Abstract

This document describes the software requirements for Callisto. Callisto is a plug-in for Jupyter which enables analysis of data from the MIMIC-III database [1]. The requirements in this document satisfy the user requirements laid out in the URD [2] and form a developer’s view of the system. These requirements are compiled into a logical model consisting of a class model, transition models and sequence diagrams. This document complies with the ESA software standard.
Contents

1 Introduction ................................................................. 9
   1.1 Purpose ........................................................................ 9
   1.2 Scope .......................................................................... 9
   1.3 List of definitions and abbreviations ......................... 10
       1.3.1 Definitions ......................................................... 10
       1.3.2 Abbreviations ..................................................... 11
   1.4 List of references ..................................................... 11
   1.5 Overview ...................................................................... 11

2 General Description ......................................................... 12
   2.1 Relation to current projects ..................................... 12
   2.2 Relation to predecessor and successor projects ........ 12
   2.3 Function and purpose ............................................... 12
   2.4 Environment ........................................................... 12
   2.5 Relation to other systems ........................................ 13
       2.5.1 Jupyter Notebook ............................................ 13
       2.5.2 Jupyter Hub ....................................................... 13
       2.5.3 MIMIC-III ......................................................... 14
   2.6 General constraints .................................................. 14
       2.6.1 Security and Privacy ........................................ 14
       2.6.2 Language ........................................................ 14
   2.7 Model description ..................................................... 14
       2.7.1 Environment Model ........................................... 14
       2.7.2 Class Diagram .................................................. 16
       2.7.3 Data Model ........................................................ 21
       2.7.4 Sequence Diagrams ........................................... 24

3 Specific requirements ....................................................... 37
   3.1 Functional requirements .......................................... 37
       3.1.1 Example ............................................................ 37
       3.1.2 Admin ............................................................... 37
       3.1.3 User ................................................................. 38
       3.1.4 Preset ............................................................... 40
       3.1.5 Document ........................................................ 40
       3.1.6 CallistoCell ......................................................... 41
       3.1.7 Cell ................................................................. 43
       3.1.8 Filter ................................................................. 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.9</td>
<td>Graph</td>
<td>43</td>
</tr>
<tr>
<td>3.1.10</td>
<td>Query</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>Requirements traceability matrix</td>
<td>46</td>
</tr>
<tr>
<td>4.1</td>
<td>URD to SRD</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>SRD to URD</td>
<td>49</td>
</tr>
<tr>
<td>A</td>
<td>Appendix - User Interface Mockups</td>
<td>51</td>
</tr>
<tr>
<td>A.1</td>
<td>Log In screen</td>
<td>51</td>
</tr>
<tr>
<td>A.2</td>
<td>Callisto Cell</td>
<td>51</td>
</tr>
<tr>
<td>A.3</td>
<td>Different types of graphs</td>
<td>54</td>
</tr>
<tr>
<td>A.4</td>
<td>Export to Word</td>
<td>57</td>
</tr>
<tr>
<td>B</td>
<td>Transitions</td>
<td>58</td>
</tr>
</tbody>
</table>
DOCUMENT STATUS SHEET

GENERAL

Document title: Software Requirements Document v1.1
Identification: SRD/1.1
Authors: S. Lovett, T. van Schijndel
Reviewers: G. Foks
Document status: Final version
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.1</td>
<td>19-09-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Initial document structure</td>
</tr>
<tr>
<td>0.0.2</td>
<td>20-09-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Wrote text for the introduction and general description</td>
</tr>
<tr>
<td>0.1.0</td>
<td>21-09-2016</td>
<td>T. van Schijndel, S. Lovett, G. Foks</td>
<td>Added requirements</td>
</tr>
<tr>
<td>0.2.0</td>
<td>27-09-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Improved document layout and text and added sequence diagrams</td>
</tr>
<tr>
<td>0.2.1</td>
<td>5-10-2016</td>
<td>T. van Schijndel, S. Lovett, K. W. Man</td>
<td>Improved document based on feedback from N. Zannone, added an improved calls diagram and edited specific requirements.</td>
</tr>
<tr>
<td>0.2.2</td>
<td>1-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Added an improved specific requirements and traceability matrix. Also improved overall document.</td>
</tr>
<tr>
<td>0.2.3</td>
<td>6-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Improved specific requirements and class diagram.</td>
</tr>
<tr>
<td>0.2.4</td>
<td>12-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Improved sequence diagrams and added transition diagrams.</td>
</tr>
<tr>
<td>0.2.5</td>
<td>15-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Added descriptions to sequence diagrams and added transition diagrams.</td>
</tr>
<tr>
<td>0.2.6</td>
<td>18-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Added datamodel and improved sequence diagrams.</td>
</tr>
<tr>
<td>1.0</td>
<td>21-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Checked and improved entire document.</td>
</tr>
<tr>
<td>1.1</td>
<td>31-10-2016</td>
<td>T. van Schijndel, S. Lovett</td>
<td>Implemented feedback from Zannone.</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Author(s)</td>
<td>Reason</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>1.1.1</td>
<td>10-11-2016</td>
<td>K.W. Man</td>
<td>Removed a typo in 2.7.3. Corrected my own name.</td>
</tr>
</tbody>
</table>
## DOCUMENT CHANGE RECORDS

<table>
<thead>
<tr>
<th>Section</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7.2</td>
<td>Improved the Class diagram and added more elaborate descriptions for all the classes, functions and attributes.</td>
</tr>
<tr>
<td>3.1</td>
<td>Added an example for the layout for the functional requirements. Renumbered all the functional requirements. Made descriptions for the requirements more elaborate.</td>
</tr>
<tr>
<td>4</td>
<td>Because of the renumbering of the requirements, the numbers here changed as well.</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Added extra pictures for the different graphs, explained what happens when a login fails and added a part on exporting to word. Appendix A.2 was changed to Appendix B.</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Captions for the figures changed.</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 PURPOSE

This Software Requirements Document (SRD) will be the link between the User Requirements Document (URD) and the Callisto software. It will provide a translation of the specific user requirements found in section 3 of the URD [2] into software requirements found in sections 3 of this document. Whereas the URD was all about the customer’s wishes, the SRD is all about the developers’ view on what the software product must do. The user requirements explicitly state what the user wants from the software product, whereas the software requirements are the requirements the developers place on the software product to make sure the user requirements are met.

Specifically, the software requirements dictate, in developer’s terms, what The Callisto plug-in must accomplish, but not how. Therefore it is implementation-independent. This document will give a simplified view of the systems content and behaviour by providing logical models of the system.

1.2 SCOPE

The Pyoneers is a Bachelor End Project group working for the TU/e and Royal Philips. The software product is a plug-in for Jupyter which is created by The Pyoneers for the collaborative initiative lead by the Research of Chronic Disease Management department at Royal Philips, and the department of Mathematics and Computer Science Flagship Data Science of Eindhoven University of Technology. The goal of the application is to develop a data science environment in which the researchers can use the data in the MIMIC-III database. The application will allow the user to generate graphs from the MIMIC-III data in an easy and user friendly manner. Text can be added to the graph to clarify the results that the graph represents. This will help researchers in gaining insights into relevant correlations. In the end this will all contribute to making it easier for the scientists to do research using the MIMIC-III database [1] to find insights in health care and treatment processes. When the researcher feels that the document is finished, they can export the document to Microsoft Word (.docx) format to easily share their findings. They are also able to export the charts, with the underlying settings, to MATLAB code. Furthermore, Callisto will allow scientists to save and resume their work.
# LIST OF DEFINITIONS AND ABBREVIATIONS

## DEFINITIONS

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callisto Cell</td>
<td>A Jupyter cell which contains the graphical user interface for the creation of the graph and the graph itself. Furthermore it contains options for exporting to Microsoft Word and MATLAB. Referenced in the URD as 'Mimic cell'.</td>
</tr>
<tr>
<td>Disease class</td>
<td>A disease class contains all the diseases of a certain category.</td>
</tr>
<tr>
<td>Jupyter Notebook</td>
<td>A web application that allows the creation of live code, visualisations and text.</td>
</tr>
<tr>
<td>Jupyter Project</td>
<td>A document in the Jupyter Notebook which can contain a set of cells.</td>
</tr>
<tr>
<td>JupyterHub</td>
<td>An application which enables an instance of Jupyter Notebook for every user which logs in.</td>
</tr>
<tr>
<td>MATLAB</td>
<td>A multi-paradigm computing environment for numerical data.</td>
</tr>
<tr>
<td>MIMIC Graph</td>
<td>A graph created with the data of the MIMIC-III database.</td>
</tr>
<tr>
<td>Philips</td>
<td>A company focused on improving people's lifestyle with meaningful innovations in healthcare, consumers lifestyles and lighting.</td>
</tr>
<tr>
<td>Python</td>
<td>Python is a widely used high-level, general-purpose, interpreted, dynamic programming language.</td>
</tr>
<tr>
<td>Top 6 disease classes</td>
<td>The top 6 disease classes that are in the MIMIC-III database. The 6 classes are: Cardiovascular disease, chronic lung disease, chronic kidney disease, cancer, diabetes, injuries.</td>
</tr>
</tbody>
</table>


1.3.2 ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>MIT</td>
<td>Massachusetts Institute of Technology</td>
</tr>
<tr>
<td>TU/e</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>URD</td>
<td>User Requirements Document</td>
</tr>
<tr>
<td>SRD</td>
<td>Software Requirements Document</td>
</tr>
<tr>
<td>MIMIC</td>
<td>Medical Information Mart for Intensive Care</td>
</tr>
<tr>
<td>MoSCoW</td>
<td>Must have, Should have, Could have and would like but Won't have</td>
</tr>
</tbody>
</table>

1.4 LIST OF REFERENCES


1.5 OVERVIEW

The remainder of this document is split into three chapters, as defined by the ESA Engineering Standards [4]. Chapter 2 (General Discussion) gives an overview of the Callisto plug-in, including its relation to current, past and future projects, and the function and purpose of the system. Here we also list the systems with which the Callisto plug-in will interact, general constraints and background information, and a description of the logical model. Chapter 3 (Specific Requirements) gives a list of all functional requirements, defining what the system should do, and constraints on these in the form of non-functional requirements. Chapter 4 (Requirements Traceability Matrix) shows how each user requirement of the URD [2] are linked to the software requirements of chapter 3.
2 GENERAL DESCRIPTION

2.1 RELATION TO CURRENT PROJECTS

There are currently no project which this project relates to.

2.2 RELATION TO PREDECESSOR AND SUCCESSOR PROJECTS

This project is not related to any predecessor project. This project could be followed by a successor project which would implement the newest version of the MIMIC-III database and make it compatible with this project. Further project may also use the Callisto plug-in to draw insight from other medical information databases; Philips Health have expressed a desire to collect data for an intensive care database for patients in the Netherlands.

2.3 FUNCTION AND PURPOSE

The software product is a plug-in for Jupyter Notebook which is created by The Pyoneers for the collaborative initiative lead by the Research of Chronic Disease Management department at Royal Philips, and the department of Mathematics and Computer Science Flagship Data Science of Eindhoven University of Technology. It is a data science environment in which the researchers can use the data in the MIMIC-III database. The application will allow the user to generate graphs from the MIMIC-III data in an easy and user friendly manner. Text can be added to the graph to clarify the results that the graph represents. This will help researchers in gaining insights into relevant correlations. When the researcher feels that the document is finished, they can export the document to word to easily share their findings. They are furthermore able to export the charts, with the underlying settings, to Matlab code.

In the end this will all contribute to the purpose of this project: making it easier for the scientists to do research using the MIMIC-III database [1] to find insights in health care and treatment processes.

2.4 ENVIRONMENT

To have the MIMIC-III database accessible for all researchers we store in on a (Linux)server. This way they don’t all have to download the entire database. Jupyter-hub is stored on the server as well. The researchers can access Jupyter-hub from a web-browser and then log in. Once logged-into Jupyter-hub they can use the data from the MIMIC-III database. The admin will create accounts for researchers who should get access to the database that will allow the researchers to log-into Jupyter-hub.

The following users will use the system:
Researchers

The Callisto plug-in will mainly be used by clinical researchers trying to gain insights into healthcare by using the data from the MIMIC-III database. These users will use the Callisto plug-in to get information from the MIMIC-III database and help them to gain insights into relevant correlations. To achieve this the Callisto plug-in will enable them to filter data from the database and create different kinds of graphs. Researchers would also like to be able to easily share their findings to their colleagues. The Callisto plug-in will allow them to share their findings easily by means of exporting their Jupyter file to word. Since word documents are widely used all over the world, almost everyone will be able to open the document. They can then use this document as a basis for a report or publication if they so desire.

Database Administrator(s) To make sure that only the people with the right credentials are able to access the data from the MIMIC-III database, the database should be user account protected. The database admin checks if the person who applies for the password has obtained the MIMIC-III MIT certificate and is therefore allowed to use the database. This admin can then create a user account with username and password for the applying user.

2.5 RELATION TO OTHER SYSTEMS

The Callisto plug-in is an extension to Jupyter Notebook, and is capable of retrieving data from the MIMIC-III database [1]. Jupyter Hub is used to handle multiple users.

2.5.1 JUPYTER NOTEBOOK

Jupyter Notebook is a web application that enables the creation of documents containing a mixture of live code and markdown text. Live code segments run segments of Python code, for example to produce data, run simulations, or produce visualisations. Jupyter Notebooks are run on a server, allowing single users to access them. The Callisto plug-in is executed within Jupyter Notebook as a set of extensions and live code.

2.5.2 JUPYTER HUB

JupyterHub is a multiuser version of the notebook designed for centralised deployments in companies, university classrooms and research labs. The multi-user hub, JupyterHub, spawns and manages Jupyter Notebook server instances allowing for multiple users to use notebooks concurrently using the same physical server. It will provide an authorisation structure with user accounts that can be regulated by the admin.
2.5.3 MIMIC-III

MIMIC-III is a critical care database containing medical care records for over 40,000 critical care patients, developed by the MIT Lab for Computational Physiology. The Callisto plug-in is required to produce visualisations of data from this dataset. An instance of the database can be constructed and accessed using a variety of technologies such as Postgres, MySQL, and Oracle. An instance of the database shall then accept queries from The Callisto plug-in, execute them and respond with the results of the supplied queries.

2.6 GENERAL CONSTRAINTS

The Callisto plug-in is intended to be used by researchers at Philips and the TU/e. The researchers will have a technical background, but will not necessarily be familiar with editing and creating SQL queries, so there should be a simple graphical user interface.

2.6.1 SECURITY AND PRIVACY

To make sure the MIMIC-III database is only used by users licenced by MIT, the database is only accessible with the right user credentials. An admin will keep track of what users are licenced by MIT and create user credentials for those users only. Moreover, all Internet connections are encrypted to prevent eavesdropping and therefore preventing unauthorised parties gaining access to the MIMIC-III database.

2.6.2 LANGUAGE

The Callisto plug-in is written in English, since most clinical scientists will be able to understand the English language.

2.7 MODEL DESCRIPTION

This section describes the proposed system and its intended functionality from a logical viewpoint using an environment model, a class model, and sequence diagrams.

2.7.1 ENVIRONMENT MODEL

The Callisto system has two main components: the Callisto server, and the client connecting to it. The client, operated by a user, connects to the web interface provided by Jupyter Notebook via their locally installed web browser, which is responsible for rendering the Notebook user interface. Jupyter Notebook instances are spawned and managed by the Jupyter Hub module, which uses the user administration module to manage user access. The user administration module can be accessed only by administrators, and is used to manage user accounts.
In addition to its usual functionality, Jupyter Notebook is extended by the Callisto plugin, which consists of three modules: the Callisto cell, .docx export module and MATLAB export module.

The Export to MATLAB module is responsible for converting the data retrieved by a Callisto Cell and generating MATLAB script that will produce a graph with the same data when executed.

The Export to .docx module is responsible for rendering an entire Jupyter Notebook document into .docx format, for use with Microsoft Word.

The Callisto Cell module is responsible for presenting a user friendly means to retrieve data from the MIMIC-III database, which is also installed on the server. This module adds additional user interface to Jupyter Notebook that allows the selection of presets, filters, and text required to retrieve and display data from the database graphically.

FIGURE 1: ENVIRONMENT MODEL.
2.7.2 CLASS DIAGRAM

This section describes the class model diagram in Figure 2. This model presents a logical view of the main elements that Callisto requires, and contains, and how they relate to each other. Each class member maps to a specific functional requirement detailed in section 3. Members for each class are briefly described below.

- **User:**
  The User class represents a user account, which is linked to a particular researcher or administrator. Users have the following members:

  - `username: string` The name the user is assigned. It is used to identify users, and submitted by a user during login.
  - `password: string` The password associated with a user account. It is used for autho-
rization during login. This should be salted and hashed for security.

- **login() : function** Submits a username and password in order to gain access to the system. If account credentials are incorrect, access is not given.

- **createDocument() : function** Creates a new empty document with a given name that the user can open in order to modify.

- **openDocument() : function** Opens a given document for editing by the user.

- **saveDocument() : function** Saves the changes made to a given document to disk.

- **export() : function** Exports a given document in a given format (.pdf, markdown, .tex, .docx).

- **Admin:**
  An Admin is a specialised user that is also able to create and manage user accounts. The Admin can also manage the presets by adding, editing or removing them. Admins have the following members:

  - **adduser() : function** Creates a new user account with a given username and password.
  
  - **changeUserPassword() : function** Changes the password for a given username to a new value.
  
  - **removeuser() : function** Removes the specified user account. The credentials of this account will no longer be able to access the system.
  
  - **addPreset() : function** Adds the specified preset to the set of available presets. Users will be able to use this preset to create graphs.
  
  - **editPreset() : function** Changes the contents of the specified preset.
  
  - **removePreset() : function** Removes the specified preset from the set of available presets. Users will no longer be able to use this preset in their documents.

- **Document:**
  The document class is used to represent the contents of a Jupyter document. Documents are named and contain cells of various types. Cells can be added, removed and changed within a document. Additional details of the Jupyter Notebook and Cell types are omitted here to maintain a focus on Callisto and its components.

  - **cells : Cell[]** An ordered set of cells that make up the document.
  
  - **name : string** The name of the document, used for identifying it.
  
  - **addCell() : function** Adds a new cell to the document.
  
  - **removeCell() : function** Removes the specified cell from the document.
• **Cell:**
The cell class is used to contain text and formatting for display. This class is provided by Jupyter.

  - `contents : string` The contents of a cell, represented as a string.

• **Code Cell:**
A code cell can be provided with Python code to be executed. Any output of this code is rendered in the cell upon execution.

• **Markdown Cell:**
A markdown cell allows the input and rendering of markdown text.

• **Callisto Cell:**
A Callisto cell is a specialised code cell that includes graphical user input for generating a graph. The internal code for this cell is hidden and not modifiable by a user. A Callisto cell allowed a user to select a preset, apply filters, and set the graph style before generating the graph. Data for the graph is acquired by creating and executing a query based on the selections.

  - `preset : Preset` The preset associated with this cell.
  - `filters : Filter[]` The set of filters associated with this cell.
  - `graphType : string` The type of graph to be rendered. This value may be one of the following: `bar2d` for bar charts, `line2d` for 2d line charts, `line3d` for 3d line charts, `pie2d` for pie charts, `scatter2d` for two dimensional scatter plots, or `corrAnal` for a correlation analysis plot.
  - `graphTitle : string` The title of the graph.
  - `ableToExport : boolean` A flag that determines whether this cell is exported (true) or not (false).
  - `notes : string` A description or additional information about the graph.
  - `setPreset : function` Sets the current preset to a specified preset.
  - `addFilter : function` Adds the specified filter to the cell.
  - `setGraphType : function` Sets the graph type for this cell.
  - `removeFilter : function` Removes the specified filter from this cell.
- **drawGraph**: function Produces a graph in the cell based on the current properties, and data from an executed query.

- **toggleExport**: function Sets the variable ableToExport with a specified boolean value.

- **analyseCorrelation**: function Computes correlation coefficients for a data from an executed query.

- **exportToMatlab**: function Generates a MATLAB script using data from an executed query and the properties of this cell.

- **Preset**:
  The preset class represents a predetermined combination of settings required to produce a particular type of graph. The class holds a preset name, and id, and labels for the axes of the graph that it produces. It also has a boolean parameter that determines whether the described graph is 2- or 3-dimensional.

  - **name**: string The name of this preset, used for displaying user interfaces.
  - **id**: int A unique identification number for this preset.
  - **xAxis**: string Describes the X axis of the graph to be produced.
  - **yAxis**: string Describes the Y axis of the graph to be produced.
  - **zAxis**: string Describes the Z axis of the graph to be produced. This is only used for three dimensional graphs.
  - **3DGraph**: boolean A flag that is true if this preset produces a 3D graph, or false if it describes a 2D graph.

- **Filter**:
  The filter class represents a means for the user to filter a preset further. The class has a name and type, such as number range, or discrete choices, and a value, or values, of the specified type. A filter represents an additional constraint on a data set described by a preset.

  - **name**: string The name of this filter, used for displaying user interfaces.
  - **type**: string Describes the format of value. Takes two values: range for numeric intervals, or select for multiple choice.
  - **value**: string A representation of the values that are permitted for this filter. For type range this is a minimum and maximum value, and initial values for the upper and lower boundaries of the range to be specified. For type select this is a set of values that may be selected from.
• **Graph:**
The graph class contains 2- or 3-dimensional data points, axis titles, a graph title, and styling information required to generate a customised graph. The graph class is then able to produce a rendered graph displaying the data described by the data points.

  - `dataPoints` An object containing data points in two or three dimensions, representing the result of an executed query.
  - `xAxisLabel : string` The text label for the X axis.
  - `yAxisLabel : string` The text label for the Y axis.
  - `zAxisLabel : string` The text label for the Z axis. This is only used for three dimensional graphs.
  - `title : string` The title to be rendered above the graph.
  - `lineColour : Color` The colour of the lines and markers of this graph.
  - `markerStyle : string` The style of marker to be displayed.
  - `description : string` A description of the graph.
  - `setLineColour : function` Sets the value of `lineColour` to a specified Color value.
  - `setMarkerStyle : function` Sets `markerStyle` to a specified style.

• **Query:**
The query class represents the collection of selected presets and filters in a form that can be executed on the MIMIC III database. This data is then used to produce an object of the graph class.

  - `columns : string[]` A set describing which columns will be selected by this query, and from which tables.
  - `conditions : string[]` A set describing conditions that selected rows must satisfy.
  - `execute : function` Executes this query on a specified database. Returns `dataPoints` that can be used for creating a graph.
2.7.3 DATA MODEL

The Callisto plug-in is highly dependent on the data provided by the MIMIC-III database. The database, its structure and contents are well documented [1]. Here we model the data required by The Callisto plug-in in order to access MIMIC-III. An entity relation diagram describing the data organisation is given in Figure 3. Brief explanations of entities of attributes are also given.

FIGURE 3: CALLISTO PLUG-IN DATA MODEL
• **User:**
  A user represents the account belonging to a user of Callisto. Researchers are treated as standard users and admins are a specialised type of user with additional permissions. Users have the following properties:

  - `user_id:integer` A number that uniquely identifies this user.
  - `username:string` A string unique to this user that identifies them for log in purposes.
  - `password:string` The associated password for this user, used for authorisation during log in. This value is hashed and salted for security.

• **Document:**
  A document represents a project that can be created, opened, modified and saved by users.

  - `filename:string` The filename of the document, used for storing its contents and identification.
  - `title:string` The title of the document.

• **Cell**
  A cell is a generalised section of a document. Cells may contain text, or code to be executed.

  - `cell_id:integer` A number that uniquely identifies the cell within the document.
  - `contents:string` A string representation of the contents of the cell.

• **CallistoCell**
  A Callisto Cell contains data required to select, configure and create a graph corresponding to data extracted from the MIMIC-III database.

  - `graphTitle:string` The title of the graph to be produced.
  - `graphDescription:string` Text written to describe the graph.
  - `MarkerColour:colour` A string representing the colour of marker to be drawn by the graph. Represented in hex notation.
  - `MarkerStyle:char` A character representing the style of marker to be drawn by the graph.

• **Preset**
  A preset is a predetermined representation of tables, columns and rows that define a graph that can be created.

  - `name:string` The name of the preset. This should uniquely identify the preset.
- **availableGraphTypes**: A set of string values representing graph types that can be created when using this preset. Graph types that can be made available are:
  - *line3d*: a three dimensional line plot.
  - *line2d*: a two dimensional line plot.
  - *scatter2d*: a two dimensional scatter graph.
  - *bar2d*: a two dimensional bar graph.
  - *pie2d*: a two dimensional pie chart.
- **baseQuery**: A string containing the base query for the preset.

- **Filter**
  A filter represents a constraint on data to be fetched by the query.
  - **name**: The name of the filter. This should uniquely identify the filter.
  - **type**: A string representing the type of the filter. This is either "select" for a choice from a discrete set, or "slider" for a pair of numeric values within a defined range.
  - **value**: The value or values associated with the type. For "select", this is a list of options to be selected from. For "slider" this is values corresponding to maximum and minimum values and default values.

- **Query**
  A query represents a completed query ready for execution on some database in order to produce a collection of data, for use in producing a graph.
  - **database**: An multi-value attribute representing the database on which this query should be executed, its location, and authorisation information.
  - **query**: A string representing the query to be executed on the specified database.
2.7.4  SEQUENCE DIAGRAMS

This section takes the use cases described in Appendix A of the URD [2] and produces sequence diagrams from them.

2.7.4.1  Changing the cell type to a Callisto cell with the multi-layer menu

In a Jupyter project there are cells and every cell has a type. A Callisto cell is a cell which has the type 'Callisto'. In this sequence diagram it is shown how a Callisto cell is created. First of all you have to have a project open in Jupyter. Then inside this project the user should select the cell which they want to change into a Callisto cell after which it is highlighted. After that the user should select 'cell' in the menu for the cell menu to appear. Finally, the user selects the option 'Callisto', after which the cell is changed into a Callisto cell.

Goals: To change the cell type to a Callisto cell.
Summary: Changing the cell type to a Callisto cell with the multi-layer menu.
Preconditions: There is an project open in Jupyter with a cell in it.
Priority: Must have.

![Sequence Diagram](image)

FIGURE 4: SEQUENCE DIAGRAM FOR CHANGING THE CELL TYPE TO A CALLISTO CELL WITH THE MULTI-LAYER MENU.

2.7.4.2  Changing the cell type to a Callisto cell with the drop-down cell type selector

This sequence diagram is quite similar to the previous sequence diagram. The difference is how the user changes the cell type. In the previous diagram the user used the top menu to select the cell type, while in this sequence diagram they will use the drop-down selector. The user first selects the cell, after which it is highlighted. then they click on the cell drop-down, which
will open the drop-down menu for cell types. Finally, the user selects 'Callisto cell' from the dropdown selector and the cell changes to a Callisto cell.

**Goals:** To change the cell type to a Callisto cell.
**Summary:** Changing the cell type to a Callisto cell with the multi-layer menu.
**Preconditions:** There is a project open in Jupyter with a cell in it.
**Priority:** Must have.

![Sequence Diagram](image)

**FIGURE 5:** SEQUENCE DIAGRAM FOR CHANGING THE CELL TYPE TO A CALLISTO CELL WITH THE DROPDOWN MENU.

### 2.7.4.3 Configuring a Callisto cell to get one of the preset graphs

After a user opens a Jupyter project and the user created a Callisto cell, it's time to talk about the sequence diagram of the actual creation of a graph. The user first clicks on the preset dropdown, this will show a list of all presets. The user then selects the preset that is desired and click on 'make graph'. The Callisto cell will then execute the query and try to draw the graph.

**Goals:** To get one of the preset graphs.
**Summary:** Configuring a Callisto cell to get one of the preset graphs.
**Preconditions:** The Callisto cell where the graph should be produced is already selected.
**Priority:** Must have.
2.7.4.4 Configuring the styling of a graph

The user is able to create graphs but these graphs should also look nice. Therefore, the user should be able to configure the styling of a graph. First they click on the styling dropdown which displays the drop-down list with graph style categories. After which they selects the styling category they want, which will return the options for that specific styling category. Finally the user sets the styling he wants for that specific category. The selected stylings are then displayed.

Goals: To change the styling of a graph.
Summary: The user changes the graph by means of changing colour, graph type or kind of marker.
Preconditions: The graph that the user wants to edit is already selected.
Priority: Should have.
2.7.4.5 Setting a new filter for a Callisto cell

The user might want to filter the data that will be displayed in their graph. Therefore they need to be able to add filters. They do this by clicking the '+ add filter' button. The cell will then display a new filter option in which the user can select the filter type. After that the parameters for the input are displayed and the user can input the filter parameters. This will result in the selected parameters being displayed.

Goals: To set up a new filter for the data of the MIMIC-III database.
Summary: Setting a new filter for a Callisto cell.
Preconditions: The Callisto cell already has a preset set.
Priority: Must have.
2.7.4.6 Deleting a present filter for a Callisto cell

The user might want to remove a filter for the graph if they no longer want the filter to be applied. To do this they click the ‘x’ symbol next to the filter. They can then look at the result by clicking ‘make graph’ again to have the graph redrawn.

Goals: To delete a filter for the data of the MIMIC-III database.
Summary: Deleting a filter for a Callisto cell.
Preconditions: The Callisto cell already has a preset and at least one filter set.
Priority: Must have.
2.7.4.7 Add a new query as a technical user

To add a query to the system, all the technical user has to do is add a new query configuration file. This new query can then be used by the Callisto plug-in.

Goals: Add a new query to the system.
Summary: The technical user adds a new query to the system enabling the clinical users to use it afterwards.
Preconditions: Jupyter Notebook including the Callisto plug-in is already installed.
Priority: Must have.

FIGURE 9: SEQUENCE DIAGRAM FOR DELETING A PRESENT FILTER.

FIGURE 10: SEQUENCE DIAGRAM FOR ADDING A NEW QUERY AS A TECHNICAL USER.
2.7.4.8  Change an existing query as a technical user

To change a query in the system, all the technical user has to do is edit an existing query configuration file. This edited query can then be used by the Callisto plug-in.

**Goals:** Change an existing query in the system.  
**Summary:** The technical user changes an existing query to the system enabling the clinical users to use the updated query afterwards.  
**Preconditions:** Jupyter Notebook including the Callisto plug-in is already installed.  
**Priority:** Must have.

**FIGURE 11:** SEQUENCE DIAGRAM FOR CHANGING A QUERY AS A TECHNICAL USER.

2.7.4.9  Exporting the document to word format

To export graphs and notes to word the user has to select download as .docx from the title menu. This will then send the content to the DOCX module which will export the document to word.

**Goals:** The document exported as word document.  
**Summary:** Exporting the document to word format.  
**Preconditions:** Jupyter has a project open with Callisto cells in it.  
**Priority:** Must have.
2.7.4.10 Selecting graphs for the word file

Since the user might not want to export all the graphs and notes the user can select those that they want to be exported when they export the file like described above. To do this they only have to check the checkbox near the graph if they want to include it. To exclude it they just have to uncheck the checkbox. After this the user can continue as described above to export the document.

**Goals:** To select which graphs with descriptions are going to be present in the exported word document.

**Summary:** Selecting which graphs with descriptions are going to be present in the word document when exported.

**Preconditions:** A Project is open in Jupyter.

**Priority:** Should have.
2.7.4.11 Exporting a graph to MATLAB

The user might also like to export the graphs they created to MATLAB. To do this they press the 'export to MATLAB' button near the graph they want to export. The Callisto cell will then create a MATLAB script and try to download it.

Goals: To export the data and contents of the graph to a MATLAB usable graph.
Summary: Exporting the graph's data and contents to a MATLAB graph.
Preconditions: A Callisto cell is already present with a graph to be exported to MATLAB. MATLAB is already installed.
Priority: Should have.
2.7.4.12 The admin adds a new user account

The admin will need to add user accounts sometimes. To do this he opens the terminal. This will make Jupyter Hub open a terminal. The admin can then create user accounts and assign passwords to each account. After he is done he closes the terminal.

Goals: To add a new user.
Summary: The admin adds a new user account.
Preconditions: The admin is logged in into Jupyter hub.
Priority: Must have.
2.7.4.13 The admin deletes a user account

The admin will need to delete user accounts sometimes as well. To do this he opens the terminal. This will make Jupyter Hub open a terminal. The admin can then delete a user account. The terminal will request a confirmation and if the user confirms, the account will be deleted. After he is done the admin can close the terminal.

Goals: To delete a user.
Summary: The admin deletes a user account.
Preconditions: The admin is logged in into Jupyter hub.
Priority: Must have.
2.7.4.14 The admin edits a user accounts password

The admin might need to change the passwords for user accounts sometimes. To do this he opens the terminal. This will make Jupyter Hub open a terminal. The admin can then request a password change. The terminal will then request a new password. Once the admin enters the new password the account is updated. After he is done the admin can close the terminal.

Goals: To edit a users password.
Summary: The admin edits a user accounts password.
Preconditions: The admin is logged in into Jupyter hub.
Priority: Must have.
FIGURE 17: SEQUENCE DIAGRAM FOR CHANGING A USER ACCOUNTS PASSWORD.
3 SPECIFIC REQUIREMENTS

Listed in this section are all the software requirements for The Callisto plug-in. All requirements are prioritised with the MoSCoW model [3]. The model works as follows:

**M** *Must have*: These requirements are essential and must therefore be included in the product.

**S** *Should have*: These requirements are important, but not as important as the must have requirements. Hence, they can be left out if there is no other choice.

**C** *Could have*: These requirements are good to have in the product, but only when there is time left or if it is easy to implement.

**W** *Won’t have*: These requirements will not be implemented into the project, but can be realised at a later version of the product.

3.1 FUNCTIONAL REQUIREMENTS

This example shows how the requirements generally look:

3.1.1 EXAMPLE

<table>
<thead>
<tr>
<th>SR - number</th>
<th>priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>the function or variable name</td>
<td>A description of the requirement.</td>
</tr>
<tr>
<td><strong>Input</strong>:</td>
<td>If it is a function, this states what the function takes as input.</td>
</tr>
<tr>
<td><strong>Precondition</strong>:</td>
<td>If it is a function, this states the preconditions for the function.</td>
</tr>
<tr>
<td><strong>Postcondition</strong>:</td>
<td>If it is a function, this states the postconditions for the function.</td>
</tr>
</tbody>
</table>

These requirements are based on the class diagram in section 2.7.2.

3.1.2 ADMIN

**SR - 1**

*must have*

```java
addUser()
```

A function that lets the admin add a new user. He needs to input the new username and a password.

**Input**: username, password

**Precondition**: Admin is logged in and the server is running.

**Postcondition**: User is created with specified username and password.

**SR - 2**

*must have*

```java
changeUserPassword()
```

A function that lets the admin change a user's password. He needs to input the new password.

**Input**: password

**Precondition**: User is logged in and the server is running.

**Postcondition**: User's password is changed to the specified password.
A function that lets the admin change the password of a user.

**Input:** username, password

**Precondition:** Admin is logged in and the server is running.

**Postcondition:** Password of specified user is changed to specified password.

---

### SR - 3

**must have**

removeUser()

A function that lets the admin remove a specified user from the database.

**Input:** username

**Precondition:** Admin is logged in and the server is running.

**Postcondition:** User with specified username is deleted from database.

---

### SR - 4

**must have**

addPreset()

A function that lets the admin add a preset to the database. This preset can then be used to create graphs.

**Input:** preset

**Precondition:** Admin is logged in and the server is running.

**Postcondition:** Specified preset is added to the database.

---

### SR - 5

**must have**

editPreset()

A function that allows the admin to edit an already existing preset in the database.

**Input:** preset

**Precondition:** Admin is logged in and the server is running.

**Postcondition:** Specified preset can be edited in the database.

---

### SR - 6

**must have**

removePreset()

A function that lets the admin remove a preset from the database.

**Input:** preset

**Precondition:** Admin is logged in and the server is running.

**Postcondition:** Specified preset is removed from the database.

---

### 3.1.3 USER

**SR - 7**

**must have**

**username:** string

A string containing the user’s username. This is used by the admin to add/change/delete user profiles and by the user itself to login.

---

**SR - 8**

**must have**

**password:** string
A string containing the user’s password. This is used by the admin to add/change/delete user profiles and by the user to login.

**SR - 9**  
*must have*  
**logIn()**  
A function that lets the user log in to the server if they have a valid combination of username and password.  
**Input:** username, password  
**Precondition:** The server is running and the user is on the login screen.  
**Postcondition:** If the username and password are a valid match in the database the user is logged in.

**SR - 10**  
*must have*  
**createDocument()**  
A function that lets the user create a Jupyter document.  
**Input:** document name  
**Precondition:** The server is running and the user is in the main menu.  
**Postcondition:** A new document is created with the specified name, and can be opened.

**SR - 11**  
*must have*  
**openDocument()**  
A function that lets the user open a Jupyter document.  
**Input:** document  
**Precondition:** The server is running, the document is a valid document, and the user is in the main menu.  
**Postcondition:** Specified document is opened.

**SR - 12**  
*must have*  
**saveDocument()**  
A function that lets the user save the document.  
**Input:** document  
**Precondition:** The server is running, the document is a valid document, and the user is in the main menu.  
**Postcondition:** Specified document is saved.

**SR - 13**  
*must have*  
**export()**  
A function that lets the user export the Jupyter document to Word.  
**Input:** document  
**Precondition:** The server is running, the document is a valid document, and the user is in the main menu.  
**Postcondition:** Specified document is exported.
3.1.4 PRESET

SR - 14
name: string
A string containing the name of the preset. This is what is displayed to the user

SR - 15
id: integer
An integer representing the id of the preset. This is used to retrieve the correct preset

SR - 16
xAxis: string
A string containing the x-axis label to be used for the graph.

SR - 17
yAxis: string
A string containing the y-axis label to be used for the graph.

SR - 18
zAxis: string
A string containing the z-axis label to be used for the graph if there is a third dimension.

SR - 19
3DGraph: boolean
A boolean that is true if the preset is for a 3D graph and false if it is for a 2D graph.

3.1.5 DOCUMENT

SR - 20
cells: Cell[]
An array containing all the cells of the Jupyter document.

SR - 21
name: string
A string containing the name of the document.

SR - 22
addCell()
A function that adds a cell to the document.
**Precondition:** The server is running.
**Postcondition:** A cell is added to the document.

SR - 23
removeCell()
A function that lets the user remove a specified cell from the document.

**Input:** Cell

**Precondition:** The server is running.

**Postcondition:** The specified cell is removed.

### 3.1.6 CALLISTOCALL

**SR - 24**

*must have*

**preset:** Preset

A preset for the graph. This can be selected from a list by the user.

**SR - 25**

*must have*

**filters:** Filter[]

An array of filters that filter the data that is used in the graph.

**SR - 26**

*must have*

**graphType:** string

A string containing the graph type.

**SR - 27**

*must have*

**graphTitle:** string

A string containing the graph title.

**SR - 28**

*must have*

**ableToExport:** boolean

A boolean that is true if this cell should be exported and false if it shouldn’t be exported.

**SR - 29**

*must have*

**notes:** string

A string containing the notes under a graph. Users can use this to e.g. write a piece of text beneath their graph to further explain it or discuss what the graph shows.

**SR - 30**

*must have*

**setPreset()**

A function that sets the preset that is used for the graph.

**Input:** preset

**Precondition:** The server is running and the preset is a valid preset.

**Postcondition:** The specified preset is set.

**SR - 31**

*must have*

**addFilter()**

A function that adds a filter to the graph.

**Input:** Filter, filters

**Precondition:** The server is running and the filter is a valid filter.

**Postcondition:** The specified filter is added to the filters array.
SR - 32  
must have  
setGraphType()  
A function that sets the graph type.  
Input: graphType  
Precondition: The server is running and the graphType is a valid graph type.  
Postcondition: The graph is set to the specified graphType.

SR - 33  
must have  
removeFilter()  
A function that removes a filter from the graph.  
Input: Filter, filters  
Precondition: The server is running and the filter is a valid filter.  
Postcondition: The specified filter is removed from the filters array.

SR - 34  
must have  
drawGraph()  
A function that draws the graph. After changes to the graph this can be executed again to update the graph.  
Precondition: The server is running and the query is valid and executed.  
Postcondition: The graph is drawn.

SR - 35  
must have  
toggleExport()  
A function that changes the ableToExport boolean to change whether the accompanying graph will be exported or not.  
Input: ableToExport  
Precondition: The server is running  
Postcondition: The boolean of whether the Callisto cell is marked to export will change.

SR - 36  
must have  
analyseCorrelation()  
A function that performs a correlation analysis on the selected data.  
Precondition: The server is running and the query is valid and executed.  
Postcondition: The correlation between the datapoints is analysed.

SR - 37  
should have  
exportToMatlab()  
A function that exports the graph and notes to MATLAB script.  
Input: Graph, notes  
Precondition: The server is running  
Postcondition: The graph and notes are exported to MATLAB script.
3.1.7 CELL

SR - 38 must have
cell:

```plaintext
contents: string
A string containing the contents of a cell.
```

3.1.8 FILTER

SR - 39 must have
name:

```plaintext
name: string
The name of the filter displayed in the GUI.
```

SR - 40 must have
type:

```plaintext
type: string
The type of data that the filter requires. Either single value or a range.
```

SR - 41 must have
value:

```plaintext
value: string
The value of the data required by the filter.
```

3.1.9 GRAPH

SR - 42 must have
dataPoints[ ]:

```plaintext
dataPoints[ ]: array
An array of 2- or 3-dimensional data points.
```

SR - 43 must have
xAxisTitle:

```plaintext
xAxisTitle: string
The title of the x axis to be displayed with the graph.
```

SR - 44 must have
yAxisTitle:

```plaintext
yAxisTitle: string
The title of the y axis to be displayed with the graph.
```

SR - 45 must have
zAxisTitle:

```plaintext
zAxisTitle: string
The title of the z axis to be displayed with the graph (for 3-dimensional graphs only).
```

SR - 46 must have
title:

```plaintext
title: string
The title of the graph that will be displayed with the graph.
```

SR - 47 must have
lineColour:

```plaintext
lineColour: string
A string representing the colour of the lines and markers to draw for the graph.
```
**SR - 48**  
*markerStyle*: string  
A string representing the marker style for the points of the graph. The options are:

- Circle
- Square
- Diamond
- Cross
- X
- Open circle
- Open square
- Open diamond

**SR - 49**  
*description*: string  
A string containing a small description for a graph to further elaborate on the graph in a few words.

**SR - 50**  
*setLineColour()*  
A function that sets line colour for the line in the graph.  
*Input*: lineColour  
*Precondition*: The server is running.  
*Postcondition*: The colour of the line is changed to the specified colour.

**SR - 51**  
*setMarkerStyle()*  
A function that sets marker styles for the points in the graph.  
*Input*: markerStyle  
*Precondition*: The server is running  
*Postcondition*: The style of markers for the data is changed to specified style.

**3.1.10 QUERY**

**SR - 52**  
*columns*: string[]  
Columns to be extracted.

**SR - 53**  
*condition*: condition  
Conditions that rows in the database must satisfy in order to be included in the result.
SR - 54

gotoexecute()

Executes the provided query and returns the resulting rows.

Input: Query

Precondition: The server is running and the query is valid

Postcondition: The specified query is executed.
## 4 REQUIREMENTS TRACEABILITY MATRIX

### 4.1 URD TO SRD

<table>
<thead>
<tr>
<th>URD</th>
<th>SRD</th>
</tr>
</thead>
<tbody>
<tr>
<td>URF - 1</td>
<td>SR - 20, 22, 34, 42, 43, 44, 45, 47, 48</td>
</tr>
<tr>
<td>URF - 2</td>
<td>SR - 34, 52, 53, 54</td>
</tr>
<tr>
<td>URF - 3</td>
<td>SR - 20, 23</td>
</tr>
<tr>
<td>URF - 4</td>
<td>SR - 27, 46</td>
</tr>
<tr>
<td>URF - 5</td>
<td>SR - 27, 46</td>
</tr>
<tr>
<td>URF - 6</td>
<td>SR - 49</td>
</tr>
<tr>
<td>URF - 7</td>
<td>SR - 49</td>
</tr>
<tr>
<td>URF - 8</td>
<td>SR - 55</td>
</tr>
<tr>
<td>URF - 9</td>
<td>SR - 55</td>
</tr>
<tr>
<td>URF - 10</td>
<td>SR - 55</td>
</tr>
<tr>
<td>URF - 11</td>
<td>SR - 10, 11, 12, 21</td>
</tr>
<tr>
<td>URF - 12</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 13</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 14</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 15</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 16</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 17</td>
<td>SR - 26, 32, 34</td>
</tr>
<tr>
<td>URF - 18</td>
<td>SR - 50</td>
</tr>
<tr>
<td>URF - 19</td>
<td>SR - 51</td>
</tr>
<tr>
<td>URF - 20</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 21</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 22</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 23</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 24</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 25</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 26</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>URF - 27</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 28</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 29</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 30</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 31</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 32</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 33</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 34</td>
<td>SR - 14, 15, 16, 17, 18, 19, 24, 30</td>
</tr>
<tr>
<td>URF - 35</td>
<td>SR - 25, 31, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 36</td>
<td>SR - 25, 31, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 37</td>
<td>SR - 25, 31, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 38</td>
<td>SR - 25, 31, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 39</td>
<td>SR - 25, 31, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 40</td>
<td>SR - 25, 31, 33, 39, 40, 41</td>
</tr>
<tr>
<td>URF - 41</td>
<td>SR - 39, 40, 41</td>
</tr>
<tr>
<td>URF - 42</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 43</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 44</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 45</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 46</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 47</td>
<td>SR - 36</td>
</tr>
<tr>
<td>URF - 48</td>
<td>SR - 7, 8, 9</td>
</tr>
<tr>
<td>URF - 49</td>
<td>SR - 1, 7, 8</td>
</tr>
<tr>
<td>URF - 50</td>
<td>SR - 5, 7, 8</td>
</tr>
<tr>
<td>URF - 51</td>
<td>SR - 1, 7, 8</td>
</tr>
<tr>
<td>URF - 52</td>
<td>SR - 2, 7, 8</td>
</tr>
<tr>
<td>URF - 53</td>
<td>Done by Jupyter Hub</td>
</tr>
<tr>
<td>URF - 54</td>
<td>SR - 4</td>
</tr>
<tr>
<td>URF - 55</td>
<td>SR - 5</td>
</tr>
<tr>
<td>URF - 56</td>
<td>SR - 6</td>
</tr>
<tr>
<td>URF - 57</td>
<td>SR - 13</td>
</tr>
<tr>
<td>URF - 58</td>
<td>SR - 13, 28, 35</td>
</tr>
<tr>
<td>URF - 59</td>
<td>SR - 38</td>
</tr>
<tr>
<td>URF - 60</td>
<td>SR - 37</td>
</tr>
<tr>
<td>URF - 61</td>
<td>SR - 37</td>
</tr>
<tr>
<td>URF - 62</td>
<td>SR - 37</td>
</tr>
<tr>
<td>URF - 63</td>
<td>SR - 37</td>
</tr>
<tr>
<td>URF - 64</td>
<td>SR - 37</td>
</tr>
</tbody>
</table>
### 4.2 SRD TO URD

<table>
<thead>
<tr>
<th>SRD</th>
<th>URD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR - 1</td>
<td>URF - 49, 51</td>
</tr>
<tr>
<td>SR - 2</td>
<td>URF - 52</td>
</tr>
<tr>
<td>SR - 3</td>
<td>URF - 50</td>
</tr>
<tr>
<td>SR - 4</td>
<td>URF - 54</td>
</tr>
<tr>
<td>SR - 5</td>
<td>URF - 55</td>
</tr>
<tr>
<td>SR - 6</td>
<td>URF - 56</td>
</tr>
<tr>
<td>SR - 7</td>
<td>URF - 48, 49, 50, 51, 52</td>
</tr>
<tr>
<td>SR - 8</td>
<td>URF - 48, 49, 50, 51, 52</td>
</tr>
<tr>
<td>SR - 9</td>
<td>URF - 48</td>
</tr>
<tr>
<td>SR - 10</td>
<td>URF - 11</td>
</tr>
<tr>
<td>SR - 11</td>
<td>URF - 11</td>
</tr>
<tr>
<td>SR - 12</td>
<td>URF - 11</td>
</tr>
<tr>
<td>SR - 13</td>
<td>URF - 57, 58</td>
</tr>
<tr>
<td>SR - 14</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 15</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 16</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 17</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 18</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 19</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 20</td>
<td>URF - 1, 3</td>
</tr>
<tr>
<td>SR - 21</td>
<td>URF - 11</td>
</tr>
<tr>
<td>SR - 22</td>
<td>URF - 1</td>
</tr>
<tr>
<td>SR - 23</td>
<td>URF - 3</td>
</tr>
<tr>
<td>SR - 24</td>
<td>URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>SR - 25</td>
<td>URF - 35, 36, 37, 38, 39, 40</td>
</tr>
<tr>
<td>SR - 26</td>
<td>URF - 12, 13, 14, 15, 16, 17</td>
</tr>
<tr>
<td>SR - 27</td>
<td>URF - 4, 5</td>
</tr>
</tbody>
</table>
SR - 28 | URF - 58
SR - 29 | URF - 8, 9, 10
SR - 30 | URF - 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34
SR - 31 | URF - 35, 36, 37, 38, 39, 40
SR - 32 | URF - 12, 13, 14, 15, 16, 17
SR - 33 | URF - 40
SR - 34 | URF - 1, 2, 12, 13, 14, 15, 16, 17
SR - 35 | URF - 58
SR - 36 | URF - 42, 43, 44, 45, 46, 47
SR - 37 | URF - 60, 61, 62, 63, 64
SR - 38 | URF - 59
SR - 39 | URF - 35, 36, 37, 38, 39, 40, 41
SR - 40 | URF - 35, 36, 37, 38, 39, 40, 41
SR - 41 | URF - 35, 36, 37, 38, 39, 40, 41
SR - 42 | URF - 1
SR - 43 | URF - 1
SR - 44 | URF - 1
SR - 45 | URF - 1
SR - 46 | URF - 4, 5
SR - 47 | URF - 1
SR - 48 | URF - 1
SR - 49 | URF - 6, 7
SR - 50 | URF - 18
SR - 51 | URF - 19
SR - 52 | URF - 2
SR - 53 | URF - 2
SR - 54 | URF - 2
APPENDIX - USER INTERFACE MOCKUPS

In appendix A we will describe informal requirements for the graphical user interface that will allow users to select presets and filters, and to generate graphs. First we describe the user interfaces, and then the transitions between them.

A.1 LOG IN SCREEN

To use anything Callisto related the user first has to log in to JupyterHub on the server. The user has to fill in a valid username and password as seen in Figure 18.

![Figure 18: Log In Screen](image)

If the combination of the username and password is invalid, an error message will appear stating "Invalid username or password". This can be seen in Figure 19.

![Figure 19: Failed Login](image)

A.2 CALLISTO CELL

Here we present a series of images depicting a mockup of the user interfaces presented to a user within a Callisto cell. Figures 20 to 24 show the general sequence of actions required by a user in order to produce a graph.

**Step 0**, creating a Callisto cell, has to be done before anything can be done with creating the graphs. To add a cell click the ‘+’ icon in the top left. To change the cell into a Callisto cell, click
on the bar on the top of the page that says ‘Code’ and set it to ‘Callisto’ using the dropdown menu. This is shown in Figure 20.

![Figure 20: STEP 0: CREATING A CALLISTO CELL.](image)

**Step 1** shows the selection of a preset. This can be done by clicking the bar below ‘preset’ This list includes all 2D and 3D presets which are specified in the URD. Furthermore the user is able to select whether they want to export that particular graph or not, this can be done via the ‘Export’ checkmark.

![Figure 21: STEP 1: SELECTING A PRESET.](image)

**Step 2** shows the configuration of the filters. To add a filter, just click ‘+ Add filter’. The GUI supports 5 types of filters: Age, Gender, Ethnicity, Diagnosis and Treatment. The GUI makes sure that every type of filter can only be set once. Once the user selected one of the filter types, the corresponding options for that filter appear. All of the filter types have a list of options, except for age. For the age type the user has to specify an age range. The filter types Ethnicity, Diagnosis and Treatment have a long list of options attached to it. For these cases the GUI allows the user to search that list.
STEP 2: ADDING AND CONFIGURING FILTERS.

Step 3 shows the configuration of the graph styling. The title and description for the graph can be set by typing something in the field behind 'Title' or 'Description' respectively. Furthermore, the graph type can be set. The following graph types can be chosen: Line chart, Scatter plot, Bar chart, Pie chart, 3D Filled line and Correlation analysis. It depends on the kind of preset, which graph types can be selected. To change the type just click on the box next to 'Graph type' and select the correct type from the dropdown list. The colour of the line/markers can be selected by clicking the bar behind 'Color' and selecting the colour by picking from the colour roster as shown in Figure 23.

Lastly, in Step 4, the graph will be generated. This can be done by clicking the 'Make graph' button. Whenever the user changes something about the graph, they reload it by clicking this button again. Once the graph is created, the user is able to interact with the graph, this includes...
zooming and moving the graph. The user is able to reset the graph back to its zoomed out state.

**FIGURE 24:** STEP 4: GENERATE AND INTERACT WITH THE GRAPH.

### A.3 DIFFERENT TYPES OF GRAPHS

Other types of graphs like 3D graphs can be generated as well. An example of this is shown in Figure 25.
The same goes for Bar charts, shown in Figure 26.

As well as pie charts, shown in Figure 27.
And scatter plots, shown in Figure 28.

The user can also perform correlation analysis on some datasets. An example of such an analysis is how in Figure 29.
A.4 EXPORT TO WORD

If the user is satisfied with the graph and wants to use it in a word document, they can save the graphs as a word document. They can do this by going to file > Download as > Word Doc (.docx). This is shown in Figure 30.

FIGURE 30: EXPORT TO WORD.
B TRANSITIONS

In this appendix we describe the actions available to users between the user interfaces described above. The behaviour of interfaces and their actions are modelled using Petri-nets. Interfaces are modelled as states, and actions as transitions. Since the Callisto service as a whole is hosted on a server and accessed via the users web browser, the user may terminate their connection with the server at any time. Thus from all places in the Petri-nets it is possible for the user to close their connection. This is handled in different ways depending on the context.

Figure 31 shows transitions related to logging into the system, managing users, and opening notebook documents. Upon accessing the system for the first time, users are presented with a login screen (see Figure 18). Users with valid credentials (username and password) are able to submit their credentials for authorisation. If invalid credentials are provided, the "login fail" transition is fired and users are prompted to try again. If the credentials are valid, the "log in success" transition fires, and the main menu is shown.

From the main menu, normal users have the option to open a document, and proceed to the notebook view. For new documents, this process also includes creation. Administrators also have the option to access the user management screen from the main menu. This transition is not available to normal access level users.

From the notebook view users can perform document actions via the "document actions" transition, or close the document via "close document". Upon closing the document they will return to the main menu. Document actions are actions that a user may perform upon the an open notebook document such as adding, removing or editing cells, saving, and exporting. Callisto cells are created this way.

Figure 32 shows a detailed view of the workings of a Callisto cell. Generic cells are created from the notebook view, and can be set to be Callisto cells. Callisto cells start at "new cell", where the cell contains no information. The user may select a preset via the "select preset" transition, and then the cell makes available all the main options, and appears in the "Callisto cell" place. This place corresponds with the default view of a Callisto cell, with filters, visualisation and preset information visible. From this place, the user can change the selected preset, draw the graph, or modify the current filters. Filters can be added via "add filter" which will add filters to the "filters" place. "Remove filters" can remove filters only if tokens for them exist in "filters". Similarly, editing filters requires that a filter exists, otherwise the transition cannot be fired. Not modelled here are the different types of filters, the maximum number of filters permitted, or the restriction on having maximally one filter of each type. This simplification is to constrain the transitions diagram to broadly demonstrate the transitions between the user interface views.
FIGURE 31: NOTEBOOK VIEW TRANSITION DIAGRAM.

FIGURE 32: CALLISTO CELL TRANSITION DIAGRAM.