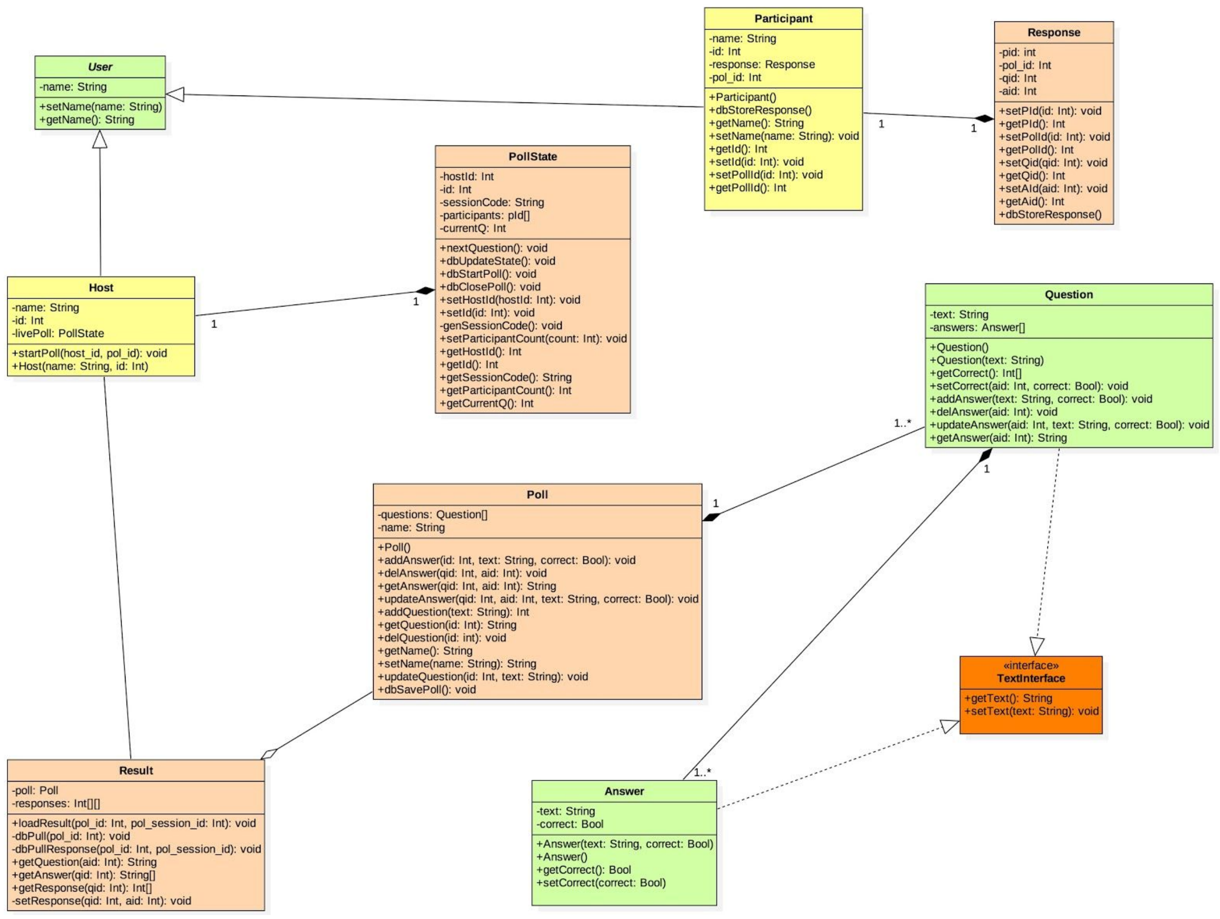
Team:

- Lauren Raddatz
- Brandon Aguirre
- Marco Ortiz Torres

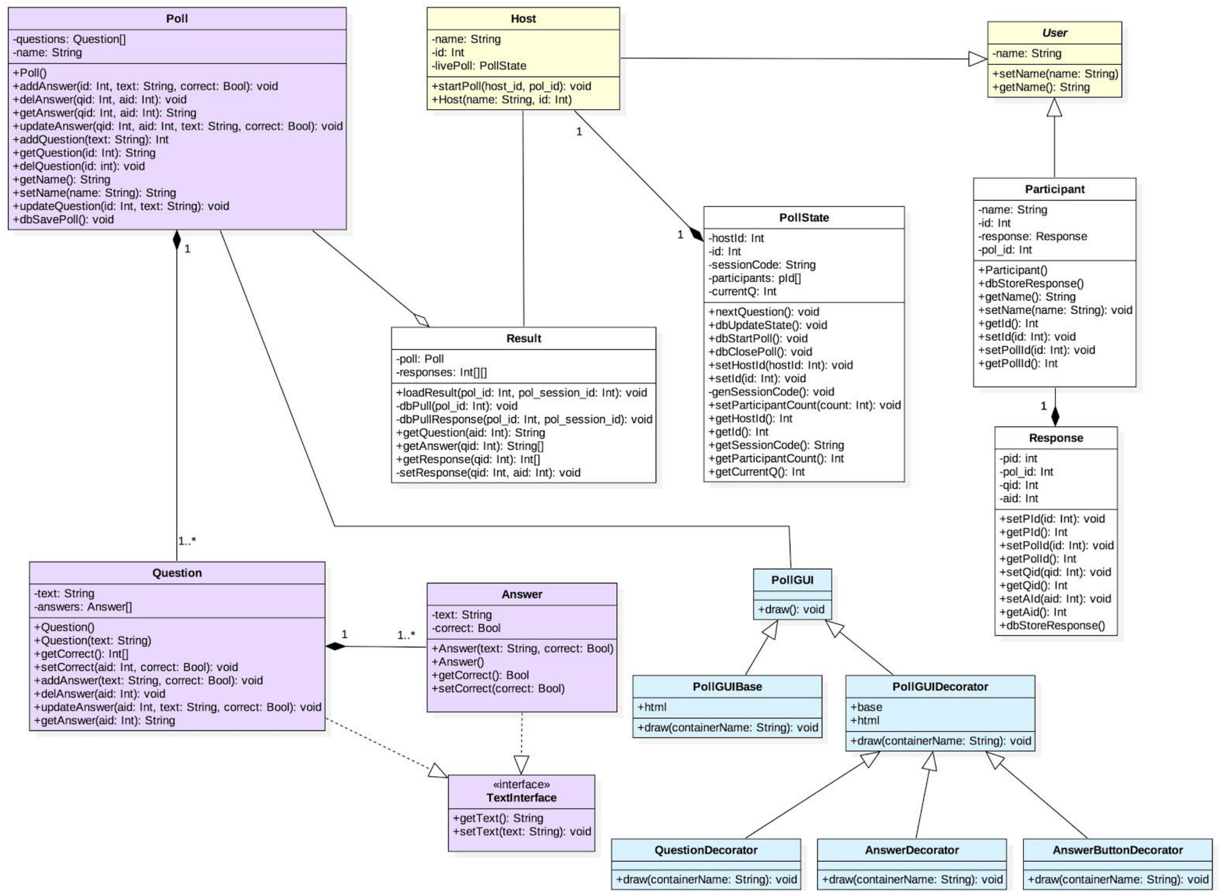
Vision: Create a virtual polling system to replace iClickers.

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).

Previous Class Diagram:


User
-name: String
+setName(name: String)
+getName(): String
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
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()
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
Host
-name: String
-id: Int
-livePoll: PollState
+startPoll(host_id, pol_id): void
+Host(name: String, id: 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
1..*
1
1
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
«interface»
TextInterface
+getText(): String
+setText(text: String): void
1..*
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
Answer
-text: String
-correct: Bool
+Answer(text: String, correct: Bool)
+Answer()
+getCorrect(): Bool
+setCorrect(correct: Bool)

Completed Class diagram:


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
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
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
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
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()
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
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
1
1
1
1..*
1
1..*
1
1

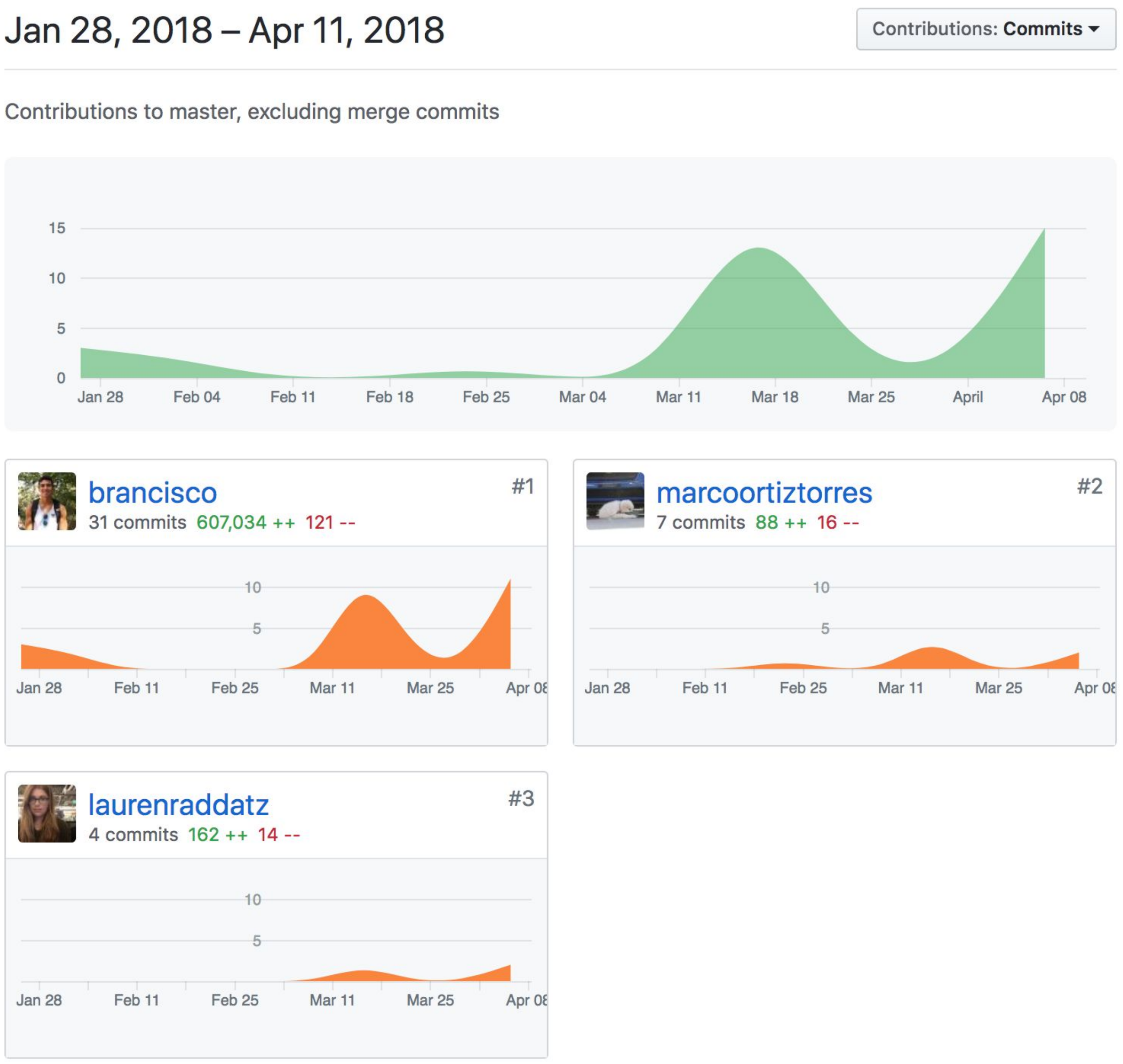
Breakdown of Work:

Brandon - Built out the poll creator functionality. Used Javascript to implement classes on the front end including Poll, Question, Answer, TextInterface, PollGUI, PollGUIBase, PollGUIDecorator, QuestionDectorator, AnswerDecorator, AnswerButonDecorator. The interface now allows users to make the poll by adding questions and answers, and saving to the database.

Lauren - Implementing the host dashboard.

Marco - Set up the join poll functionality and populating the participant's screen with the current question of the session.

GitHub Graph:



Jan 28, 2018 – Apr 11, 2018

Contributions: **Commits**

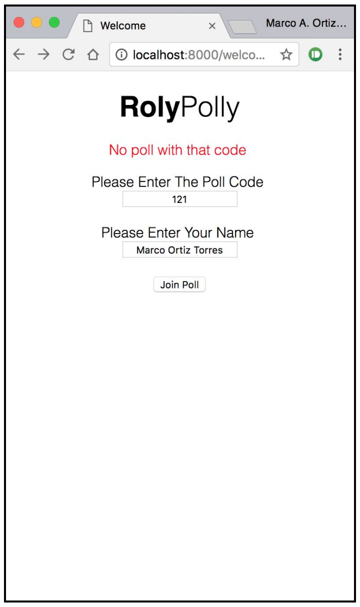
Contributions to master, excluding merge commits

brancisco #1
31 commits 607,034 ++ 121 --

marcoortiztorres #2
7 commits 88 ++ 16 --

laurenraddatz #3
4 commits 162 ++ 14 --

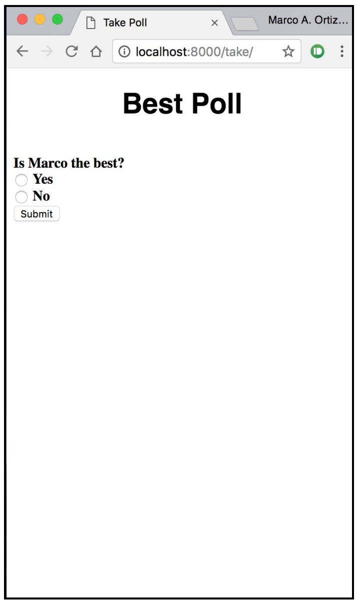
Estimate Remaining Effort:



No poll with that code

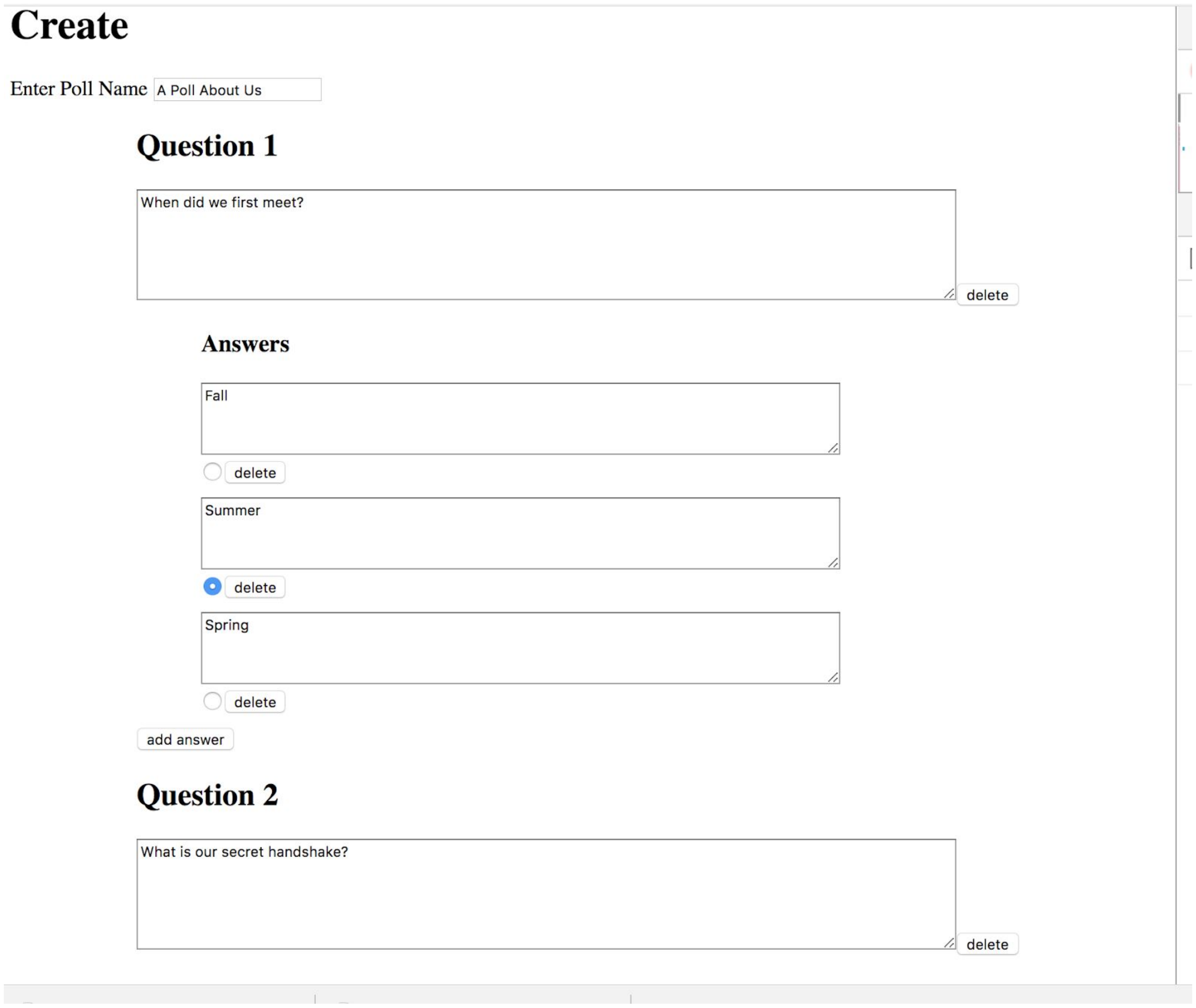
Please Enter The Poll Code
121

Please Enter Your Name
Marco Ortiz Torres



Is Marco the best?
Yes
No

In the images above you can see the participant is able to enter a code and their name. If the participant enters a poll code that is not in the database, then an error will appear that the poll does not exist. Once a correct code is entered, the participant will be redirected to the poll with the current live question. The screen on the right is pulling questions and answers from the database. What is left to do is to record the participants response and have the page to update when a new question is answered.


Create

Enter Poll Name A Poll About Us

Question 1

When did we first meet?

Answers

Fall

Summer

Spring

Question 2

What is our secret handshake?

Here you can see the poll creator. At the top you can name the poll. There is an option just off the screen on the bottom to add new questions, and a button after each set of answers to add an answer. We can also select the correct answer by choosing a radio button under each answer.

Design Patterns:

- **Decorator** - the main Design pattern we implemented was the Decorator design pattern in for our poll creator GUI. **In our completed class diagram, this is represented by the light blue classes.** The section we have has three decorators, one that decorates a base of html with a question, one that decorates with an answer, and one with an answer button. As the User creates their poll, the GUI is refreshed, and the decorator draws to the browser the html which is all added together through two forloops. Outer for loop calls the question decorator, and the inner for loop calls the inner forloop.

- **Composite** - our second design pattern is the Composite design for our poll hierarchy. **In our completed class diagram, this is represented by the purple classes.** Where Poll has a 1 to many relationship with Question and Question has a 1 to many relationship with Answer. Due to these relationship, we found it best to make poll a Composite of Questions and Question a composite of Answers.

Final Iteration:

For our final iteration we plan to complete the most important features such as launching a poll as a host user, and taking a poll as a participant user. We have some refactoring to do, due to the way the the django model has added some classes to what we had. This will move some of our functionality away from classes we planned on implementing over to the django defaults.