**Product Design**

|  |  |
| --- | --- |
| **Team** | **TBD-07c** |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revision Number*** | ***Revision Date*** | ***Summary of Changes*** | ***Author(s)*** |
| *1* | *09/17/2016* | *Initial revision* | *Matt Maloney* |
| 2 | 09/18/2016 | Imported finalized Sequence Diagrams, and finished design rationale section. | Matt Toro |
| 3 | 10/02/2016 | update class diagram, and sequence diagram | Alex Huang |
| 4 | 11/30/2016 | Updated Class Diagram to properly represent the application | Matthew Toro |
| 5 | 12/4/2016 | Finalized Product Design, update Components and Functions | Matthew Toro |

# **Architectural Model**

This diagram represents the major subsystems of the product. Initially focus on the domain layer and its components before decomposing the user interface component. Note that a common interface allows both the GUI and a Command Line Interface to access the domain model in the same manner without regard to the type of presentation technique.



# **Components and Functions**

|  |  |
| --- | --- |
| Accounts | <Component state   * This component contains all the information for user sign up. All patients, admins, doctor, nurses, and users are defined here. This is the main component that communicates with the others.   Component behavior   * This component handles when users login and logout. It also handles updates to profiles and provides the ability to search for patients> |
| Communications | <Component state   * Communications is responsible for Patient, Doctor, and Nurse communications through appointments and messages   Component behavior   * This component allows Patients and Doctors to create appointments with each other. It allows Nurses to view these appointments for the current week. This component also allows users within the system to send and receive messages from other users within the system> |
| Logger | <Component state   * This components keeps track of all system changes   Component behavior   * Allows Admins to view activities within their hospital such as Doctors scheduling appointments for a patient. Organizes these activities into Daily Logs for ease of use |
| Records | <Component state   * This components holds all Patient Medical Information. All patient prescriptions, and test are defined here   Component behavior   * Gives doctors the ability to add a prescription or a test to a patient. The Doctors and Nurses also have the ability to view/update the patient’s medical information. Only Doctors can release test and delete prescriptions from a Patient Record |

# **Class Diagram(s)**



# **Sequence Diagram(s)**

# **Design Rationale**

More issues surely will expose themselves once the coding begins, but as of the initial revision of this document, the toughest decision has been the class structure of a new user. We were contemplating as to whether a new user should be represented as a user object which contains a register function. This would allow us to keep track of the use even if they don’t register and allow for registering a user to just interact with the system. Our other option was to create a register class that any user of the site would access instead. Meaning anytime someone wanted to register they interacted with the register object instead. We came to the decision to use the register object, but when implementation of the code begins we may choose differently. We decided this because we are not sure how to implement these functionalities exactly,and it seems like it will be easier to implement this way. Using the register class to create new user objects rather than having a register function within the user class.