**Product Design**

|  |  |
| --- | --- |
| **Team** | Bouncing Bovines |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revision Number*** | ***Revision Date*** | ***Summary of Changes*** | ***Author(s)*** |
| 0.1 | 02/17/2016 | Started work on document | All members |
| 0.2 | 02/23/2016 | More work on Class diagram | All members |
| 0.3 | 02/23/2016 | Ongoing additions to sequence diagrams | All members |
| 0.4 | 03/05/2016 | R1 Revisions | Ryan Morrissey |
| 0.5 | 03/30/2016 | R2 Planning Revisions | All members |
| 1.0 | 05/09/2016 | R2 Revisions | Ryan Morrissey |

# **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**

|  |  |
| --- | --- |
| Person | Component state   * ID * Email * Password * Name * Phone Number   Component behavior   * Delete Account |
| Patient (is a Person) | Component state   * Emergency Contact Name * Emergency Contact Number * Date of Birth * Insurance Provider   Component behavior   * View Medical Information * Create Appointment * Modify Appointment * Cancel Appoint * Add Medical Document |
| Staff (Is a Person) | Component state   * Salary * Hours   Component behavior   * None |
| Doctor (Is a Staff) | Component state   * None   Component behavior   * Remove Patient * Modify Appointment * Create Appointment * Cancel Appointment * Add Medical Document * Transfer Patient |
| Nurse (Is a Staff) | Component state   * None   Component behavior   * Add Medical Document * Modify Appointment * Create Appointment |
| Manager (Is a Staff) | Component state   * None   Component behavior   * Add Staff * Remove Staff * Assign Patient * Transfer Patient |

# **Class Diagram(s):**

Link to the Class diagram:

<http://i.imgur.com/4xlVVAJ.png>

# **Sequence Diagram(s)**

(UC-01) Registration http://i.imgur.com/Ai7ZTTF.png

(UC-02) Update Patient Profile Information http://i.imgur.com/oIAJchE.png

(UC-03) Update Patient Medical Information http://i.imgur.com/Go2Ypdu.png

(UC-04) Administrator Registration http://i.imgur.com/VIIKd6i.png

(UC-05) Create or Update Patient Appointment http://i.imgur.com/ETwgIq4.png

(UC-06) Add Prescriptions http://i.imgur.com/jcVPyEQ.png

(UC-07) Remove Prescriptions http://i.imgur.com/R8DMhLB.png

(UC-08) Cancel Patient Appointment http://i.imgur.com/CMzmQCH.png

(UC-09) View Patient Medical Documents http://i.imgur.com/ze7pAwb.png

(UC-10) View Patient Prescriptions http://i.imgur.com/wqIa2jq.png

(UC-11) Patient Transfer http://i.imgur.com/J8eq6gP.png

(UC-12) Upload Patient Information <http://i.imgur.com/4QMsTBF.png>

(UC-13) Release Test Results <http://i.imgur.com/KqfIFpr.png>

(UC-14) View Activity Logs <http://i.imgur.com/33Q1lXB.png>

(UC-17) View Stats http://i.imgur.com/pAJ9m0E.png

(UC-18) Admission/Discharge from hospital http://i.imgur.com/rse1pZM.png

Album **-** http://imgur.com/a/cBnOV

# **Design Rationale**

* Understanding the scope of the project.
* Tests are viewed through medical documents.
  + Should we make a new class for tests?
  + Pro: would help when differentiating and making lists between different types of medical records.
  + Con: requires an additional subsystem to handle recording, submitting, and viewing tests that would be almost identical to that of generic medical documents.
  + We decided that while having a different list for the tests could be useful in the future, the additional work to essentially remake the medical documents class outweighs the benefits at this time.
* Method of applying admission and discharge.
  + How should we record patients staying at the hospital for an extended amount of time?
  + We thought about making a class for hospital stay that would have attributes such as : patient, reason for stay, date of admission, etc. This class would have a \* to 1 relationship with a hospital.
  + We ran into problems deciding how these admissions would be created and handled. We decided to move this feature to R2 because we are unsure of our abilities. If we have extra time before the first release we may revisit this idea.
* What features to include in the first release.
  + Do we want to substitute functionality for more features.
  + Pro: Release one would include more features that the customer requires.
  + Con: Basic functionality might not be as smooth.
  + We decided to only include major functions and no additional features. We want the customer to be pleased with the overall function of the system and allow patients to begin registering and entering basic account information.
* While making our class diagram we realized that we had missed some Use Cases that we will need for R1 so we went back and finished them before working on the related sequence diagrams.
* We had to decide as a group how we wanted to standardize the look of our sequence diagrams so that we would all get the most out of them.
* The action of logging an activity was not given its own use case because it seems to function more as an automatic rider to other use cases. The User does not actively have to log the activities so it might work better as a background process.