Document No. 123456-D

1 March 2015

Baseline

GBL
Answer Repository
Software System

Contract No: 123456

Prepared for:

Global Business Logisitics

5551 551st St. NW Suite 555

Tacoma, WA 98455

Prepared by:

Red Menace Software Solutions

1900 Commerce St.

Tacoma, WA 98402

Christopher Helmer

Dan Taylor

Kimberly Stewart

Jacob Peterson

Brandon Lioce

Table of Contents

1.0 Introduction

The Red Menace design team has chosen its design starting from the idea of Model-View-Control. The view is the GUI features. These features allow the users to access. The LogoPanel will hold a logo for the GUI just for aesthetics. The LoginPanel will allow the user to log in with a user name and pin. After a user logs in, the other panels will allow the user to access the different parts of the program. There are two distinct model sets of classes. The UserInfo and User classes maintain data about the users who can log into the system. These classes also authenticate any users. The other models are the Keyword and Category classes. These classes hold the answers and give the search results. The final class, Session, is the system control. Session manages the queries and collects the results.
Our code will also be using the Observer/Observable pattern. When session collects the results from a search it will send a notification. The answer panel will pick up that notification and display the results of the search. This process will help decouple the classes in the view and the model.

2.0 Summary

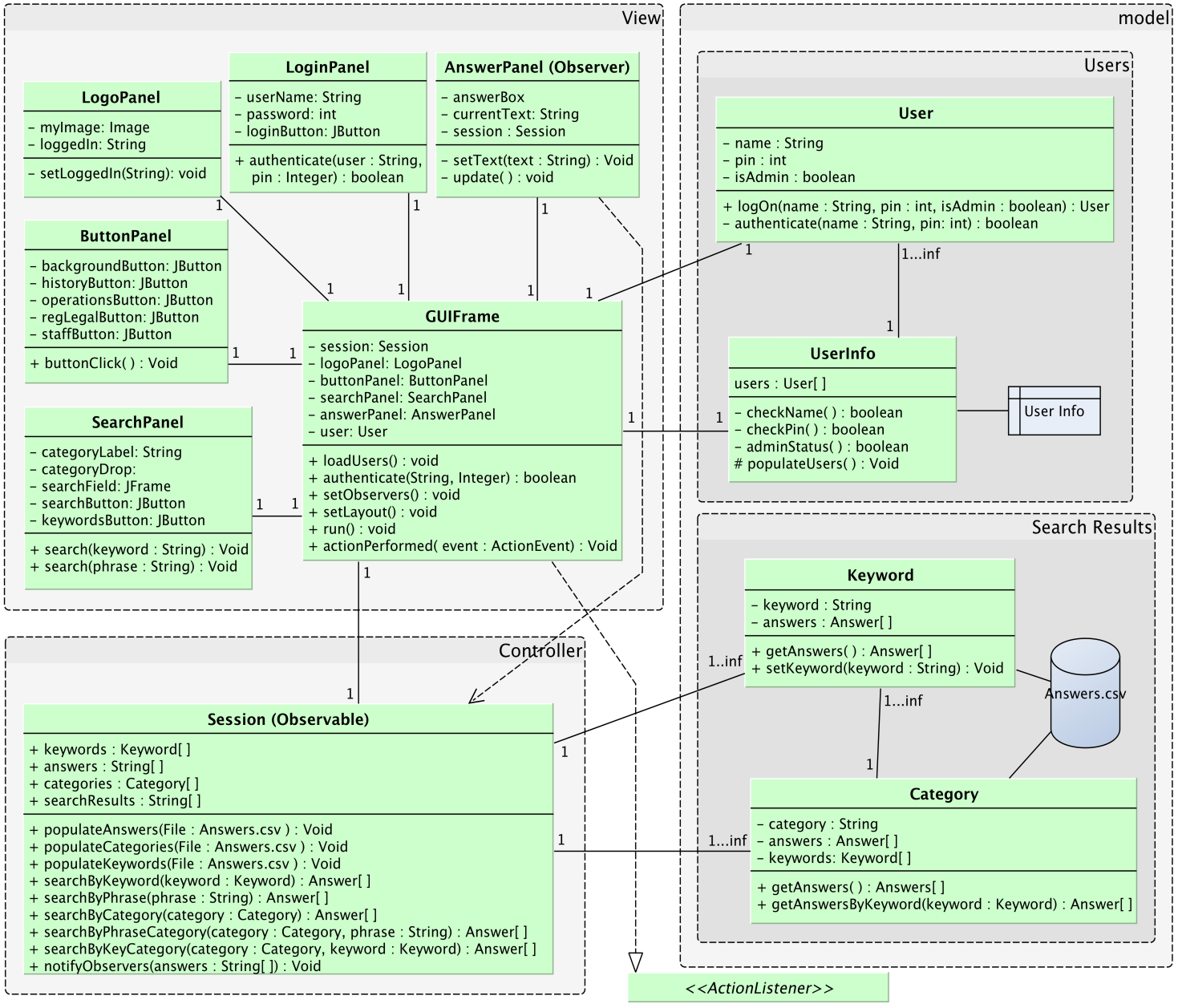
From the start, we wanted our program to follow the guidelines for a good design. We would attempt to make all of the classes loosely coupled, avoid god classes, make it efficient with the data we would be working with, and eliminate redundancy and proliferation of classes to make the code easy to read, easy to write, and easy to maintain/modify.

We began designing with the model-view-controller design in mind. This would allow for us to make different objects to represent different parts of the RFP database (such as keywords and users), use a GUI to display the information and allow the user to change it around to their liking, and a controller to handle all of the data behind the scenes.

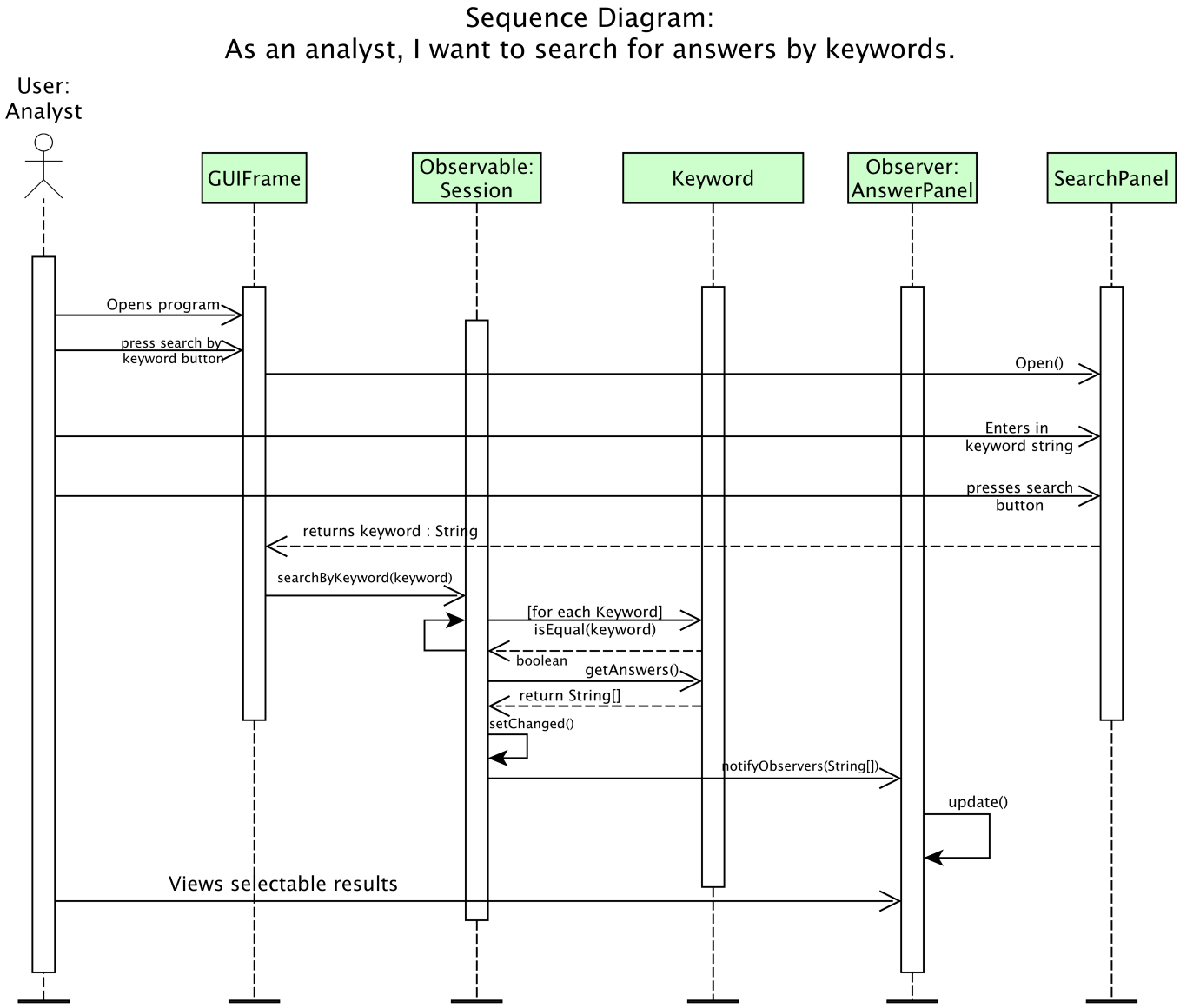
Within the controller, we decided it was best to stick with a single class. All of the data fields and accessors and sorters will be using all of the data, so splitting the functionality up to create smaller classes didn't make much sense. Keeping everything centralized keeps the data in a single place and helps to avoid classes that contain just setters and getters.

The GUI design is straightforward, with a single JFrame holding various panels that will change around based on the user interaction. The GUIFrame class interacts with the User class, allowing users of the system to log in and have their information displayed (this feature will be nothing more than this for now, but acts as groundwork that can easily be expanded in the future). We designed our program to use the Observer Design pattern. This would allow the GUI to efficiently be updated as the program is used. It would also allow for a loose coupling between the main controller of the program and the GUI itself. As the user searches for approved answers, the Session does all of the sorting and concatenating of the data while the AnswerPanel displays the data without having to do much data receiving and hard computation itself.

3.0 Class Diagrams

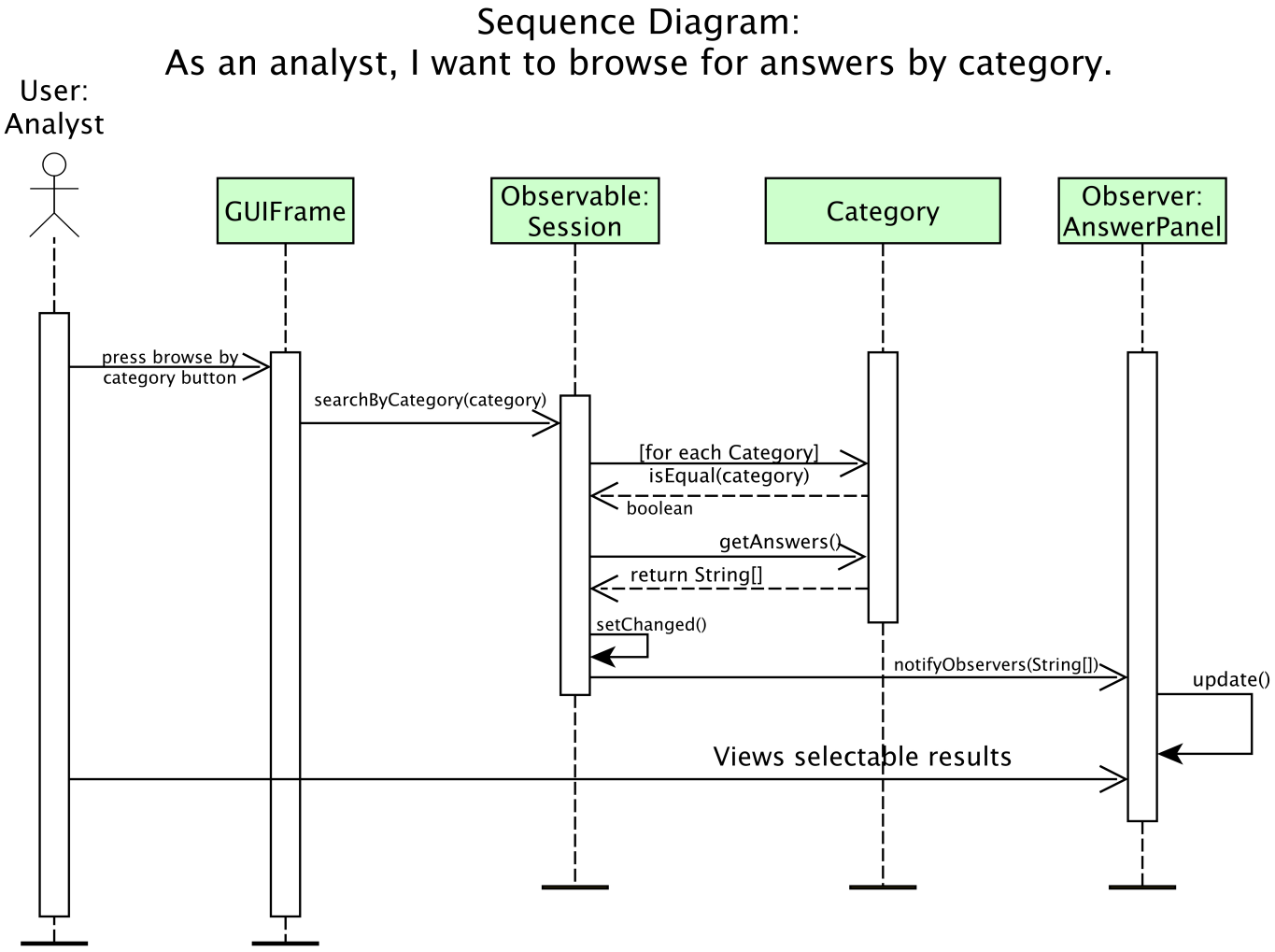


4.0 Sequence Diagrams



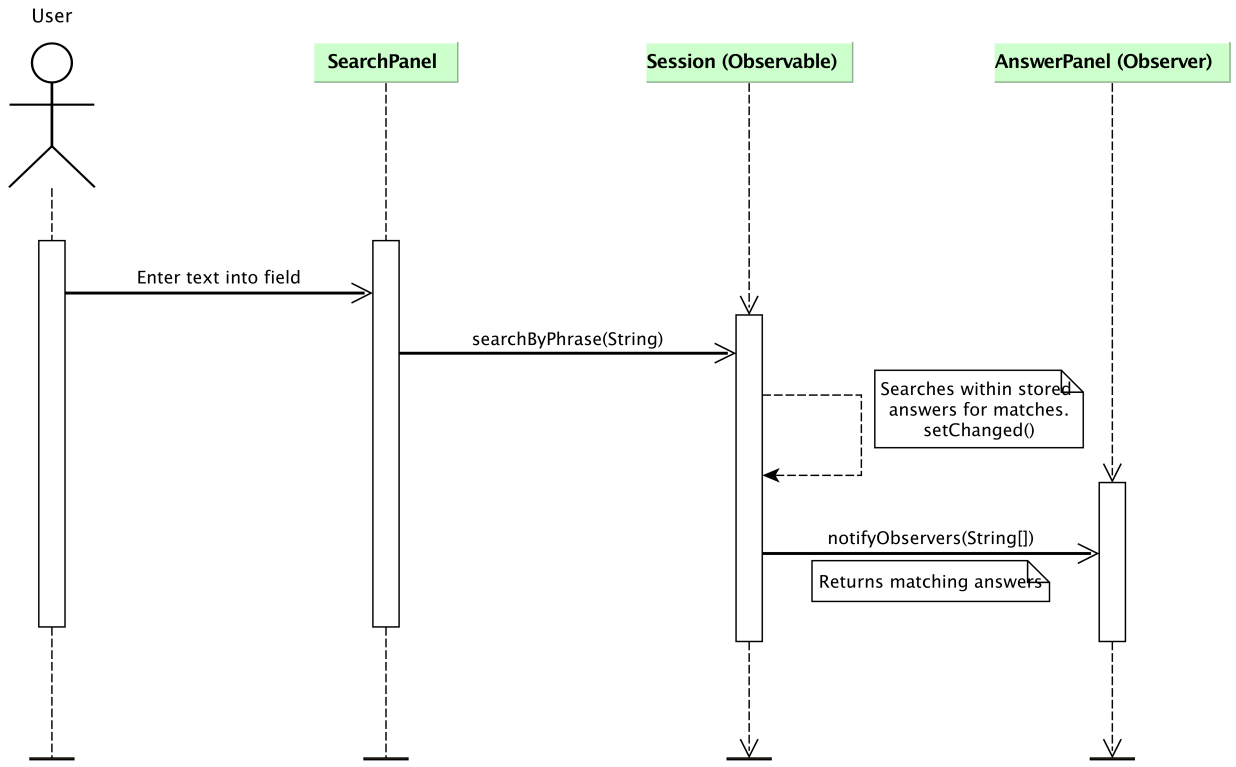
The analyst opens the program and presses the "Search by Keyword" button. The button triggers the search panel to open up. Then the user enters the keyword into an available text field. When the search button is pressed, the keyword is sent to the GUIFrame which calls searchByKeyword(keyword) in the Session class. The session class traverses the list of Keywords until it finds a match. Then Sessions gets the list of answers as a String[] from the keyword. It calls

setChanged() on itself and notifyObservers(String[]) to send the answers to the AnswerPanel. The AnswerPanel updates itself and the selectable text is displayed.



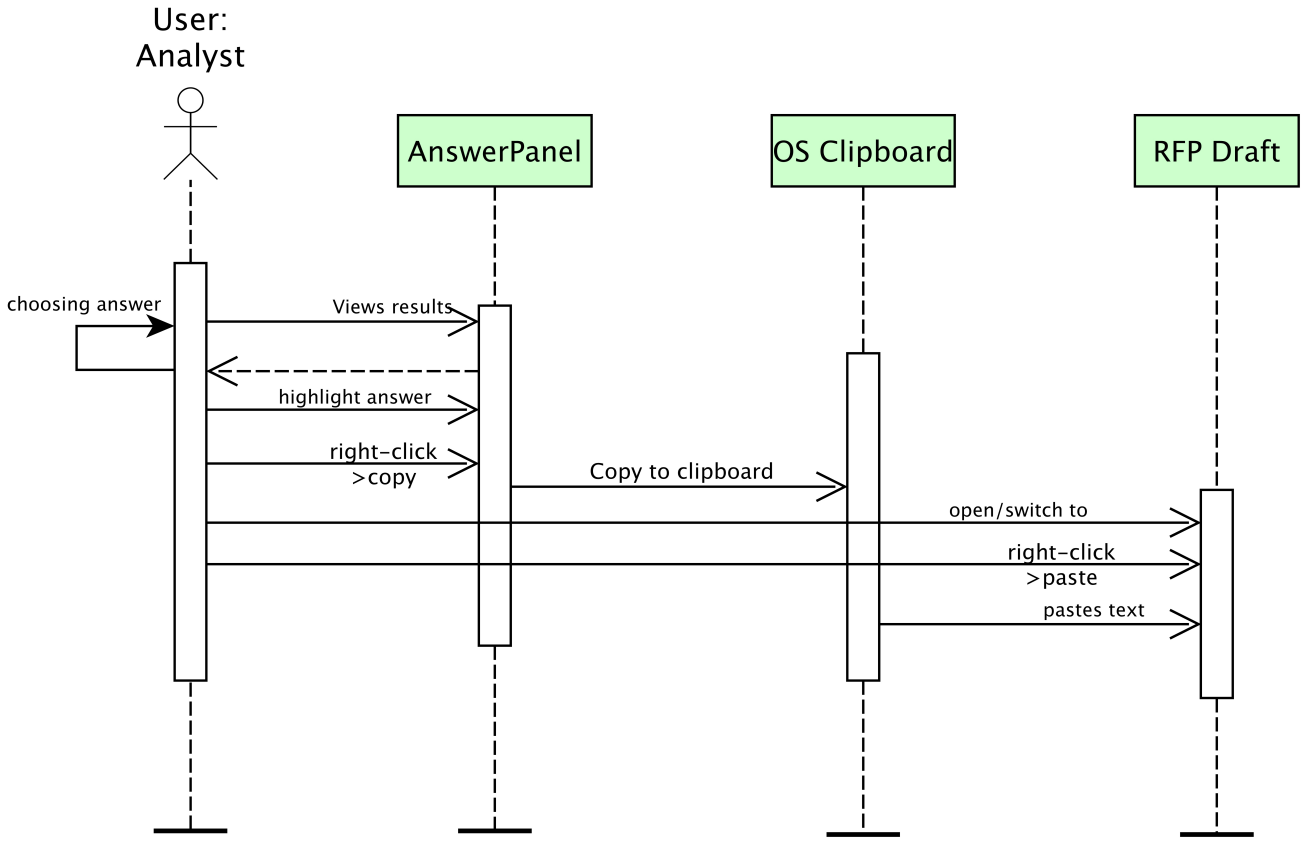
The analyst opens the program and presses the "Browse by Category" button. The category is sent to the GUIFrame which calls searchByCategory(category) in the Session class. The session class traverses the list of Categories until it finds a match. Then Sessions gets the list of answers as a String[] from the category. It calls setChanged() on itself and notifyObservers(String[]) to send the answers to the AnswerPanel. The AnswerPanel updates itself and the selectable text is displayed.

Sequence Diagram:
As an analyst I want to be able to find answers within the repository by searching a phrase



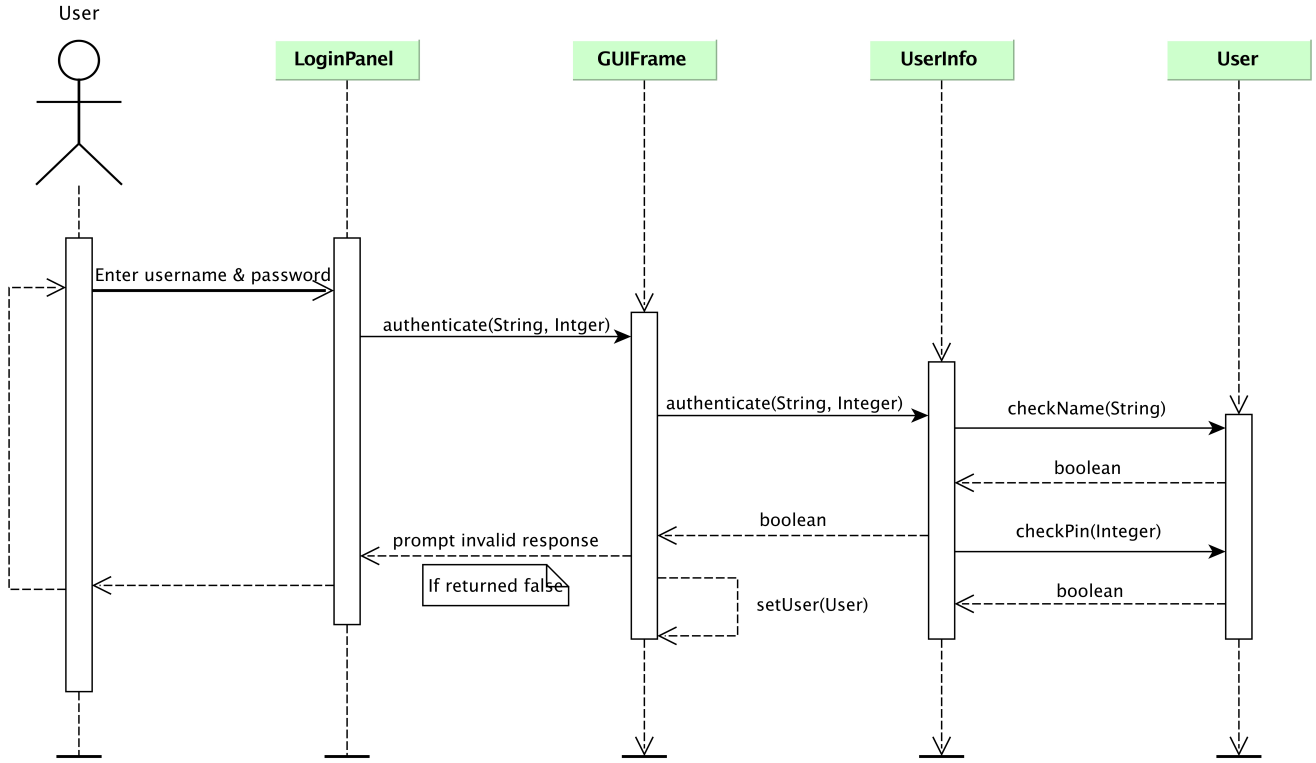
To locate an answer within the repository by using a phrase, the user will enter the desired phrase into the text field within the SearchPanel class. The phrase will be sent to the Session class by calling on its searchByPhrase(String) method. This method will then perform a string matching search on the answers stored within its stored array of answers. The state of the Session (Observable) class will be set to changed, and the notifyObservers(Object) method will be called, with the array of matching answers as its parameter. The AnswerPanel will then display the results, if any. If there are none, a default response will be displayed.

Sequence Diagram:
As an analyst, I want to copy and paste an answer to an RFP.



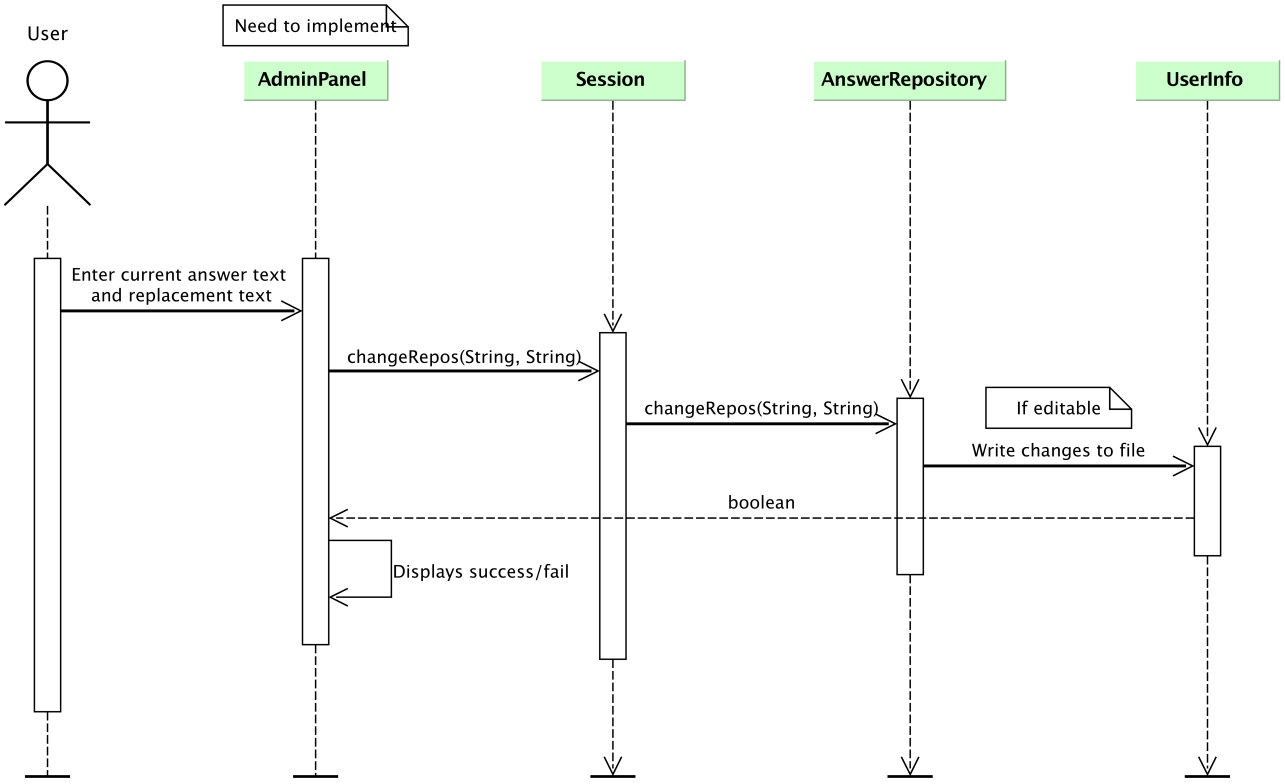
After the answer list is viewable, the Analyst views the available answers until they find the one that best suits the question they are working on. They then highlight the answer and right-click > copy the answer. The answer is then copied to the system's clipboard. The analyst opens or switches to the RFP response draft and right-click > pastes the answer into the open draft.

Sequence Diagram:
As a user I want to logon the local device in the role of an Analyst or an Administrator



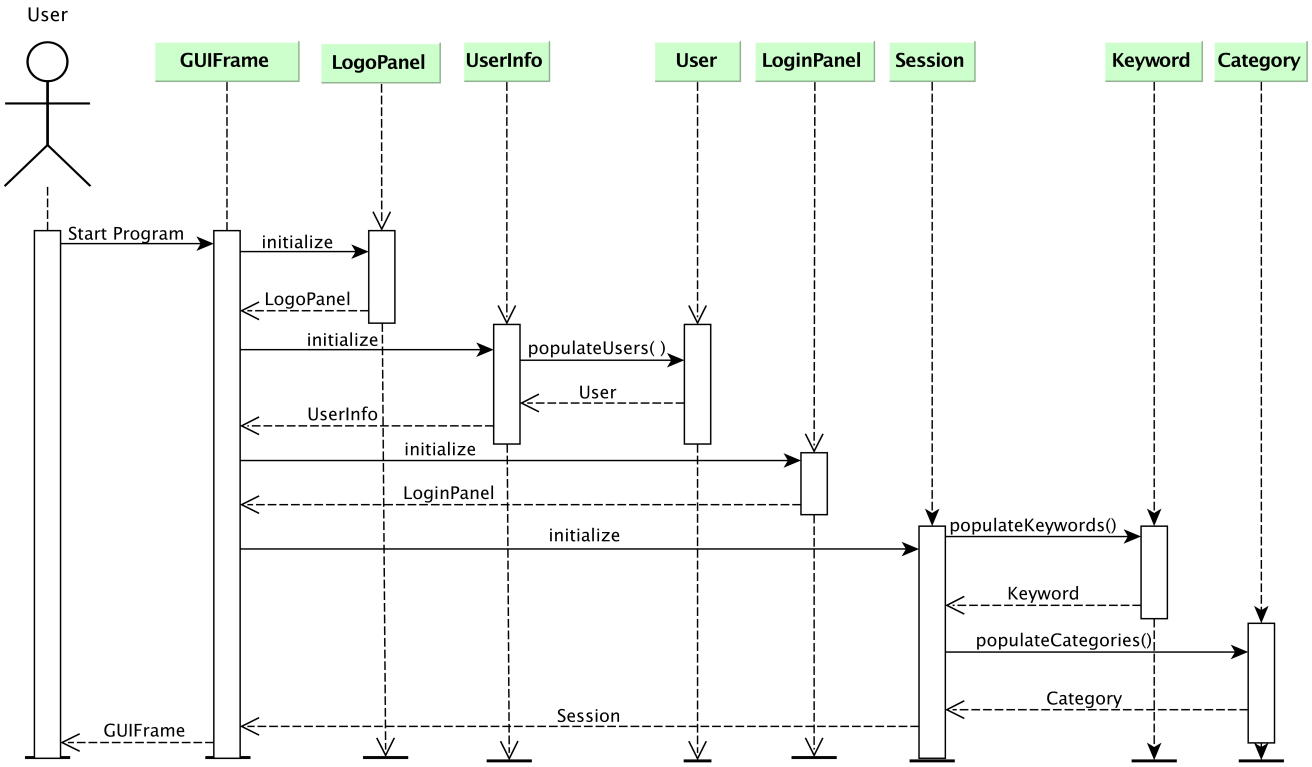
In order to login, the user will interact with the LoginPanel, which is a part of the main JFrame GUI. The user will enter their appropriate credentials into the text fields on the panel and then click the 'Login' button, which will call the authenticate method. From there, the GUI will call its authenticate method which will query UserInfo. UserInfo will then search through its stored data of User objects to see if there is a name match. If there is, it will check to see if that user's pin matches the one provided. If the login is successful, the class will return the appropriate User object to the calling class where it will set the current user of the system. However, if the login was unsuccessful, the classes will return false and the user will be prompted to retry.

Sequence Diagram:
As an administrator I want to be able to change answers within the repository



In carrying out the task of editing the repository, the user will enter a string into a text field labeled for the current answer within the repository. Next, the user will enter a different string into a separate field, labeled for the updated answer. Upon clicking submit, the AdminPanel will call the changeRepos(String, String) method within the Session class. This method will call the same within the AnswerRepository class. There, the new string will be written to the data file. As long as the first string was successfully located within the repository, the method will return true. If the string was not found, the method will return false and the user will be prompted for a retry.

Sequence Diagram:
Start Up Sequence Diagram



When the program is fired up, it begins with the GUIFrame class. The GUIFrame class displays all the GUI features and will have a reference to the model. This can pretty much be done in any order, but for all intents and purposes, the LogoPanel will be done first. It returns itself to be placed into the GUIFrame. The next object to be started is the UserInfo. This class will hold all the users and authenticate the users that try to log in. Then the LoginPanel will be the next to be instantiated. We are not initializing any other panels, as depending on who the user is and the user's role, Analyst vs Admin, they may have different functionality present. The next class is the Session Class. This class, the control class for our program, then creates the Keyword and Category classes to aid in the searching features. This lays the basic framework for the program to function.