10 - Rolypolly

Part 7: Final Report

Brandon Aguirre, Lauren Raddatz, Marco Ortiz Torres

1. **Vision**
   - Create a virtual polling system to replace iClickers.

   Project description
   - A web application which allows users to connect live to a central host and answer questions created by the host (similar to iClicker).
2. **List the features that were implemented (table with ID and title).**

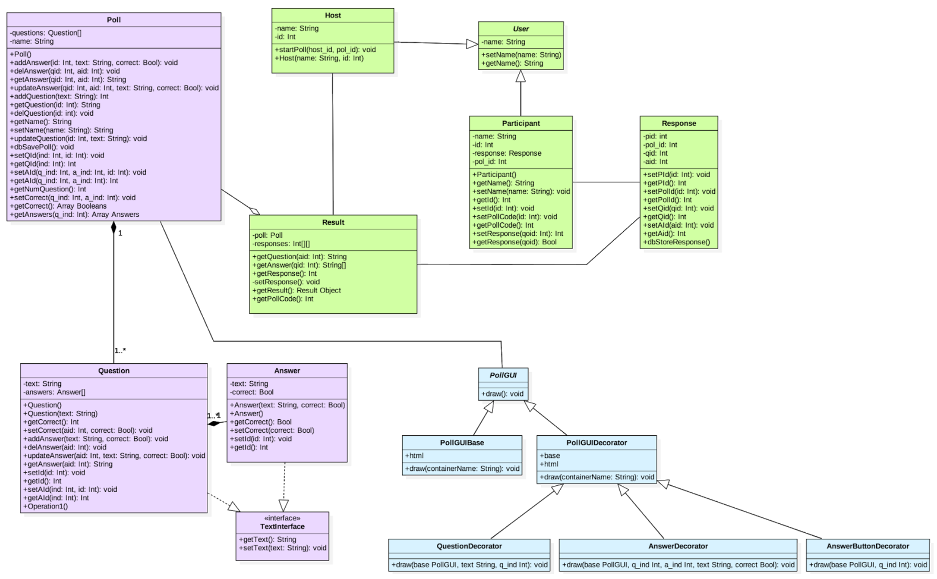
ID	Description	Priority
US-01	As a *host* I want to **create a poll** so that I can collect results	1
US-02	As a *host* I want to **see the group results** of a poll I created so that I can gather data about the performance of the *participants* who took my poll.	3
US-03	As a *participant* I want to **get my results** after I answer a question so that I know if I'm right or wrong (or how I am categorized in the results)	3
US-04	As a *participant* I want to **join a poll** so that I can answer questions	3
US-05	As a *participant* I want to **answer questions** so that I can contribute to a poll	1
US-06	As a *host* I want to **make a poll live** so that *participant* can take my poll	2

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

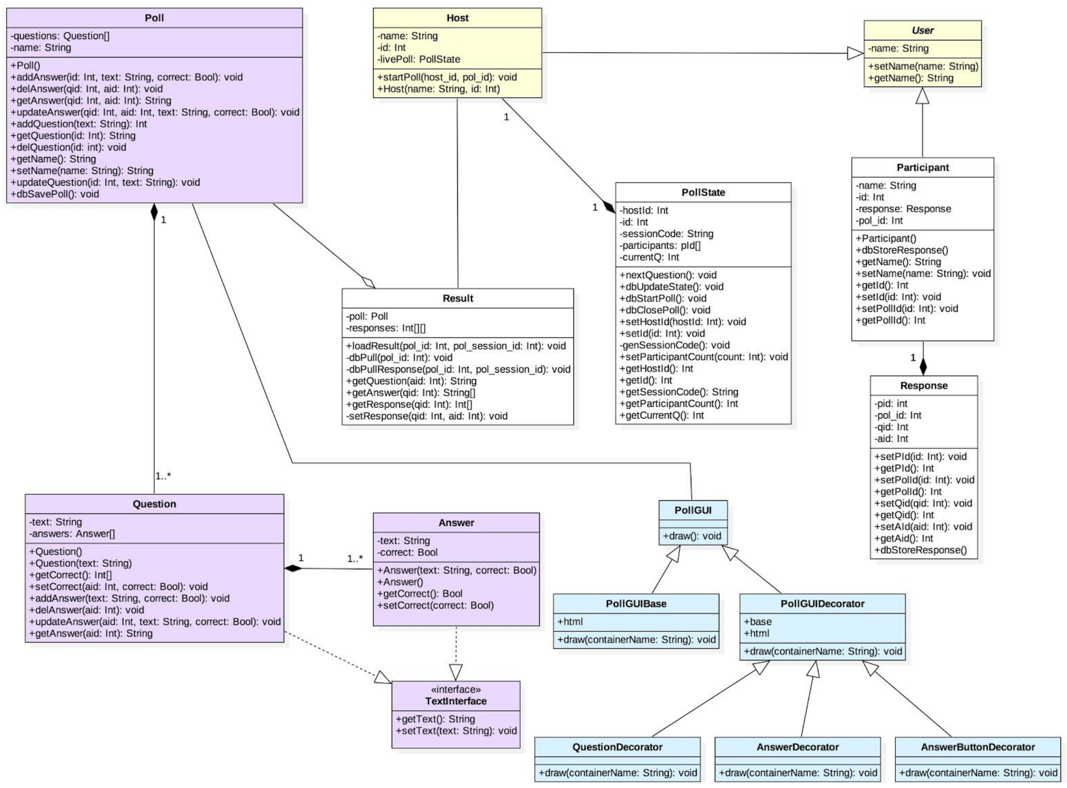
US-07	As a *host* I want to **invite participants** to my poll so that they know my poll exists	2

4. Show your Part 2 class diagram and your final class diagram.

Final:


Poll
-questions: Question[]
-name: String
+Poll()
+addAnswer(id: Int, text: String, correct: Bool): void
+delAnswer(qid: Int, aid: Int): void
+getAnswer(qid: Int, aid: Int): String
+updateAnswer(qid: Int, aid: Int, text: String, correct: Bool): void
+addQuestion(text: String): Int
+getQuestion(id: Int): String
+delQuestion(id: int): void
+getName(): String
+setName(name: String): String
+updateQuestion(id: Int, text: String): void
+dbSavePoll(): void
+setQId(ind: Int, id: Int): void
+getQId(ind: Int): Int
+setAId(q_ind: Int, a_ind: Int, id: Int): void
+getAId(q_ind: Int, a_ind: Int): Int
+getNumQuestion(): Int
+setCorrect(q_ind: Int, a_ind: Int): void
+getCorrect(): Array Booleans
+getAnswers(q_ind: Int): Array Answers
Host
-name: String
-id: Int
+startPoll(host_id, pol_id): void
+Host(name: String, id: Int)
User
-name: String
+setName(name: String)
+getName(): String
Participant
-name: String
-id: Int
-response: Response
-pol_id: Int
+Participant()
+getName(): String
+setName(name: String): void
+getId(): Int
+setId(id: Int): void
+setPollCode(id: Int): void
+getPollCode(): Int
+setResponse(qoid: Int): Int
+getResponse(qoid): Bool
Response
-pid: int
-pol_id: Int
-qid: Int
-aid: Int
+setPId(id: Int): void
+getPId(): Int
+setPollId(id: Int): void
+getPollId(): Int
+setQid(qid: Int): void
+getQid(): Int
+setAId(aid: Int): void
+getAid(): Int
+dbStoreResponse()
Result
-poll: Poll
-responses: Int[][]
+getQuestion(aid: Int): String
+getAnswer(qid: Int): String[]
+getResponse(): Int
-setResponse(): void
+getResult(): Result Object
+getPollCode(): Int
1
1..*
Question
-text: String
-answers: Answer[]
+Question()
+Question(text: String)
+getCorrect(): Int
+setCorrect(aid: Int, correct: Bool): void
+addAnswer(text: String, correct: Bool): void
+delAnswer(aid: Int): void
+updateAnswer(aid: Int, text: String, correct: Bool): void
+getAnswer(aid: Int): String
+setId(id: Int): void
+getId(): Int
+setAId(ind: Int, id: Int): void
+getAId(ind: Int): Int
+Operation1()
1..*
1
Answer
-text: String
-correct: Bool
+Answer(text: String, correct: Bool)
+Answer()
+getCorrect(): Bool
+setCorrect(correct: Bool)
+setId(id: Int): void
+getId(): Int
«interface»
TextInterface
+getText(): String
+setText(text: String): void
PollGUI
+draw(): void
PollGUIBase
+html
+draw(containerName: String): void
PollGUIDecorator
+base
+html
+draw(containerName: String): void
QuestionDecorator
+draw(base PollGUI, text String, q_ind Int): void
AnswerDecorator
+draw(base PollGUI, q_ind Int, a_ind Int, text String, correct Bool): void
AnswerButtonDecorator
+draw(base PollGUI, q_ind Int): void

Part 2:

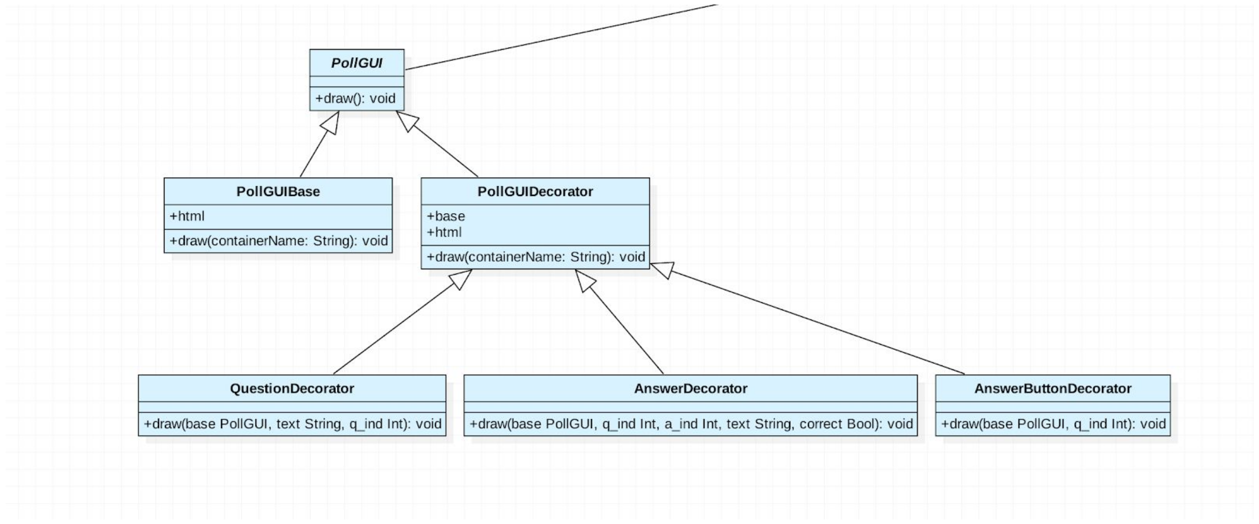

Poll
-questions: Question[]
-name: String
+Poll()
+addAnswer(id: Int, text: String, correct: Bool): void
+delAnswer(qid: Int, aid: Int): void
+getAnswer(qid: Int, aid: Int): String
+updateAnswer(qid: Int, aid: Int, text: String, correct: Bool): void
+addQuestion(text: String): Int
+getQuestion(id: Int): String
+delQuestion(id: int): void
+getName(): String
+setName(name: String): String
+updateQuestion(id: Int, text: String): void
+dbSavePoll(): void
Host
-name: String
-id: Int
-livePoll: PollState
+startPoll(host_id, pol_id): void
+Host(name: String, id: Int)
User
-name: String
+setName(name: String)
+getName(): String
1
1
PollState
-hostId: Int
-id: Int
-sessionCode: String
-participants: pId[]
-currentQ: Int
+nextQuestion(): void
+dbUpdateState(): void
+dbStartPoll(): void
+dbClosePoll(): void
+setHostId(hostId: Int): void
+setId(id: Int): void
-genSessionCode(): void
+setParticipantCount(count: Int): void
+getHostId(): Int
+getId(): Int
+getSessionCode(): String
+getParticipantCount(): Int
+getCurrentQ(): Int
Participant
-name: String
-id: Int
-response: Response
-pol_id: Int
+Participant()
+dbStoreResponse()
+getName(): String
+setName(name: String): void
+getId(): Int
+setId(id: Int): void
+setPollId(id: Int): void
+getPollId(): Int
1
Response
-pid: int
-pol_id: Int
-qid: Int
-aid: Int
+setPId(id: Int): void
+getPId(): Int
+setPollId(id: Int): void
+getPollId(): Int
+setQid(qid: Int): void
+getQid(): Int
+setAId(aid: Int): void
+getAid(): Int
+dbStoreResponse()
Result
-poll: Poll
-responses: Int[][]
+loadResult(pol_id: Int, pol_session_id: Int): void
-dbPull(pol_id: Int): void
-dbPullResponse(pol_id: Int, pol_session_id): void
+getQuestion(aid: Int): String
+getAnswer(qid: Int): String[]
+getResponse(qid: Int): Int[]
-setResponse(qid: Int, aid: Int): void
1
1..*
Question
-text: String
-answers: Answer[]
+Question()
+Question(text: String)
+getCorrect(): Int[]
+setCorrect(aid: Int, correct: Bool): void
+addAnswer(text: String, correct: Bool): void
+delAnswer(aid: Int): void
+updateAnswer(aid: Int, text: String, correct: Bool): void
+getAnswer(aid: Int): String
1
1..*
Answer
-text: String
-correct: Bool
+Answer(text: String, correct: Bool)
+Answer()
+getCorrect(): Bool
+setCorrect(correct: Bool)
«interface»
TextInterface
+getText(): String
+setText(text: String): void
PollGUI
+draw(): void
PollGUIBase
+html
+draw(containerName: String): void
PollGUIDecorator
+base
+html
+draw(containerName: String): void
QuestionDecorator
+draw(containerName: String): void
AnswerDecorator
+draw(containerName: String): void
AnswerButtonDecorator
+draw(containerName: String): void

(4 continued) What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

- One of the main things that changed in our class diagram was the removal of the PollSate class. We realized that the PollState was an Anti-Pattern Blob class. It had functionality that was redundant.
- Another thing that changed was that we were able to remove any of the dbPull and dbPush methods from our classes which had them. This was because the Django framework implemented all methods that interact with the database for us. There for we were able to implement our classes which called the Django model db functions in them.
- Lastly, one thing that changed a small amount was the Poll, and Question Classes. These changed due to needing a few more methods that we didn't think of earlier on.

5. Show the classes from your class diagram that implement each design pattern.

Decorator


PollGUI
+draw(): void
PollGUIBase
+html
+draw(containerName: String): void
PollGUIDecorator
+base
+html
+draw(containerName: String): void
QuestionDecorator
+draw(base PollGUI, text String, q_ind Int): void
AnswerDecorator
+draw(base PollGUI, q_ind Int, a_ind Int, text String, correct Bool): void
AnswerButtonDecorator
+draw(base PollGUI, q_ind Int): void

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?**

   From going through the process of creating, designing, and implementing a system, our team learned several lessons. One thing that was initially very apparent was the huge benefit you reap when you create a well thought-out architecture model from the get go. The UML use-case diagrams, sequence diagrams, activity diagrams, and class diagrams gave us a lot of structure when we began to start writing code and gave us a very clear direction. Specifically, thinking through the details of our class diagram before we began implementing the system forced us to make several design decisions up front, rather than having to face them once we had already started to implement it. With proper UML diagrams in place, it was rather straightforward to map them to functional classes and code. Overall, he better the architecture, the easier it is to write the corresponding code.

   Another big lesson we learned was the importance of design patterns. Without the knowledge of design patterns like decorator, our code likely would’ve been very repetitive and we wouldn’t have been able to implement the poll creation functionality as easily, and thus would have had a harder time accomplishing the main purpose of our app. The design patterns were created to improve the designs of systems, and it was worth asking ourselves how we could incorporate them into our system to improve its object-oriented design and performance.