Project Specification

for the

SIUE Astronomical Observatory

by

Jaime Acevedo

Matthew Bunch

Ryan Sharp

Of

Team Observatory

Revision 1.0

As of: 17 September 2014

Change Log:

Revision	**Change Note(s)**
1.0	• Initial release

Reviewed and Approved By:

Name	Signature	Date
	_____	_____
	_____	_____
	_____	_____

Table of Contents

FIGURES

TABLES

1. INTRODUCTION

The following sections will provide focus to the project. It will help answer key questions about the purpose of the proposed software. In the following weeks it will serve to help formulate plans and stay on target.

1.1. Business and Domain Description

The Department of Physics at Southern Illinois University Edwardsville has recently added Astronomy as a new concentration to physics majors. Astronomy focuses heavily on the study of celestial objects, their physics and chemistry, and calculations of their positions and motions. Moreover, astronomy studies the origin and evolution of the universe. In order to effectively perform these studies and research, an observatory is available on campus. The observatory allows for close examination of objects based on the input of parameters. Currently the observatory is not in use by students and automation of the telescope is still under development. The client is part of the physics department faculty. He is responsible for overseeing the automation of the observatory and its future availability to students of the major. A major focus of the observatory will be to further the client's eclipsing binary and multiple star system research, and also expand undergraduate student astronomical research.

1.2. Concept of Proposed System / Scope

The current observatory system is still under development. The client has been working on automation of the telescope for the past two years. The software that is currently present is responsible for the functioning of the telescope itself. As of now, there is not a way of allowing wide use of the observatory by students and faculty. The client has a program he runs on his server to interact with the telescope to perform operations. As this is still under development, functionality is limited.

The observatory is envisioned to serve as a tool for astronomy laboratories required within the physics coursework. Undergraduate and Graduate students will make use of it to complete these labs. Another avenue the observatory will provide is the potential for research to be conducted by graduate students and faculty.

The proposed scheduling software will allow users to submit a proposal, including viewing times and observational parameters (astronomical parameters such as the right ascension, declination, and epoch, etc. of celestial objects), in order to use the observatory. A user-friendly web-site will be used to submit such observation proposals. The scheduling aspect of the proposed software will actually allocate times for users to access the telescope remotely. An administrator role will be designated to the client within the software to allow him to view and

manage pending observation proposals of his students. This management system will also be another user-friendly interface separate from the scheduling interface. Since observations made by users will be in the form of photographs, the proposed scheduling software will also provide the capability to send the images (through email) they observed during their session with the observatory. It is important to note that the proposed scheduling software will not actually be taking the photographs during observing sessions, nor will it control functions of the telescope. This is all done by the automated software that runs the telescope in the observatory. The proposed system is responsible for collecting the observation parameters from users and properly formatting it to be used by the automation software. Distribution of photographs taken by the telescope to the appropriate user is the second purpose of the proposed software.

1.2.1. Operational Concept and Scenario

a) **Operational policies and constraints:**
   The proposed software will consist of logging in through the physics lab iMacs to schedule time with the observatory; the observatory is used remotely with limited access; the automation system will dictate available times
b) **Description of the proposed system:**
   Please see the above section
c) **Modes of system operation:**
   The proposed scheduling software will have a user mode and an administrator mode. The user mode will be the scheduling interface for students and the client. The admin mode will let the client see the current schedule and make changes accordingly.
d) **User classes and involved personnel:**
   There will be users and there will be an administrator. Students are the users. The client is a user as well as an administrator. As of now, the client is the only personnel involved with the observatory and he holds interest in the proposed scheduling software.
e) **Support environment:**
   Physics laboratories will host the proposed scheduling software. It will run on the iMacs provided within a web browser. Storage for observations is located on the servers the client has setup with the observatory.
f) **Operational Scenarios:**
   See 3.2. Functions and 3.2.1. Agile Use Cases

1.3. Product Overview

1.3.1. Product Perspective

The observatory scheduling software will help complete the scheduling process of Southern Illinois University Edwardsville's Astronomical Observatory. The observatory scheduling software will be an element aside from the observatory automation system. The automation software that is currently present is responsible for managing and operating the telescope within the observatory. The current software requires a file with parameters to be inputted into the observatory's automation software. The interfaces between the observatory scheduling software and current automation software consist of the client manually transferring the file outputted by the scheduling software and to the input of the automation software. This is at the client's request. The interfaces between the automation software and proposed scheduling software are not applicable at this time.

Below is a diagram of the proposed process:

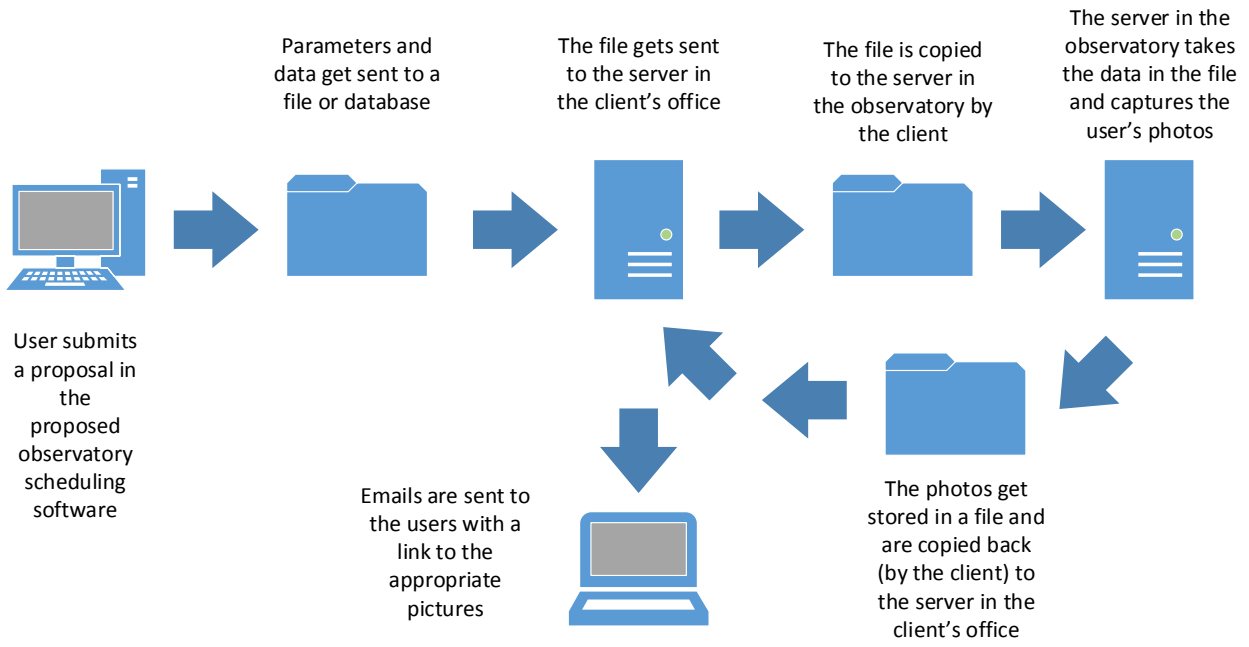

Figure 1: The Proposed Process with the Scheduling Software

1.3.2. Product Functions

The observatory scheduling software will perform three important functions to complete automation and allow wider student use of the observatory.

(1.) The information-gathering interface will allow observational parameters to be entered from user accounts. The parameters will consist of information specifying desired positioning of the observatory's telescope, the name of the object that will be observed, time and dates of observations, and the duration of these observations.
(2.) The software will determine schedules for use of the observatory from the parameters collected. These can be modified and will assist the administrator in prioritizing use among individuals.
(3.) Another important function of the software will be the flagging and distribution of data. This process will help determine what was observed during observation times and keep records of such. Records will consist mainly of photographs, and these will be distributed to the proper user. Preferred distribution will be through emailing a download link that hosts the captured photographs.

1.3.3. User Characteristics

Undergraduate Students – Users

a) **Function:**
Will use the observatory scheduling software to reserve time with the telescope to complete course laboratories that rely on its use
b) **Location:**
Will have access to the observatory scheduling software inside the astronomy/physics laboratories
c) **Type of device:**
Will use iMac desktop computers, provided in the astronomy/physics laboratories, to use the observatory scheduling software
d) **Number in each group:**
No more than 100 students in a given semester
e) **Nature of their use of the software:**
Will use the observatory scheduling software to reserve time to use the telescope located in the observatory, and to gather photographs captured by the telescope; will be used for academic purposes and possible research opportunities

Graduate Students – Users

a) **Function:**

Will use the observatory scheduling software to reserve time with the telescope to complete course laboratories that rely on its use

b) Location:
Will have access to the observatory scheduling software inside the astronomy/physics laboratories

c) Type of device:
Will use iMac desktop computers, provided in the astronomy/physics laboratories, to use the observatory scheduling software

d) Number in each group:
No more than 15 graduate students in a given semester

e) Nature of their use of the software:
Will use the observatory scheduling software to reserve time to use the telescope located in the observatory, and to gather photographs captured by the telescope; will be used for academic purposes and research

Client, Dr. Sabby – User, Operator

a) Function:
Will use the observatory scheduling software to schedule his students some time with the telescope for laboratories; will use the observatory scheduling software to perform admin capabilities; will use the observatory scheduling software to reserve time with the telescope in order to further his research

b) Location:
Will have access to the observatory scheduling software inside the astronomy/physics laboratories, as well as his personal computer located in office or home

c) Type of device:
Will use iMac desktop computers, provided in the astronomy/physics laboratories, to use the observatory scheduling software; may also use his own personal computer

d) Number in each group:
1 individual – Dr. Sabby

e) Nature of their use of the software:
Will use the observatory scheduling software to reserve time to use the telescope located in the observatory, and to gather photographs captured by the telescope; will be used for academic purposes and research; will also use the software to administrate student use

1.3.4. Limitations

There will not be a direct line from the proposed observatory scheduling software to the server that hosts the telescope automation software. The setup, proposed by the client, is to take observational parameters from the observatory scheduling interface and then write the data to a file or database. From there, the client would like to upload the contents of the file (or database) to a server located in his office. He would then proceed to send the data from the server in his office to the server located in the observatory. This process is arranged as such in an effort to intentionally provide limited access to the telescope automation software. The manual process that exists prevents the proposed software from completing full automation of the observatory. This also means that the client will have to manually transfer photographs captured from the observatory's server back to his office server in order to complete the emailing aspect of the proposed software.

1.4. Definitions

This section contains terms and their definitions that are specific to the stakeholders' business processes and product domain. The following terms and definitions are intended to explain some of the dialog and technical meanings of the terms associated with the observatory scheduling software:

celestial body – an object beyond the earth's atmosphere that is being observed by the telescope
declination – a main observational parameter to be specified by the user in the observatory scheduling software; it is to be specified in the format "+hours:minutes:seconds" or "-hours:minutes:seconds", where each piece is a numerical value
epoch – a main observational parameter to be specified by the user in the observatory scheduling software; it is to be specified in the format "n", where n is a sequence of numbers referencing a period in time
laboratories – refers to the physics labs students use in the physics curriculum
meridian – one of the imaginary lines joining the north and south poles at right angles to the equator, designated by degrees of longitude from 0° at Greenwich, England to 180°; used as a reference to determine Universal Time (see below)
object name – the name of the celestial body being observed through the telescope
observatory – refers to the location of the telescope located on Southern Illinois University Edwardsville's campus
observatory proposal – the form that a student submits within the observatory scheduling interface to ask for permission to use the telescope at a specific time

right ascension – a main observational parameter to be specified by the user in the observatory scheduling software; it is to be specified in the format "hours:minutes:seconds", where each piece is a numerical value
stakeholder – person or organization interested in the successful outcome of the observatory scheduling software
telescope – optical instrument used to view celestial bodies
Universal Time (UT) – the mean time for the meridian at Greenwich, England; used as a reference for scheduling specific dates and times
user – individual that will use the observatory scheduling software
UT Date – the date based on Universal Time (see above); it is an observational parameter to be specified by the user in the observatory scheduling software; it is to be specified in the format "mm/dd/yyyy"
UT Start Time – a user-specified time based on Universal Time (see Universal Time above for definition); it is an observational parameter to be specified by the user in the observatory scheduling software; it is to be specified in the format "hour:month (hh:mm)"

1.5. Stakeholders

Included below is a list of all people and organizations interested in the successful outcome of the observatory scheduling software:

Organizations:
Department of Physics, Southern Illinois University Edwardsville

- The department will use the observatory scheduling software in their astrology curriculum. More specifically, the astrology laboratories, associated with astrology courses, will utilize the software to allow students to schedule time with the *observatory.*

Department of Computer Science, Southern Illinois University Edwardsville

- The department requires a senior project for each computer science student before he or she graduates. The department is interested in seeing a successful process of semi-agile software engineering. The final product is produced from the process.

Individuals:
Dr. Jeffrey Sabby

- The individual will use the observatory scheduling software to further his eclipsing binary and multiple star system research. He will also use the software to facilitate student usage of the observatory in his course labs.

Dr. Dennis Bouvier

- This individual is serving as Oversight; Oversight guides the development process and has the final say over all aspects of the project; the individual will evaluate the success of the development process and the final project

Jaime Acevedo

- This individual is a student enrolled in senior project. He is one of the three individuals assigned to plan and develop the observatory scheduler software. He is responsible for delivering the developed product.

Matthew Bunch

- This individual is a student enrolled in senior project. He is one of the three individuals assigned to plan and develop the observatory scheduler software. He is responsible for delivering the developed product.

Ryan Sharp

- This individual is a student enrolled in senior project. He is one of the three individuals assigned to plan and develop the observatory scheduler software. He is responsible for delivering the developed product.

2. REFERENCES

2.1. Bibliography

1. Systems and software engineering — Life cycle processes — Requirements engineering. (2011). *ISO/IEC/ IEEE*, [online] 1(29148), p.94. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6146379 [Accessed 13 Sep. 2014].
2. Miles, R. and Hamilton, K. Learning UML 2.0. Sebastopol, CA: O'Reilly Media, Inc., 2006.

3. SPECIFIC REQUIREMENTS

3.1. External Requirements

The product will provide the functionality for the user to provide specific input to an end system and that input will be stored into a database for the telescope to eventually process. In order to provide this functionality, the product will need to provide various interfaces for various functions. The interfaces are described in this section.

Home Screen Interface

a) **Name of Item:** Home
b) **Description:** The Home Screen interface will be an interactive form that will be manually filled out by a user. The form will contain sections to fill out the necessary parameters that get sent out to the telescope end system for processing. The user will be able to log in into a home screen which will allow the user to pick a date, time, specific coordinates, objects, and certain amount of pictures. The primary objective of this interface is to provide accurate information to be sent out to the telescope end system. The home screen interface will be designed to be friend user oriented. The Home Screen will also provide links to take the user to other external interfaces.
c) **Source of Input or Destination of Output:** The source of this input will come directly from the user. The interface will take input from a keyboard. The inputs destination will be to the applications database.
d) **Valid range, accuracy. And/or tolerance:** The date and time fields will need to be appropriate future dates that reflect real-life time. The interface will be equipped to validate these dates and times. The interface will also need to check for conflicting reservations. If a reservation is conflicting it will provide essential feedback to the user as soon as possible.
e) **Units of Measure:** The form will need to take certain units of measurement for the telescope to process. Right Ascension (RA), Declination (DEC), and Epoch are units that will need to be provided by the user on this interface.
f) **Timing:** The interface should be able to schedule reservations at any given time.
g) **Relationships to other inputs/outputs:** This interface will have a frequently active connection to the applications database by sending user input and storing it to specific tables. As mentioned earlier the interface will provide links to other external interfaces.
h) **Screen formats/organization:** The screen format will be oriented to be user friendly and be expandable to take up the entire screen of the computer.

i) **Window formats:** The format of the window will depend on the certain browser that the user will be using. Specifically we will want the application to run on a Safari browser.
j) **Data formats:** Data will be entered and process to be formatted and finally entered into the database. Data will need to be appropriate to the certain field. If the information in the field is in incorrect format the application must reject the input.
k) **Command formats:** N/A
l) **End messages:** A successful scheduled reservation will notify the user in the following way:
    a. Notify user via E-mail of scheduled reservation providing the information that the user entered.
    b. A feedback message will "pop-up" and let the user know that an e-mail was sent regarding information.

User Reservation Queue Interface

a) **Name of Item:** User Reservation Queue
b) **Description:** The Reservation Queue will be an interactive GUI that will be visible to every user. The GUI will contain the user's previously scheduled reservations and the user's future scheduled reservations. The user will be able to access this interface from a link provided from the home screen interface. The primary objective of this interface is to provide accurate information about a user's reservation. The User Reservation Queue will be designed to be user friendly oriented. The User reservation Queue will also provide links to take the user to other external interfaces.
c) **Source of Input or Destination of Output:** The source of this output will come directly from the applications database. The interface will send out a query to open to populate the interface. The outputs destination will be displayed on this interface.
d) **Valid range, accuracy. And/or tolerance:** The query that displays out the information on the web page will need to be accurate and reflect the users previous and future reservations made from the home screen interface.
e) **Units of Measure:** The GUI will display certain units of measurement for the telescope to process. Right Ascension (RA), Declination (DEC), and Epoch are units that will be displayed on this interface.
f) **Timing:** The interface should be able to display the users scheduled reservations at any given time.
g) **Relationships to other inputs/outputs:** This interface will have a frequently active connection to the applications database by querying for user reservations and

displaying the results to the interface. As mentioned earlier the interface will provide links to other external interfaces.

h) **Screen formats/organization:** The screen format will be oriented to be user friendly and be expandable to take up the entire screen of the computer.
i) **Window formats:** The format of the window will depend on the certain browser that the user will be using. Specifically we will want the application to run on a Safari browser.
j) **Data formats:** Data will be displayed and formatted by whatever data is stored in the database. Data will need to be appropriate to the certain field.
k) **Command formats:** N/A
l) **End messages:** N/A

Admin Control Interface

a) **Name of Item:** Admin Control Interface
b) **Description:** The Admin Control Interface will be a GUI that will be available for only admins of the application for use. The GUI will provide the functionality to manage other user queues, add users, override pending reservations, and add privileges to users. The admin will be able to access this interface from a link provided from the home screen interface. The primary objective of this interface is to have full control of the application and full oversight of the user activity. The Admin Control Interface will be designed to be friend user oriented. The Admin Control Interface will also provide links to take the user to other external interfaces.
c) **Source of Input or Destination of Output:** All scheduled queues will be displayed on this interface. The source of this output will come directly from the applications database. The interface will send out a query to open to populate the interface. The admin can ultimately override reservations which will send a query to the database. This input will come directly from the admin user on this interface.
d) **Valid range, accuracy. And/or tolerance:** The query that displays out the information on the web page will need to be accurate and reflect all users previous and future reservations made from the their home screen interface.
e) **Units of Measure:** The GUI will display certain units of measurement for the telescope to process. Right Ascension (RA), Declination (DEC), and Epoch are units that will be displayed on this interface.
f) **Timing:** The interface should be able to display all users scheduled reservations at any given time.
g) **Relationships to other inputs/outputs:** This interface will have a frequently active connection to the applications database by querying for user reservations and

displaying the results to the interface. As mentioned earlier the interface will provide links to other external interfaces.

h) **Screen formats/organization:** The screen format will be oriented to be user friendly and be expandable to take up the entire screen of the computer.

i) **Window formats:** The format of the window will depend on the certain browser that the user will be using. Specifically we will want the application to run on a Safari browser.

j) **Data formats:** Data will be displayed and formatted by whatever data is stored in the database. Data will need to be appropriate to the certain field.

k) **Command formats:** Not Applicable

l) **End messages:** Not Applicable

3.2. Functions

The final product will need to provide specific functions to ensure that the users request gets ultimately stored into the applications database and processed by the end system telescope software. Key user situations will also need to be taken into considerations when designing the product. These functions are described in this section.

3.2.1. Agile Use Cases

Use Case Name:	**Create Account**
Related Requirements	Must be a SIUe student enrolled in Clients course.
Goal in Context	To successfully add an account in the system.
Preconditions	No pre-conditions.
Successful End Condition	Account is Created and is Assigned to default role of Student.
Failed End Condition	None
Primary Actor(s)	User
Secondary Actor(s)	None
Trigger	User wants to create an account.

Main Flow	**Step**	**Action**
	1	User must go to website and click on the link to create the account
	2	User will fill out necessary information to create account.
	3	User will get a notification to check e-mail and verify account.
	4	User will click link and verify student account.
Extensions	**Step**	**Branching Action**
	4.1.1	Account has been successfully added into the application.

Table I: Agile Use Case for Creating an Account

Use Case Name:	**Assign Role**
Related Requirements	Must have Admin role to have this functionality. Role must be pre-existing for a User (Admin, Graduate, Undergrad)
Goal in Context	To successfully add a role in the system.
Preconditions	Role must be pre-existing
Successful End Condition	Role is assigned.
Failed End Condition	None
Primary Actor(s)	Admin
Secondary Actor(s)	Graduate, Undergraduate
Trigger	Admin wants to assign new role.

Main Flow	**Step**	**Action**
	1	Admin Selects Role to be Assigned
	2	Admin Assigns Role to user
Extensions	**Step**	**Branching Action**
	1.1.1	Role can be created for Admin, Graduate, and Undergraduate

Table II: Agile Use Case for Assigning a Role

Use Case Name:	**Enter Reservation**
Related Requirements	Must be a user in the system for this functionality.
Goal in Context	To successfully schedule a Reservation in the system.
Preconditions	User must be pre-existing,
Successful End Condition	Reservation is created and user is notified via e-mail.
Failed End Condition	Reservation is not created due to timing constraints.
Primary Actor(s)	User
Secondary Actor(s)	Admin
Trigger	User wants to make a reservation in the system to schedule an observation.

Main Flow	Step	Action
	1	User logs into home page and clicks link to create a reservation
	2	User is re-directed to page and begins to fill out required information
	3	User submits reservation and receives feedback
Extensions	**Step**	**Branching Action**
	1.1.1	User gets an error message displaying “Invalid username/password”.
	1.1.2	User will have the option to reset password.
	3.1.1	User receives a message that their reservation has been submitted and is scheduled. User will also receive an e-mail confirming.
	3.2.1	User will receive an error message display “Due to timing constraints, the time slot you have specified is taken” Please specify another time slot.

Table III: Agile Use Case for Entering a Reservation

Use Case Name:	Modify Reservation
Related Requirements	User must have one of the following Roles: Admin, Student, or Graduate.
Goal in Context	To successfully modify a pre-existing reservation.
Preconditions	In order to modify a reservation, the reservation must already exist in the database.
Successful End Condition	Targeted reservation is modified.
Failed End Condition	None
Primary Actor(s)	User
Secondary Actor(s)	None
Trigger	User wants to change a setting on their reservation.

Main Flow	Step	Action
	1	User selects reservation to be modified
	2	The selected reservation is in edit mode and user can now make changes
	3	User makes changes and confirms modification
Extensions	**Step**	**Branching Action**
	1.1.1	User is able to select any of their own pre-existing reservations.
	1.2.1	User of role type Admin is able to select any user pre-existing reservations.

	3.1.1	User receives a message that their reservation has been submitted and is scheduled. User will also receive an e-mail confirming.
	3.2.1	User will receive an error message display “Due to timing constraints, the time slot you have specified is taken” Please specify another time slot.

Table IV: Agile Use Case for Modifying a Reservation

Use Case Name:	**View Picture**
Related Requirements	User must have one of the following Roles: Admin, Student, or Graduate.
Goal in Context	To successfully view their pictures taken from a reservation.
Preconditions	User must have scheduled a reservation.
Successful End Condition	Pictures are available for view.
Failed End Condition	None
Primary Actor(s)	Users
Secondary Actor(s)	None
Trigger	User wants to view their pictures.

Main Flow	**Step**	**Action**
	1	User selects link to view pictures
	2	Pictures are displayed.
Extensions	**Step**	**Branching Action**
	2.1.1	User is able to view pictures in correlation with date scheduled.

Table V: Agile Use Case for Viewing a Photo

Use Case Name:	**View Reservation**
Related Requirements	User must have one of the following Roles: Admin, Student, or Graduate.
Goal in Context	To successfully view their Reservations that has been successfully scheduled.
Preconditions	User must have scheduled a reservation.
Successful End Condition	Reservations are viewable to user.
Failed End Condition	None

Primary Actor(s)	Users
Secondary Actor(s)	None
Trigger	User wants to view their reservations.

Main Flow	**Step**	**Action**
	1	User selects link to view reservation
	2	Reservations are displayed.
Extensions	**Step**	**Branching Action**
	2.1.1	User is able to view reservations in correlation with date scheduled.
Main Flow	**Step**	**Action**
	1	User selects link to view Reservation
	2	Reservations are displayed.

Table VI: Agile Use Case for Viewing a Reservation

Use Case Name:	**Cancel Reservation**
Related Requirements	User must have one of the following Roles: Admin, Student, or Graduate.
Goal in Context	To successfully cancel a pre-existing reservation.
Preconditions	In order to cancel a reservation, the reservation must already exist in the database.
Successful End Condition	Targeted reservation is canceled and database is updated.
Failed End Condition	None
Primary Actor(s)	User
Secondary Actor(s)	Graduate, Undergraduate
Trigger	User wants to cancel their reservation.

Main Flow	**Step**	**Action**
	1	User selects reservation to be canceled
	2	User canceled reservation.
	3	Database is updated
Extensions	**Step**	**Branching Action**
	1.1.1	User is able to select any of their own pre-existing reservations.
	1.2.1	User of role type Admin is able to select any user pre-existing reservations.
	3.1.1	User receives a message that their reservation has been canceled. User will also receive an e-mail confirming.

	3.2.1	Database is updated successfully.

Table VII: Agile Use Case for Canceling a Reservation

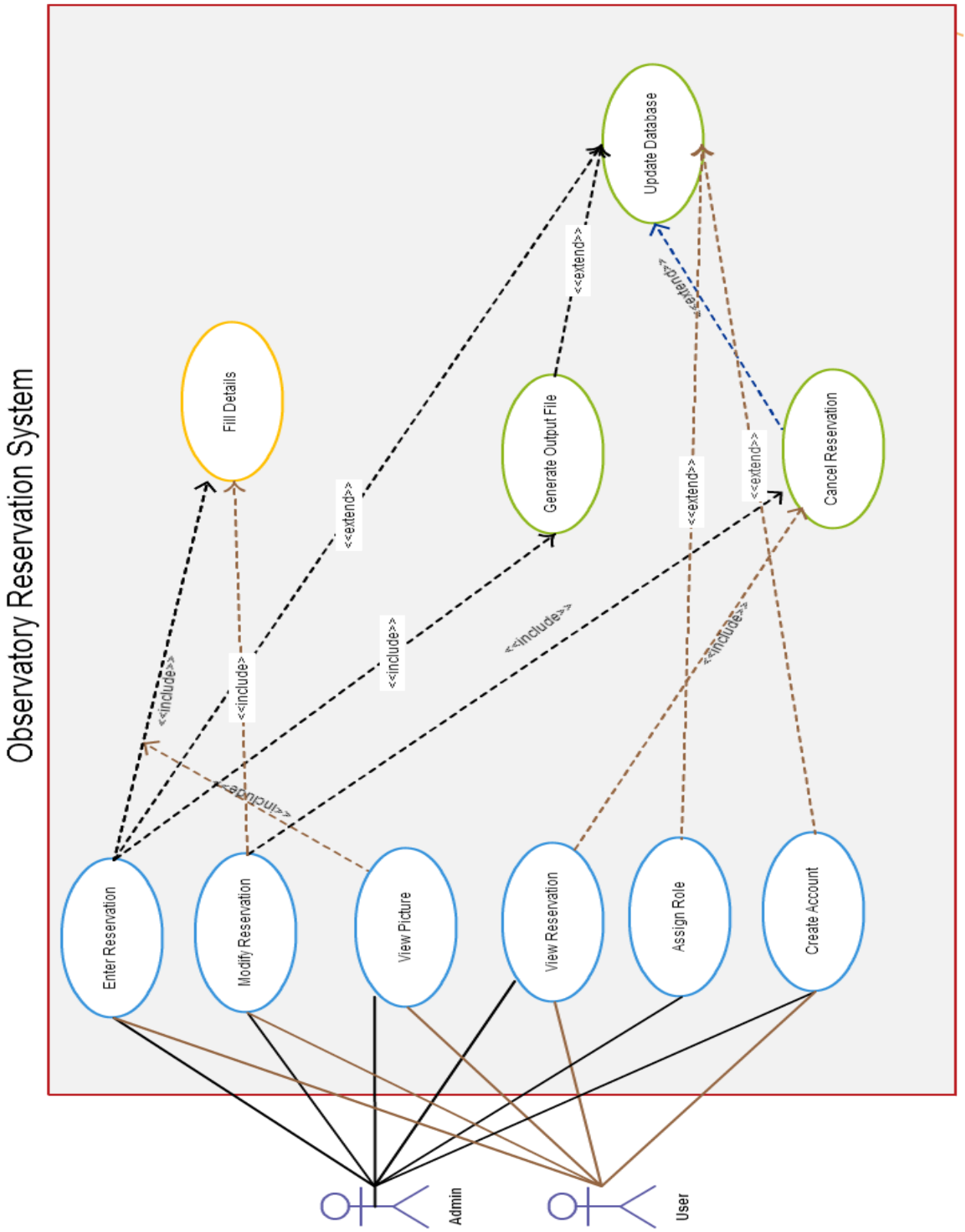

Figure 2: Agile Use Case Diagram for Proposed Observatory Reservation System

3.3. Usability Requirements

The usability of the product is essential to the users and the client. In this section, usability requirements will be defined. The requirements will assure the users confidence in using the application.

Simplification

The client has specified for the application to have a modern feel and sleek look. The look will provide for a simplistic interaction between the user and the application. Simplification is Key and is an important requirement for the application. The user must be able to easily enter information into the Home Screen Interface.

Navigation

The user must also be able to easily navigate through the entire web application with ease. The objective of this requirement is to reduce the amount of time it takes for a user to perform the main application tasks. Such task include, creating a reservation, viewing scheduled pictures, and logging in and out of the application.

Efficiency

The application must meet the goal of scheduling every reservation in a timely and appropriate manner. Users must be able to see all changes quickly once made. In terms of efficiency, the web application must generate the files for telescope quickly and prompt.

3.4. Performance Requirements

At this time, the client has not provided any performance requirements for this application. The application is intended to be used for a small audience of users. Real time updates is not a factor for this application.

3.5. Logical Database Requirements

Frequency of Use

The database will need to be opened and accessed for queries frequently throughout the applications use. The database will be used for but not limited to verifying user accounts, pulling user information, pulling user pictures, pulling user reservation requests, entering a reservation, cancelling a reservation, and modifying reservations.

Accessing Capabilities

The database should never be manually accessed by any users. The only time the database Should be accessed manually is to make data fixes and add Admin users. On the other hand, the application will be frequently accessing the database.

Data Retention Requirements

Data should be retained for an indefinite amount of time. Student accounts should be deactivated promptly after the course has ended or at the will at the admin at any given time. Data should never be deleted once entered into the database.

3.6. Design Constraints

Application Gap

The software that feeds the information to the telescope takes in information in the form of a specific file. Specification on the format of the file is coming soon. The web application will be forced to create these specially formatted files and storing them on the UNIX server. After the files are created the admin will then have to manually transfer files over to another server and upload them onto the telescope application. This "Application Gap" will disallow for a fully automated process. Until further notice, we will continue to create these files until we are instructed otherwise.

Safari © Compatible

The application needs to be able to run in a Safari © browser. All designs and interfaces will be wrapped around this.

3.7. Software System Attributes

Reliability

In order for the system to be reliable at the time of delivery we will need to ensure the indecency of the application and flow is running correctly and smoothly. As a team, we will conduct various and vigorous test to ensure the durability and insurability of the final end product. We will also meet with our client frequently at certain checkpoints of our SAGE processes.

Availability

Many factors are required to guarantee the availability level for the entire system. For example, backup and recovery procedures will be developed to make sure just in case of an event the applications data will not be deleted. The system should be available 24/7 and should not go down unless specified for maintenance changes.

Security

The observatory project will not require high security requirements. Since the end server for telescope will not be connected to the application we will not have to worry about accessibility issues. With that said, minimal security will be in place to insure all users' passwords and e-mail are securely stored in the application database.

4. Software Verification

4.1. Purpose

A verification plan is needed to provide guidance for the technical detail needed throughout the course of the testing period. This verification plan will mention the necessary requirements that will be tested as well as how it will be tested. Expected results from the necessary requirements will also be displayed. Non-functional requirements also need to be tested and what is expected out of each of them. A list of different testing methods that will be used can be shown in detail in Appendix A.

4.2. Performance Requirements Testing

The client has made it clear the project will not need to run at real time. It is a mix of an automated front-end with manual backend requirements. The observatory scheduler to be implemented accepts a completed form filled out by a user. These forms are turned into proposals and added to a priority queue. The queue needs to be updated in real time via the user's level of priority and the proposal date the user inputted. The client stated after a proposal is created, the process of taking images from observatory turns into a manual process that the client will control. Retrieving images is not something that is done automatically.

4.3. Functional Requirements Testing

4.3.1. User Proposal Form

The proposal form is the way a user creates a proposal for access to the observatory at a given time. Users need to login and the password and username should be hashed in some way.
Users need to lookup objects in a catalog which should automatically fill out that portion of the form. This will require black box testing as well as white box testing in order to make sure everything works properly.
Users can also select if they would like the picture to be taken at a specific time or anytime. White box testing will be needed in order to ensure the photographs are taken at proper times. Retrieving the photos is a manual process so we will need to do this by including the timestamp for when each photograph was taken.

4.3.2. Priority System

There are different level of priorities users in this system can have. Dr. Sabby will be the admin with the highest priority. The second highest priority is graduate students and the next highest is the undergraduate students.

Testing for this queue is most certainly needed. Admins can override anyone and graduate students can override undergraduate students. Test cases will be needed to ensure this is what is happening after the implementation is done.

There is also another level of priority. If a graduate student overrides an undergraduate student for some specific time, the undergraduate's proposal will get pushed back to the next day at the same time. Now this could keep happening so a priority referred to as 'recycled' is introduced. The longer the user stays in the recycled queue, the higher their priority. After implementation, this needs to be tested thoroughly. Black box testing is needed to see a visual of the queue being updated. Unit testing is something that will be used to test this portion of the project as well.

4.3.3. Generate Proposal File

Another functionality that might need testing is generating a proposal file in the proper format. This file is needed as it is what gets manually uploaded to the observatory's server in order to complete a proposal and take the required photographs.

Once a user creates a proposal, then this file should be created and sent to the database. This proposal file needs to be in the proper format or else the observatory software will not be able to read the file and could cause an error or just discard the proposal.

Once this is implemented, black box testing will be used in order to see if the file output is in the proper position and to ensure every detail is accurate and what the user intended.

(Note: This is a feature that was discussed, but not something the client required. The idea is still a work in progress.)

4.3.3. Scan Photos

A couple key things in this project need to be performed manually and there isn't really a way around this. If the project has to remain this way then there should be a way to speed the manual process up. When the telescope takes pictures, the images are saved to the observatory's server. Dr. Sabby needs to login to acquire these photos. The photos need to be associated with the user that took the photo and placed into a directory the user can access. In order to avoid this being a manual process, a script can be used to scan each image file's name. The name of the image should

include the user's ID as well as a timestamp. This script should scan the image names and place in the correct directory on the server. This script needs to run on Max OS X and be ran daily.

Test cases and unit testing will play a large part for testing the script and its functionality. Test cases of all different types should be used in order to stress test the script. Dr. Sabby stated 500 images could be taken in a given day. The script needs to be able to handle that and continue to run no matter the amount of pictures. Many test cases should be used to ensure the images are always being uploaded to the correct directory no matter the inputted field or timestamps. An image should only be ignored if the user's ID or timestamp is not present.

In order for the script to work properly, the results from running the script will have each image taken for a given day and uploaded to the proper directory.

(Note: This is a feature that was discussed, but not something the client required. The idea is still a work in progress.)

4.4. System Testing

The project has a few parts that need to work together in order to function properly. Below is a typical scenario of what happens in the process and how things are getting checked.

A user logins to the webpage application via email. Credentials need to be approved before this happens. After a successful login, the user fills out the proposal form. Every field needs to be tested prior to release to ensure the system is only accepting the correct and necessary data. Once the user submits the form, a file is generated and their request is placed in a priority queue. The file will have been tested during the testing phase of the project to ensure the file is formatted properly in order for the observatory software to read the file in correctly. The queue the user is placed into also needs to be updated in real time for every new request made or modified by the admin. The user is then able to logout. This is where the process shifts to a manual process. Dr. Sabby will need to add these proposal files to the observatory's software. After the files are scanned and executed by the server, image files are created. Dr. Sabby will then log back into the server and run the created script in order to add these images to the proper directory in the database. Testing will have been done prior to ensure the images are being placed in the correct directory at the correct time.

Some parts need to work together and other parts do not. This is because the two manual processes needed. This project needs to work as expected to ensure no harm is done to the observatory's software and server as well as Dr. Sabby's server. Nothing should be executed that could cause harm to the operating environment.

5. Appendices

5.1. Assumptions

Below is a list of assumptions that need to happen in order for this project to be successful.

- Users need to create proposals and fill out the required information correctly. Every field has to be correct in order for the generated file to be valid.
- The client needs to manually add the files generated onto the observatory's server and then load them into the observatory's software in order to execute
- The client needs to run the script on the his own server in order to add the images to the proper directory

5.2. Appendix A

In software development there are many software methods that can be used to test a system or find software bugs. Methods for this project include black-box testing, white-box testing, unit testing and system testing. There is the possibility for others to be used but these are the main ones for now. Below is the definition of each and why they are being used for this project.

5.2.1. Black-Box Testing

This method is being used for a most of the functional requirements. Black-box testing is testing the program without seeing the internals of it and the code. It does not focus on design, just what you see. Black-box testing for this project will be used to ensure file output is correct, to detect bugs and to make sure fields for the proposals only accept the required information and nothing else.

5.2.2. White-Box Testing

White-box testing tests the internals and structure of a program. The user does not have access to this portion of the project and it's an effective testing method if things are not going as planned. The tester created input for the fields and tries to break the program. If it does break the tester goes through the code and tries to determine where it went wrong. Debugging is very useful for this method. For this project, white-box testing will be utilized heavily and not just for the testing phase, but implementation phase as well.

5.2.3. Unit Testing

Unit testing, or component testing, is used to test specific sections of the program. This method goes hand-in-hand with the white-box testing. These tests are written by developers and testers to ensure specific functions and classes are working properly. Unit testing is a method used to increase the quality of the software. For this project, unit tests are used to verify methods and code are working as intended.

5.2.4. System Testing

System testing is used once separate interfaces are finished and integrated together. The point of this method is to make sure the two are working properly together and are bug free. Developers do not want the two to cause conflicts and create system errors when the different systems work together. For this project, system testing is used to ensure the proposal files are created as well as placed in a priority queue.