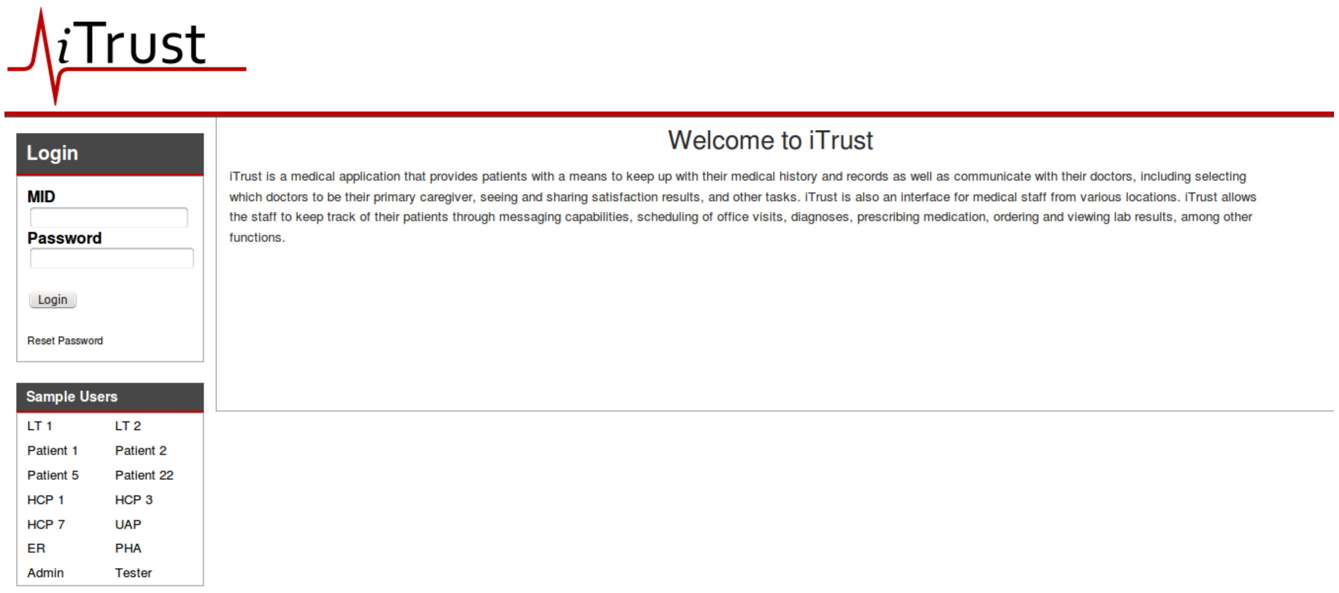
Short Tutorial on iTrust

During the course of this study, you will be working within the iTrust code base. iTrust is a JSP project written on Servlets. More simply, it's a Java program for the web. The affected files related to the tasks you will be performing will all be Java files.

You can find more at http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=start. Below we have included a shorter version describing iTrust. The long version of the requirements document is available at http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements

iTrust is a tool for medical professionals and patients to access, create, modify, and delete records. When you first reach the log-in page, the side bar will contain a log-in form and a list of sample users. For the purposes of this study, it is easiest to click on a sample user and you will automatically be logged in. Each link in this list is appropriately named based on the user time. Afterwards, you will be brought to a customized home page, including a feed of recent activity and a navigation side bar. See the figure below for an example of the login page.

Go to http://agile.csc.ncsu.edu:8081/iTrust/auth/forwardUser.jsp to view this screen in your browser and interact with the demo system. This would be a good idea to do before the study so you familiarize yourself with the user interface and system functions. Just click on the sample users to log in.

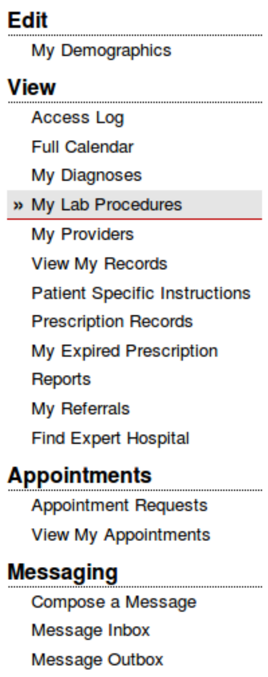


Welcome to iTrust

iTrust is a medical application that provides patients with a means to keep up with their medical history and records as well as communicate with their doctors, including selecting which doctors to be their primary caregiver, seeing and sharing satisfaction results, and other tasks. iTrust is also an interface for medical staff from various locations. iTrust allows the staff to keep track of their patients through messaging capabilities, scheduling of office visits, diagnoses, prescribing medication, ordering and viewing lab results, among other functions.

LT 1	LT 2
Patient 1	Patient 2
Patient 5	Patient 22
HCP 1	HCP 3
HCP 7	UAP
ER	PHA
Admin	Tester

Each classification of users has a different list of options in the side bar, which is broken into categories, listed in bold, with individual pages listed as links below the category heading. Use these links to perform functions throughout the system. An example of the sidebar is shown below.



Although not an exhaustive list, here are some of the functions each classification of users may perform:

- **Patient:** Access medical logs, view upcoming appointments, and message his/her healthcare practitioners.
- **Health Care Practitioner (HCP):** Access and modify patient general information, health information, view list of office visits, schedule appointments, document office visits, and read messages from patients.
- **Lab Tech (LT):** View lab procedures, mark status of procedures as received, and provide results.
- **Emergency Room (ER):** View emergency patient report.
- **Unlicensed Assistive Personnel (UAP):** Edit patient health information, view report requests, and view laboratory procedures.
- **Public Health Agent (PHA):** Monitor adverse events, view diagnosis trends

Sample Bug Task

What you need to do for each of the bug tasks,

1. Read the bug description (and optionally the corresponding use cases if needed)
2. Write the name of the class and method/variable that should be modified to fix the bug so that the expected behavior is met
3. If possible, state how the code should be modified to fix the bug.

You may verbally explain your thought process while you are solving the bug. This will help us a great deal.

Task Sample 1

The validation on the emergency contact name field of the Patient Information page does not work correctly. We want to make sure that the HCPs enter two words separated by a space in the emergency contact name field.

Expected: System checks to see that HCP entered two words separated by a space in the emergency contact name field of the Patient Information page.

Actual: System allows HCP to enter one word.

Related Use Cases: UC10 Subflow S2

http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements:uc10

Typical Scenario to Solve: (*this is given to you because this is a sample task. In the actual study, you will have to think aloud the process you used to solve the tasks*)

Begin in class `EditPatientAction`. From `EditPatientAction.updateInformation`, I noticed it uses a `PatientValidator`. Go to `PatientValidator`. All validation is done in `PatientValidator.validate`.

Enter your answer below

Class name: *`edu.ncsu.itrust.validate.PatientValidator`*

Method/Member variable name: *`validate(PatientEdit p)`*

(Mark with an X inside the appropriate angle brackets like shown in the example below.)

Difficulty Level: <X> Easy | < > Avg | < > Difficult

Confidence: < > Low | <X> Medium | < > High

This is a template of how your tasks will look like in the actual study

Task number

Bug Description goes here

Expected: expected behavior goes here

Actual: actual behavior goes here

Related Use Cases:

names of use cases go here. You will see these use cases inside your eclipse environment. These are optional and are provided only if you need to view additional information.

Enter your answer below

Class name:

Method/Member variable name:

Difficulty Level: < > Easy | < > Avg | < > Difficult

Confidence: < > Low | < > Medium | < > High