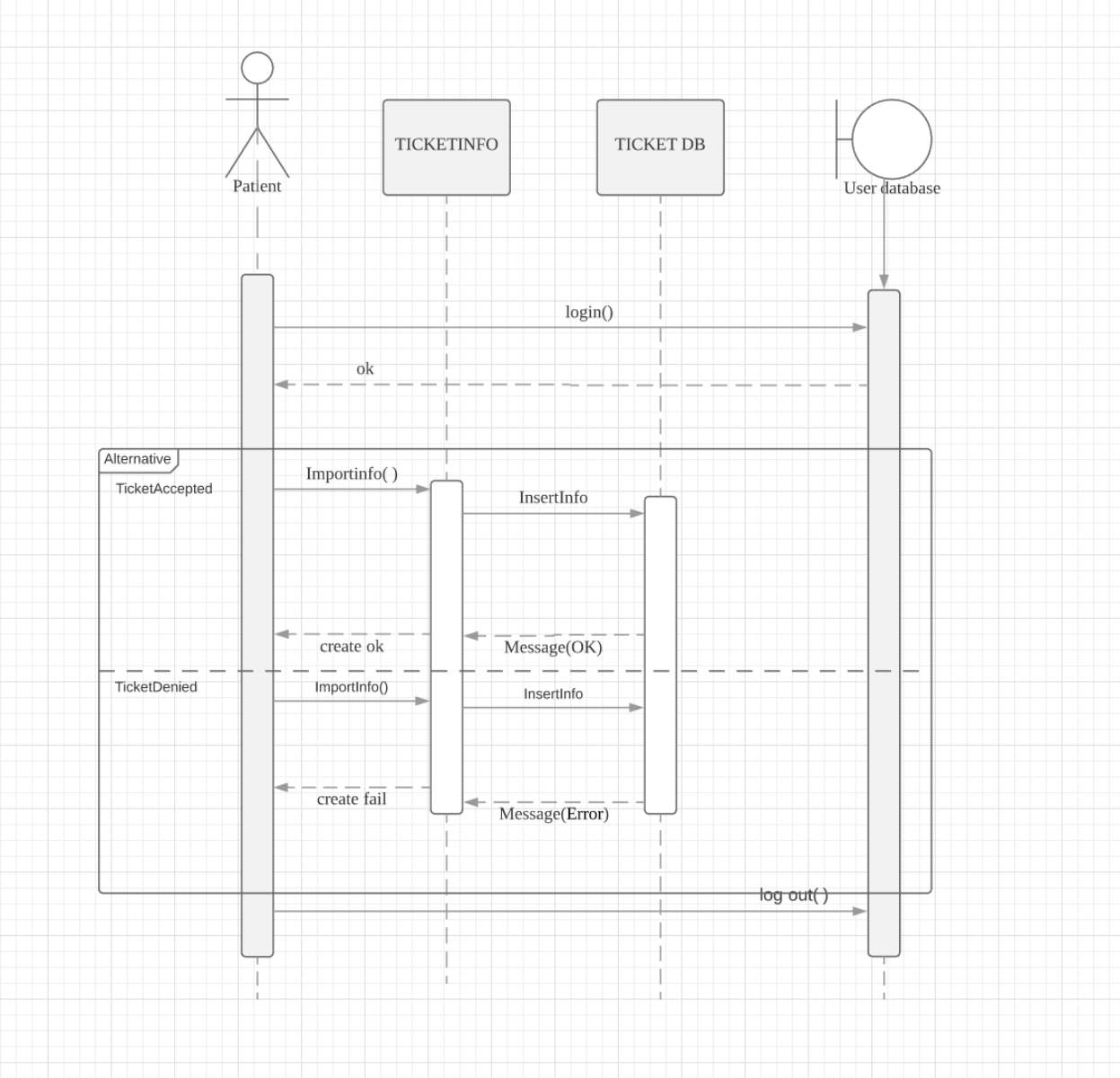
The architectural design pattern is a repository architecture in order to store large amounts of information in a central database where each component of the system can access it.

3.2. Class Diagram

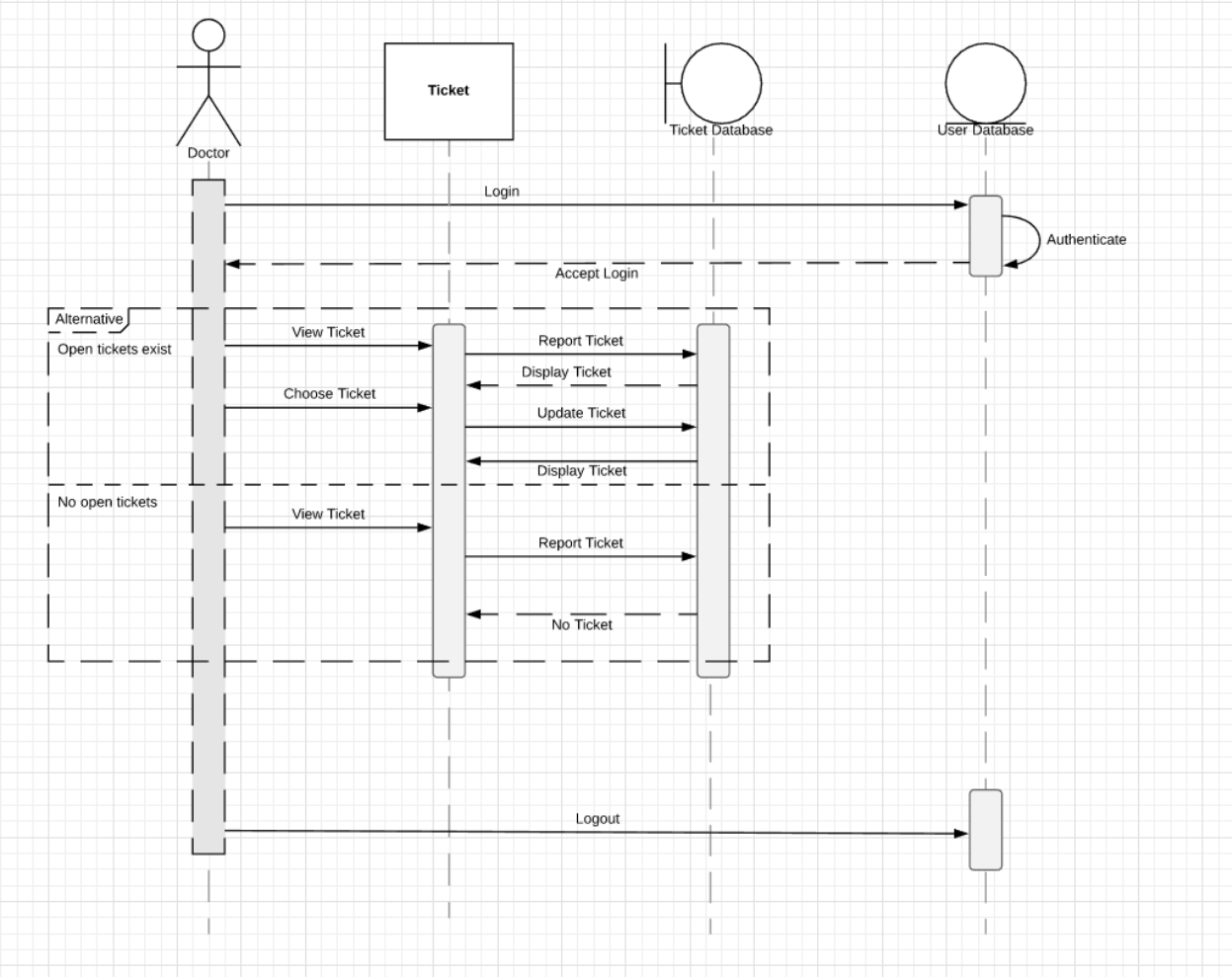


3.3. Sequence Diagram

3.3.1. Create Ticket Use Case



3.3.2. Choose Ticket Use Case



4. Implementation of System Design

To compile the code:

1. Download the build.xml file and the Mednet source code
2. Run the build.xml file as an ant build in any compiler

To use our product, you must:

1. Download the Java Archive(jar) file named Mednet.jar
2. Go into terminal
3. Change the directory to the download directory using the command cd downloads
4. Run the jar file using the command java -jar Mednet.jar
5. The program should start up

Appendix

Youtube channel
https://www.youtube.com/channel/UCyCTB3iVohY0UchLna0DemQ

Video Link
https://www.youtube.com/watch?v=Oswuq3X36wM&feature=youtu.be

Github
https://github.com/LuuChris/HealthcareApp

Slack logs
https://alpha-medical.slack.com/messages/C8XBBU09G/