CSCI 4448 Project Part 7

1. **Team:**
    a. Taylor Jesse
    b. Samuel Reed
    c. Spencer Hanson

   Title: Decision Tree Visualization and Learning Tool

   Project Vision: Our project's goal is to provide a web service to facilitate building a decision tree. When a user creates an account and logs in, we plan on providing tools to help create a survey that will focus on a specific decision.

   Project Description: This will involve a list of questions and possible answers regarding situational attributes. Once a user creates a survey, we plan on generating a link so that the user can easily distribute the survey in order to accumulate sample data. Once there is a sufficient amount of sample data, the user can create a decision tree visualization of the result. We'll apply basic entropy information theory on the sample data in order to show the user what the decision pathway looks like from the individuals who took the survey. The ultimate goal is to provide an easy way to model decision making processes and show an informative visualization.

2. **List the features that were implemented (table with ID and title).**

Requirement ID:	Description:
REQ-01	As a survey admin, I want to log on so that I can access my surveys and decision trees.
REQ-03	As a survey admin, I want to save my surveys and decision trees so that I can access them at a later time.
REQ-04	As a survey admin, I want to create a survey so that I can use it as a template for sample data.
REQ-06	As a survey taker, I want to take a survey so that I can contribute to creating sample data.
REQ-07	As a survey admin, I want to create a decision tree so that I can understand the decision making process of my survey.
REQ-08	As a survey admin, I want to see a visualization of a decision tree so that I can see and understand the results of my survey.

REQ-09	As a survey admin, I want to create an account so that I can access the website’s tools.

3. **List the features were not implemented from Part 2 (table with ID and title).**

Requirement ID:	**Description:**
REQ-02	As a survey admin, I want to log off when I am ready to exit the website.
REQ-05	As a survey admin, I want to create a link to a survey so that I can send it to other users to take.

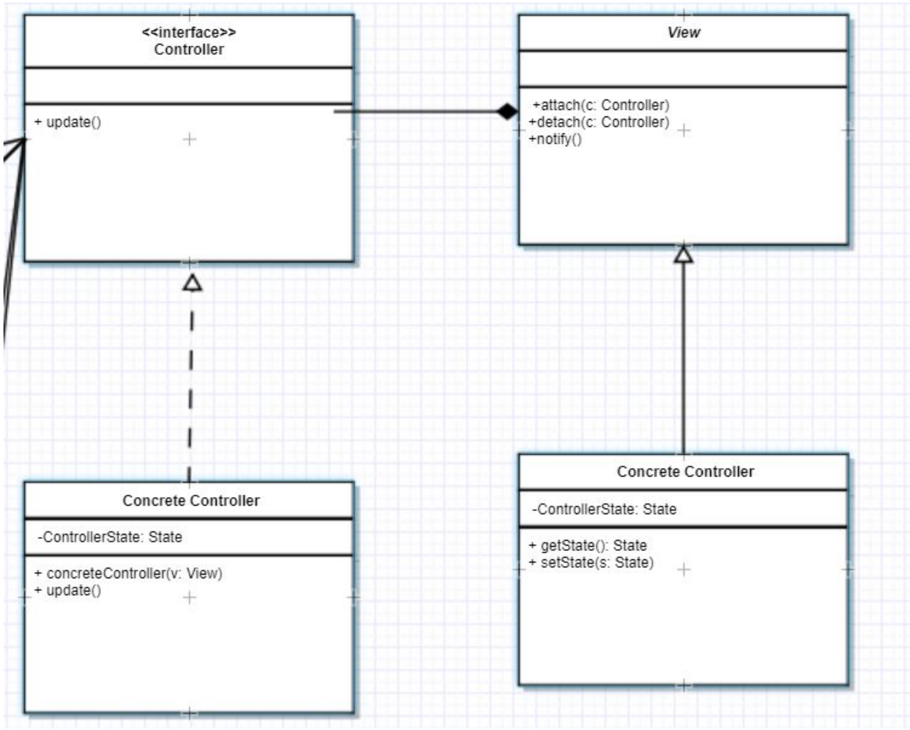
4. **Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.**

--Class Diagrams appended to back of this document--

The majority of our changes in our diagrams are in our model view controller for updating our UI and implementing our observer pattern, most of these changes occuring in our controller and view classes. The parts of the class diagrams that remained the same can be attributed to the work that we put in with all of our up front design. By walking through the different use cases and activity diagrams, we were able to solidify most of the structure of our project.

5. **Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF).**

The classes for our Observer Design Pattern:



6. **What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?**

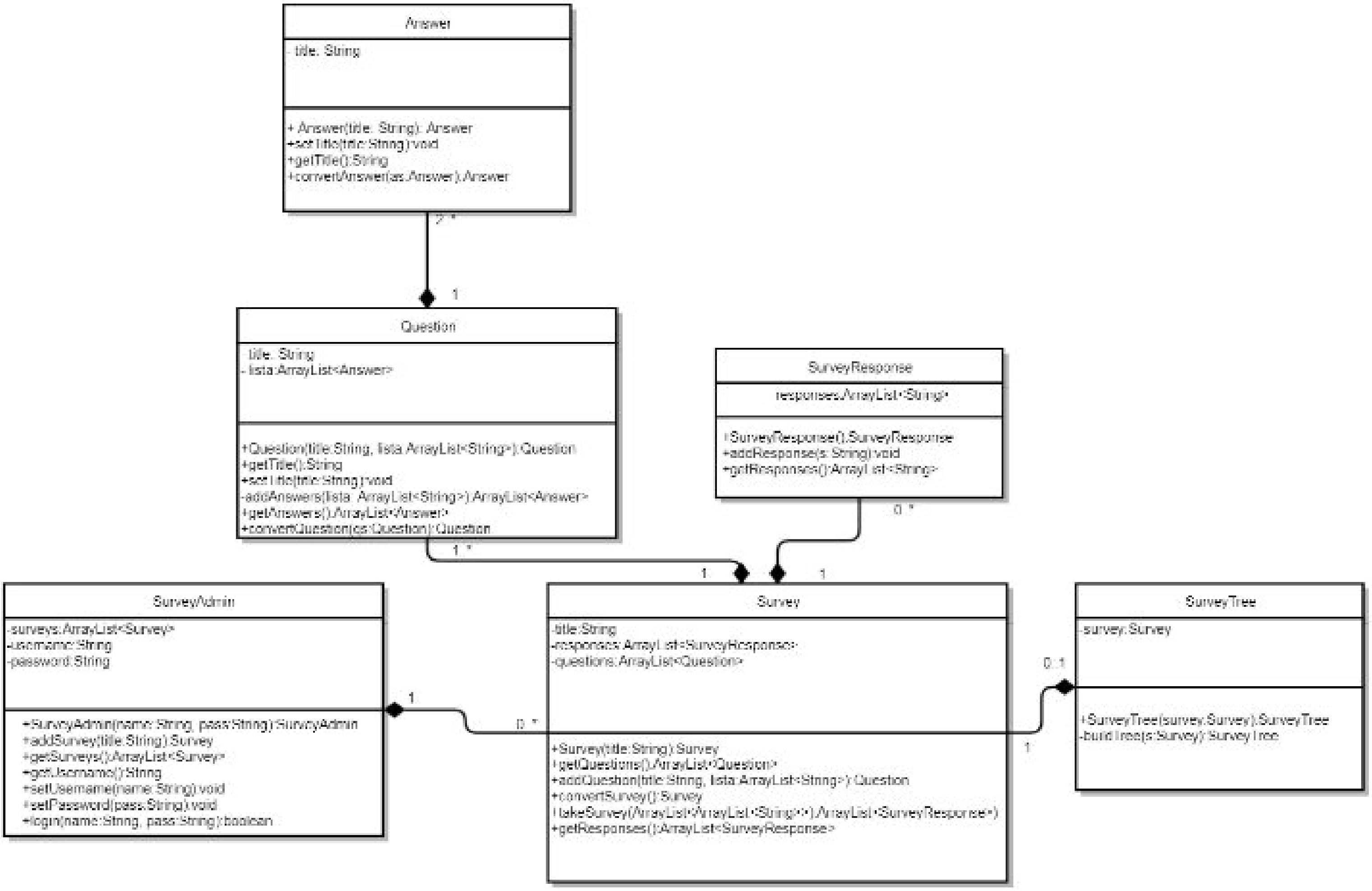
We have learned about different areas in the process of analysis and design throughout implementing our system.

First, we learned about how to work on a project with interlocking parts, using diagrams as a means to communication. By mapping out what had to be done in class diagrams, activity diagrams, and requirement tables, it made organizing and splitting up the work much easier than it usually is in group projects. Each person was responsible for figuring out their own piece, and getting it to work as we agreed in the specifications.

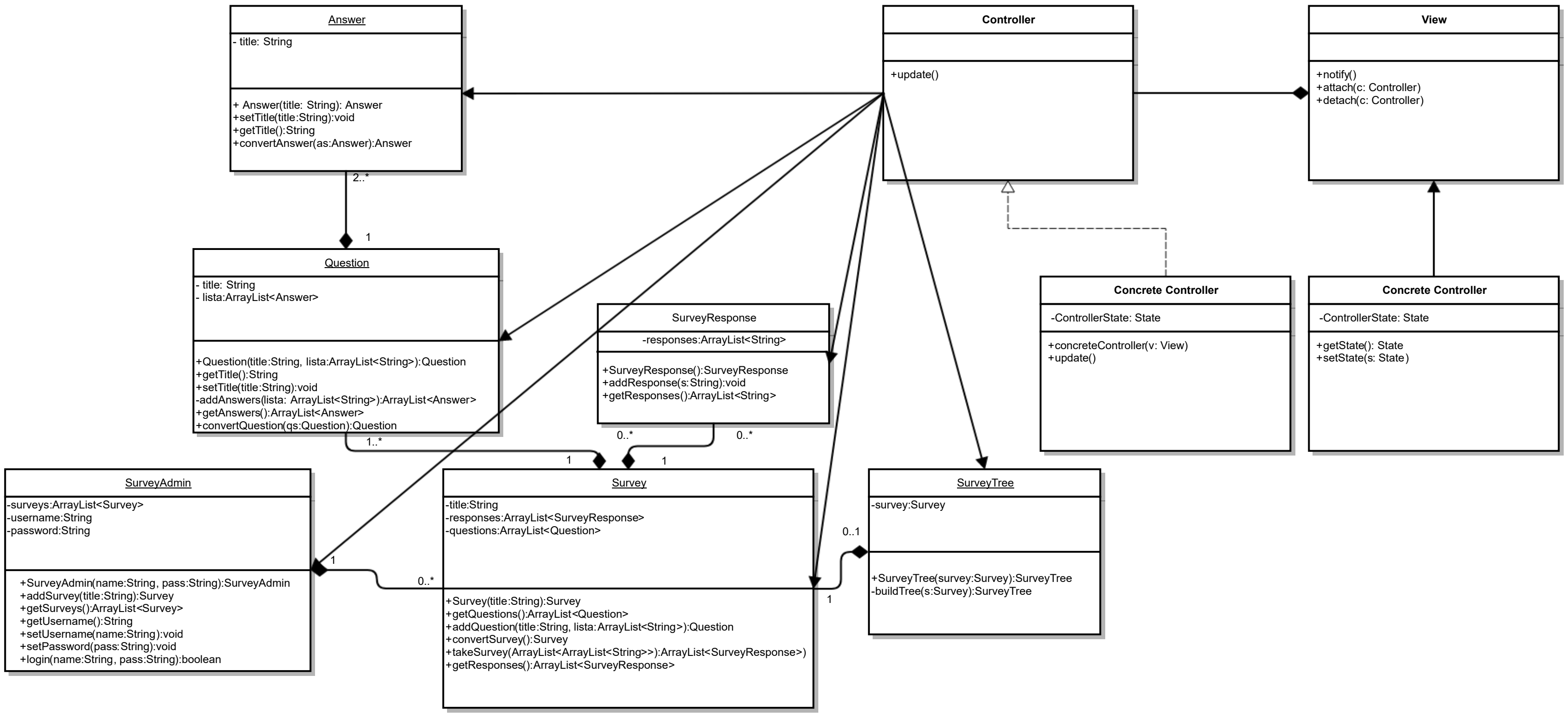
Second, we learned about the process of formally designing a flow within a product. Using activity diagrams as a guide for the design, we could then compact complex pieces together to make everything work. This was especially useful in the implementation of our menu system, where connection is necessary for a continual user experience, breaks could lead to confusion and overall a bad user interface.

Finally, breaking down the problem from an abstract notion of what is needed to be done, into actual code, was very useful. By having a collection of common patterns, it became a library of best practices, making breaking down from a very abstract context to actual code much easier than previous projects.

Part 2 Class Diagram


Answer
- title: String
+ Answer(title: String): Answer
+setTitle(title:String):void
+getTitle():String
+convertAnswer(as:Answer):Answer
2..*
1
Question
- title: String
- lista:ArrayList<Answer>
+Question(title:String, lista:ArrayList<String>):Question
+getTitle():String
+setTitle(title:String):void
-addAnswers(lista: ArrayList<String>):ArrayList<Answer>
+getAnswers():ArrayList<Answer>
+convertQuestion(qs:Question):Question
1..*
1
SurveyResponse
responses:ArrayList<String>
+SurveyResponse():SurveyResponse
+addResponse(s: String):void
+getResponses():ArrayList<String>
0..*
1
SurveyAdmin
-surveys:ArrayList<Survey>
-username:String
-password:String
+SurveyAdmin(name:String, pass:String):SurveyAdmin
+addSurvey(title:String):Survey
+getSurveys():ArrayList<Survey>
+getUsername():String
+setUsername(name:String):void
+setPassword(pass:String):void
+login(name:String, pass:String):boolean
1
0..*
Survey
-title:String
-responses:ArrayList<SurveyResponse>
-questions:ArrayList<Question>
+Survey(title:String):Survey
+getQuestions():ArrayList<Question>
+addQuestion(title:String, lista:ArrayList<String>):Question
+convertSurvey():Survey
+takeSurvey(ArrayList<ArrayList<String>>):ArrayList<SurveyResponse>)
+getResponses():ArrayList<SurveyResponse>
0..1
1
SurveyTree
-survey:Survey
+SurveyTree(survey:Survey):SurveyTree
-buildTree(s:Survey):SurveyTree

Final Class Diagram on Next Page


Answer
- title: String
+ Answer(title: String): Answer
+setTitle(title:String):void
+getTitle():String
+convertAnswer(as:Answer):Answer
2..*
1
Question
- title: String
- lista:ArrayList<Answer>
+Question(title:String, lista:ArrayList<String>):Question
+getTitle():String
+setTitle(title:String):void
-addAnswers(lista: ArrayList<String>):ArrayList<Answer>
+getAnswers():ArrayList<Answer>
+convertQuestion(qs:Question):Question
1..*
1
SurveyResponse
-responses:ArrayList<String>
+SurveyResponse():SurveyResponse
+addResponse(s:String):void
+getResponses():ArrayList<String>
0..*
0..*
1
Controller
+update()
View
+notify()
+attach(c: Controller)
+detach(c: Controller)
Concrete Controller
-ControllerState: State
+concreteController(v: View)
+update()
Concrete Controller
-ControllerState: State
+getState(): State
+setState(s: State)
SurveyAdmin
-surveys:ArrayList<Survey>
-username:String
-password:String
+SurveyAdmin(name:String, pass:String):SurveyAdmin
+addSurvey(title:String):Survey
+getSurveys():ArrayList<Survey>
+getUsername():String
+setUsername(name:String):void
+setPassword(pass:String):void
+login(name:String, pass:String):boolean
1
0..*
Survey
-title:String
-responses:ArrayList<SurveyResponse>
-questions:ArrayList<Question>
+Survey(title:String):Survey
+getQuestions():ArrayList<Question>
+addQuestion(title:String, lista:ArrayList<String>):Question
+convertSurvey():Survey
+takeSurvey(ArrayList<ArrayList<String>>):ArrayList<SurveyResponse>)
+getResponses():ArrayList<SurveyResponse>
0..1
1
SurveyTree
-survey:Survey
+SurveyTree(survey:Survey):SurveyTree
-buildTree(s:Survey):SurveyTree