Software Requirements Specification Document

easyPoll

1.0
December 1, 2016

Team 3

Nick Frichette, Nick Messina, Casey Cook, Kevin Dalle, Matt Fletcher

Submitted in partial fulfillment of the requirements of
IT 326 Software Engineering
Fall 2016

Table of Contents

Revision History

Date	Description	Author	Comments
9/16/2016	First changes	All	First changes
10/20/2016	Edit	All	Add sprint 1 and change comments
10/26/2016	Edit	All	Changed Everything
11/15/2016	Edit	All	Changed Everything
12/1/2016	Final Edits	All	Final Sprint

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Casey Cook		11/1/16
	Dr. Rishi Kanth Saripalle		
	Kevin Dalle		11/1/16
	Matthew Fletcher		11/1/16
	Nick Frichette		11/15/16
	Nick Frichette		12/1/2016

1. Introduction

easyPoll is a live polling application that allows users to create their own polls, and allow both anonymous and registered users to participate in their polls. Users are limited to one vote per poll, and can view the statistics of a poll after voting in it.

1.1 Purpose

The purpose of this SRS document is to define the project scope, common terms pertaining to the application, references, and give a general overview of the functional requirements.

1.2 Scope

1. We are creating a software application called "easyPoll".
2. This application is created to help users create polls, take polls, and display polls in real time. The point of this application is NOT to be a social media platform.
3. The benefits of this software is the ease of creating new polls and specifying exactly who you want or don't want to participate in your poll. Our objectives are to create an application that makes it easy for the user to create and participate in polls, find open public polls that align with their interests and view data pertaining to those polls, and export the data from the polls the user created. Our goal is to help create a network of data-loving users, curious about a plethora of subjects, and the ability to obtain data on whatever questions they have.
4. This application will require a web browser, a web server, and a database.

1.3 Definitions, Acronyms, and Abbreviations

Poll:	Question with two or more selectable answers posted by registered user.
easyPoll:	Web Polling Application
Registered User:	User with account
Anonymous User:	User without registered account

1.5 Overview

1. This document contains an overview of all information relating to the system for the "easyPoll Application". It contains all the information on how the software application should be designed and constructed.
2. General Information
    a. Product Perspective
    b. User Characteristics

2. General Description

easyPoll allows users to participate in polls, view poll statistics, and create and customized polls. In order to participate in private polls, and create and customize polls, a user must create an account as a registered user. In order to view poll statistics, a user must first vote in that poll.

2.1 Product Perspective

easyPoll is a unique take on the general "polling" application. There are several competitors in the market including DirectPoll and Poll Everywhere. However these competitors are specialized in unique areas, where DirectPoll is more geared towards education and Poll Everywhere is designed to suite high end business needs. easyPoll will be a powerful platform that will provide anyone the ability to dynamically create, and distribute custom polls and questions.

2.2 User Characteristics

Targeted users have a general understanding of how to navigate a website and how polling systems work. They have resources to be able to go to a website and become a registered user. Users need to understand basic English.

2.3 System Environment

Our application will run entirely in a virtual environment. easyPoll was developed with the Spring Tool Suite. When deployed it will run on a virtual server called Pivotal. The user will then access the website from a browser on their localhost. The virtual server will host all services and resources.

2.4 General Constraints

- The amount of web traffic is limited by the host server's capabilities
- Backend Language: Java
- Front End Languages: HTML, CSS, JavaScript

2.5 Assumptions and Dependencies

All users interacting with the system will need to know English. We depend on database connections, server connections, and quality internet connections at all times.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The application will be used on the following browsers: Firefox version 49.0, Chrome version 54.0.2840.59, Safari version 10.0.

3.1.2 Hardware Interfaces

There are no hardware interfaces.

3.1.3 Software Interfaces

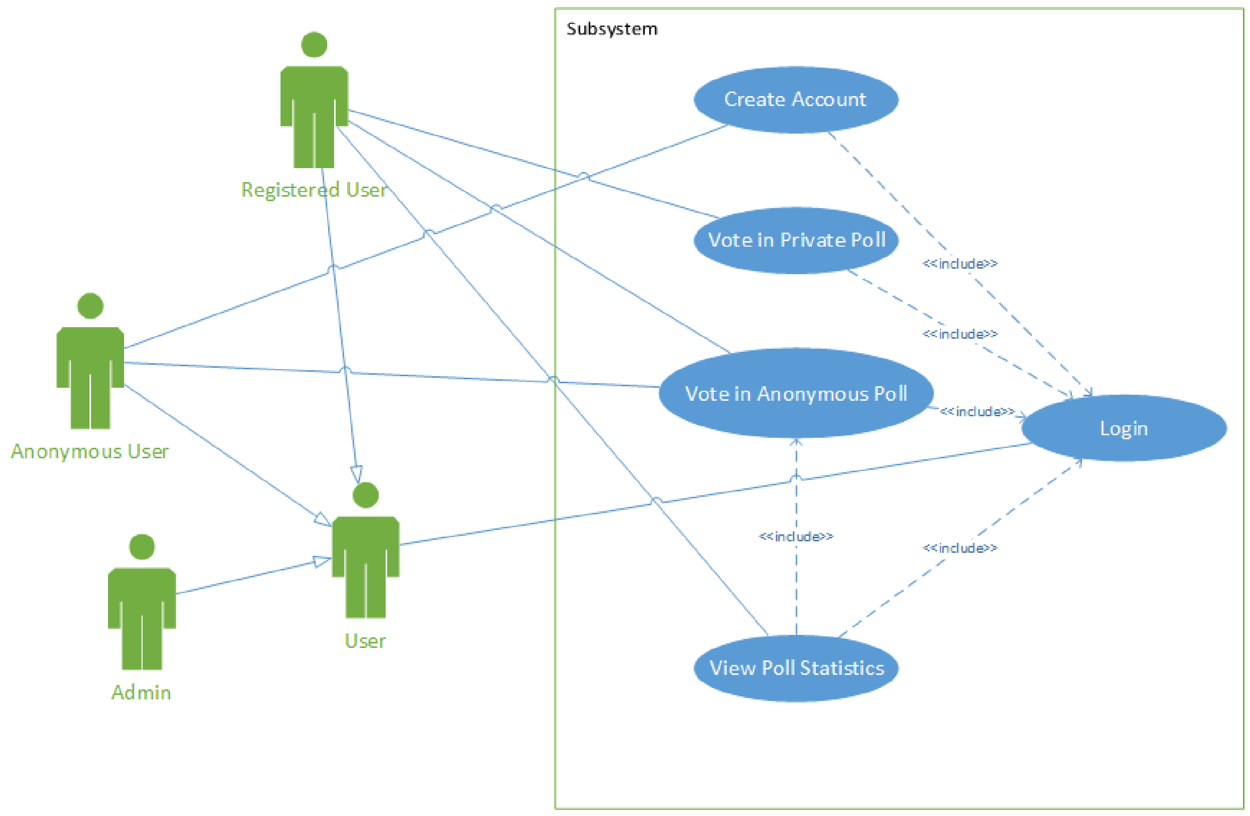
Spring MVC, JDBC

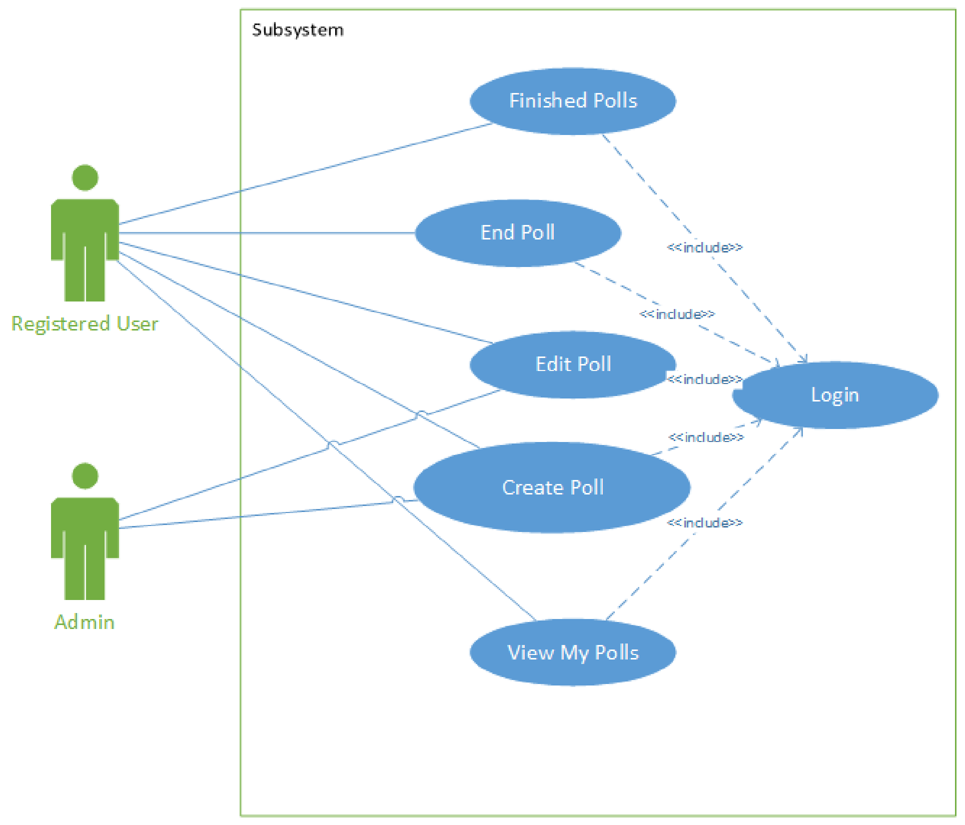
3.1.4 Communications Interfaces

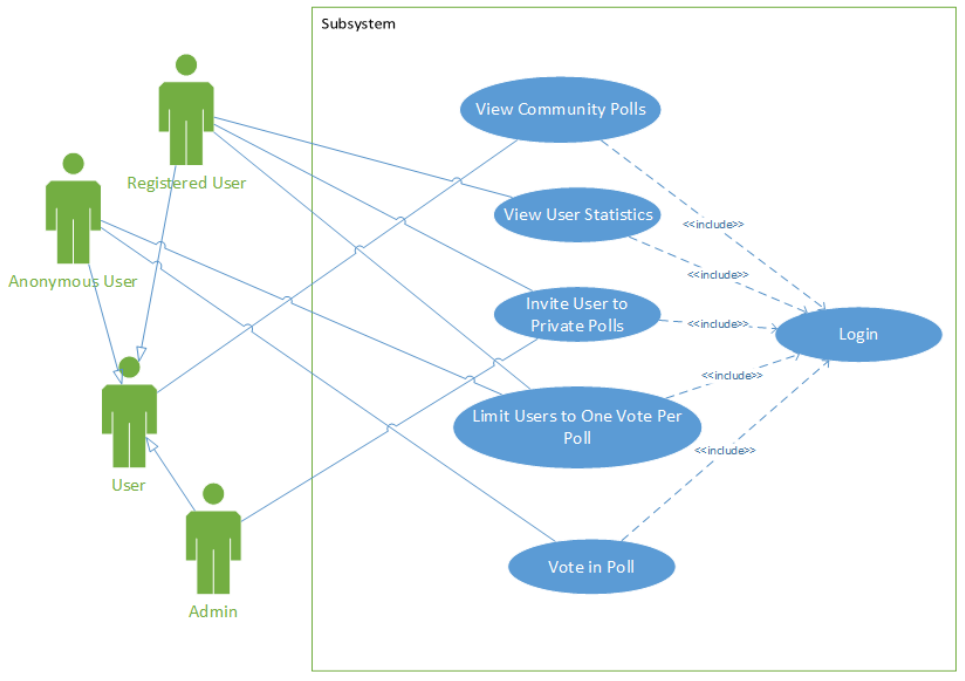
There are no communications interfaces

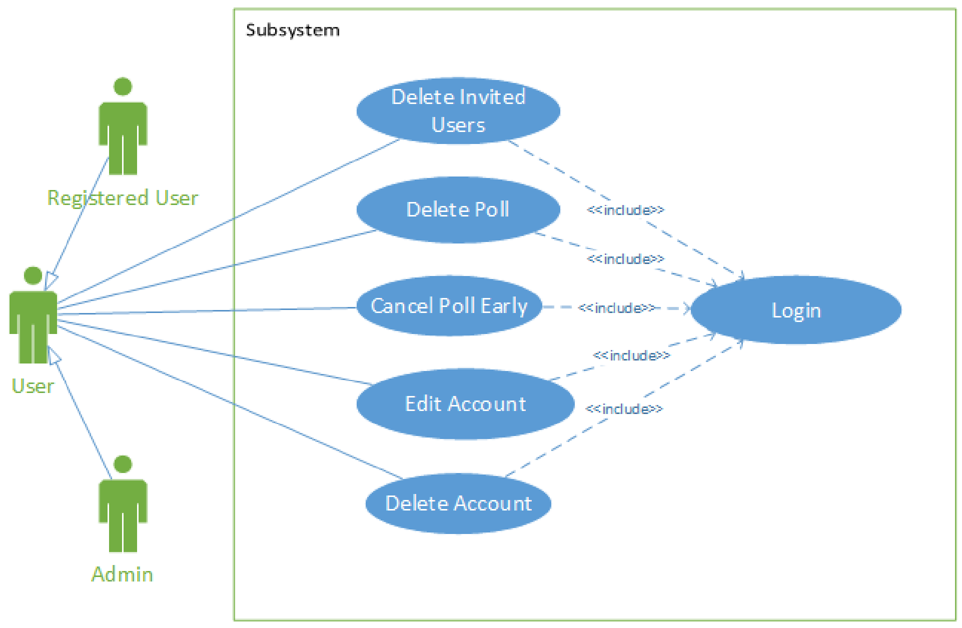
3.2 Functional Requirements

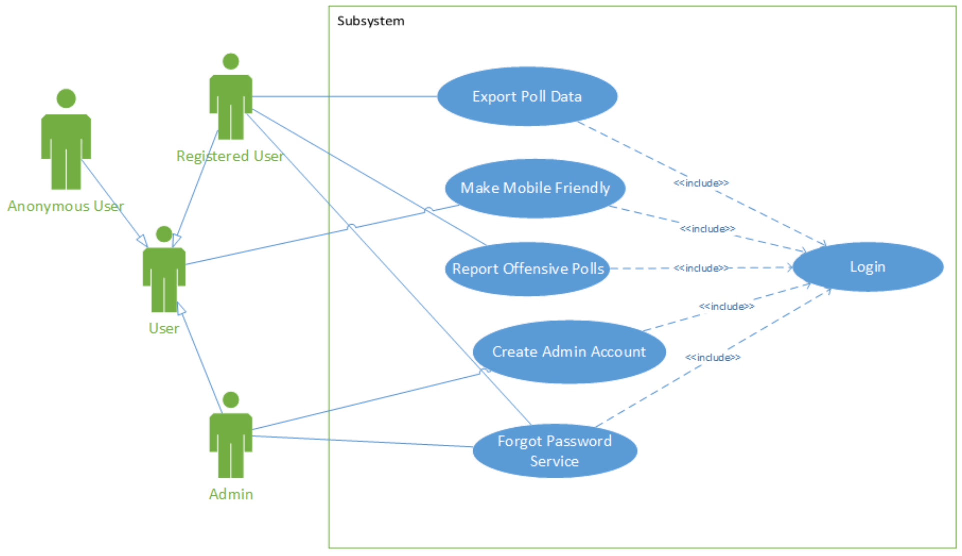
This section describes specific features of the software project.

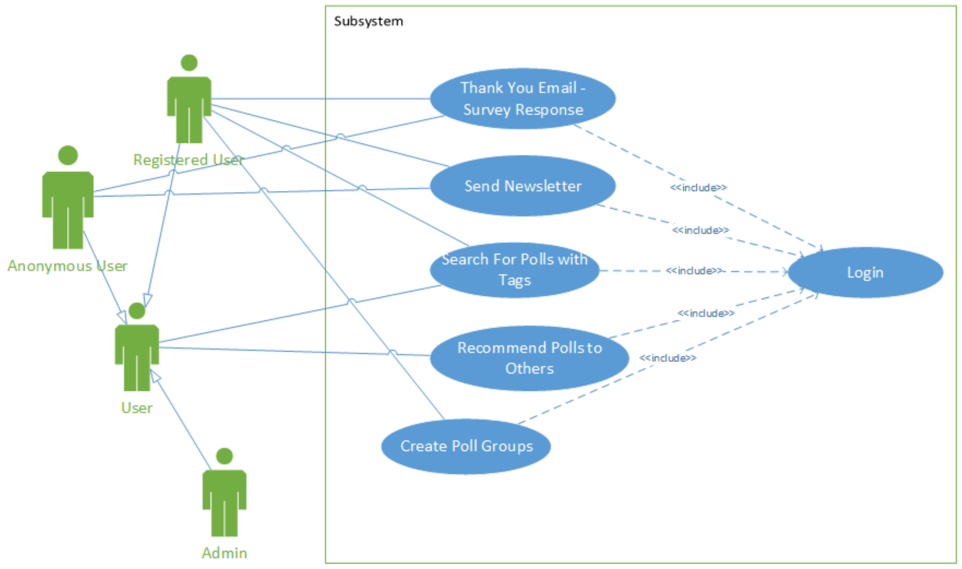


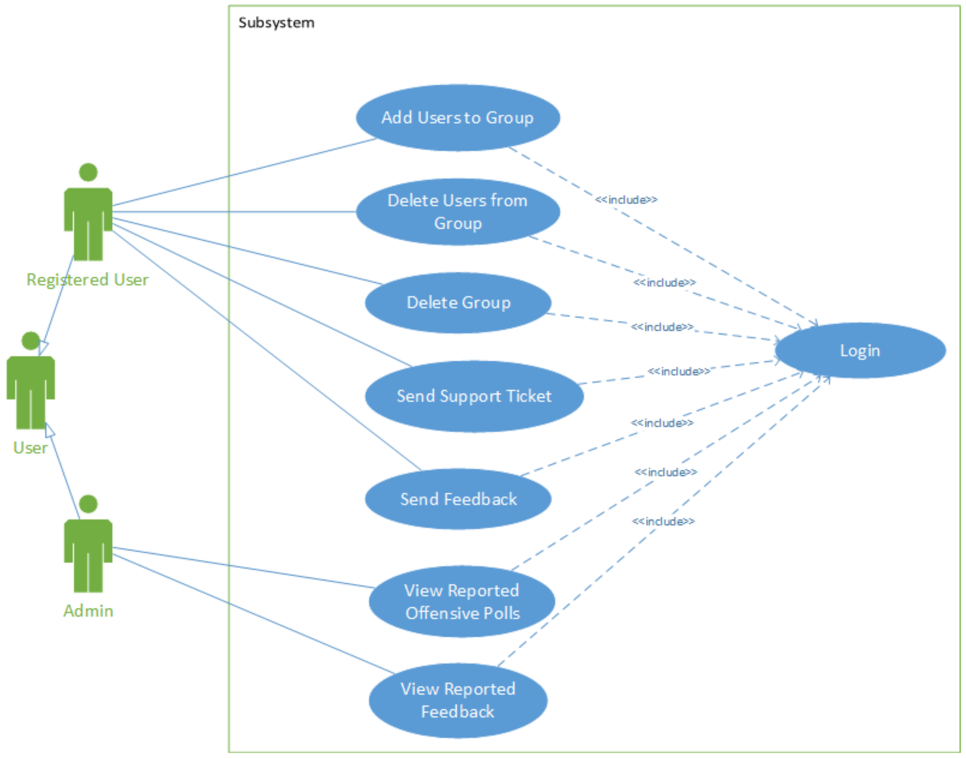

Subsystem
Finished Polls
End Poll
Edit Poll
Create Poll
View My Polls
Login
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
Registered User
Admin


Subsystem
View Community Polls
View User Statistics
Invite User to Private Polls
Limit Users to One Vote Per Poll
Vote in Poll
Login
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
Registered User
Anonymous User
User
Admin


Subsystem
Registered User
User
Admin
Delete Invited Users
Delete Poll
Cancel Poll Early
Edit Account
Delete Account
Login
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>


Subsystem
Anonymous User
Registered User
User
Admin
Export Poll Data
Make Mobile Friendly
Report Offensive Polls
Create Admin Account
Forgot Password Service
Login
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>





3.2.1 Create Account

1. Description: The user will need to be able to create an account with the website. They will do this by providing a username, password, and email. Once they have created their account, the system will track that user.
2. Actor(s): Anonymous User

3. Conditions:
   3.2.1.4.1 Pre: User does not have an account.
   3.2.1.4.2 Post: An account will now be in the system.
4. Steps or Sequence: User goes to website. Click, sign up. User enters email address, username and password. User clicks submit.

3.2.2 Login

1. Description: The user will need to be able to login to their account on the website. They will do this by supplying their email address and password they set up during their account creation.
2. Actor(s): Registered User
3. Conditions:
   3.2.2.4.1 Pre: The user is not logged in yet.
   3.2.2.4.2 Post: The user will be logged in.
4. Steps or Sequence: User goes to website. Click, login, User enters email address and password. User clicks login.

3.2.3 Anonymously Vote in Poll

1. Description: The user will be able to vote in an anonymous poll without an account.
2. Actor(s): Anonymous User
3. Conditions:
   3.2.3.4.1 Pre: User has not voted in poll.
   3.2.3.4.2 Post: The poll will have an additional anonymous vote
4. Steps or Sequence: User goes to website. User goes to a poll. Makes selection on poll. Submit.

3.2.4 Vote in Private Poll

1. Description: The user will be able to vote in a private poll by having an account with the right privileges.
2. Actor(s): Registered User
3. Condition:
   3.2.4.4.1 Pre: User has not voted in poll
   3.2.4.4.2 Post: The poll will have an additional private vote
4. Steps or Sequence: User goes to website. User logs in. User goes to their polls. User votes in poll.

3.2.5 View Poll Statistics

1. Description: The user will be able to see the results of the poll.
2. Actor(s): Registered User
3. Condition:
   3.2.5.4.1 Pre: The user has participated in multiple aspects of the site
   3.2.5.4.2 Post:
4. Steps or Sequence: User goes to website. Logs in. Goes to the poll of choice after it has been completed.

3.2.6 View Finished Polls

1. Description: The user will be able to see a poll after it has been completed
2. Actor(s): Registered User
3. Condition:
   3.2.6.4.1 Pre: A poll has been finished
   3.2.6.4.2 Post:

4. Steps or Sequence: User goes to website. Logs in. Click on my polls. Click on any completed polls.

3.2.7 End Poll

1. Description: The user will be able to define when their poll should end.
2. Actor(s): Registered User
3. Condition:
   3.2.7.4.1 Pre: The poll that the user is specifying is currently running
   3.2.7.4.2 Post: The specified poll will end when the user has specified
4. Steps or Sequence: During poll creation the user will specify when the poll should end. This value can be modified.

3.2.8 Edit Poll

1. Description: The user will be able to modify an existing poll
2. Actor(s): Registered User
3. Condition:
   3.2.8.4.1 Pre: A poll was created but not modified
   3.2.8.4.2 Post: A poll has been modified
4. Steps or Sequence: User logs in. Goes to their polls. Selects the poll they would like to modify. Changes the parameters of the poll.

3.2.9 Create Poll

1. Description: The user will be able to create a poll
2. Actor(s): Registered User
3. Condition:
   3.2.9.4.1 Pre: A poll does not exist
   3.2.9.4.2 Post: A poll now exists in the system
4. Steps or Sequence: User logs in. Goes to their polls. Clicks on create poll. Enters the poll parameters.

3.2.10 View My Polls

1. Description: The registered user can view the polls they have created.
2. Actor(s): Registered User
3. Condition:
   3.2.10.4.1 Pre:
   3.2.10.4.2 Post:
4. Steps or Sequence: User logs in. User goes to My Polls.

3.2.11 View Community Polls

1. Description: The user can view polls posted by the community and click to go vote on them.
2. Actor(s): User
3. Condition:
   3.2.10.4.1 Pre:
   3.2.10.4.2 Post:
4. Steps or Sequence: User logs in. User click on community polls tab.

3.2.12 View User Statistics

1. Description: The user can view their account statistics
2. Actor(s): Registered User
3. Condition:
   3.2.12.4.1 Pre: User has an account

3.2.12.4.2 Post:

4. Steps or Sequence: User logs in. Clicks on their account name. Views user statistics.

3.2.13 Invite User to Private Polls

1. Description: The user can invite existing users to private polls.
2. Actor(s): Registered User
3. Condition:
   3.2.13.4.1 Pre: User has an account
   3.2.13.4.2 Post: Registered users have access to the private poll they were invited to. Private poll shows up in the invited users My Polls.
4. Steps or Sequence: User logs in. User goes to their groups. Selects a group. Invites user.

3.2.14 Limit Users to One Vote Per Poll

1. Description: The user can only vote in a particular pole once
2. Actor(s): Registered User and Anonymous User
3. Condition:
   3.2.14.4.1 Pre: User has voted in a poll
   3.2.14.4.2 Post: User is denied the ability to vote in the pole again
4. Steps or Sequence: User goes to website. Clicks on poll. User selects their choice and then hits submit.

3.2.15 Vote in Poll

1. Description: The user can vote in a poll.
2. Actor(s): Registered User
3. Condition:
   3.2.12.4.1 Pre: Poll exists already
   3.2.12.4.2 Post: Poll has additional vote
4. Steps or Sequence: User logs in. User goes to a poll. User makes a selection and clicks submit.

3.2.16 Delete Invited Users

1. Description: The creator of a poll can delete invited users from a private poll
2. Actor(s): Registered User
3. Condition:
   3.2.12.4.1 Pre: User has an account and is the creator of the poll
   3.2.12.4.2 Post: Deleted user can no longer vote in the poll
4. Steps or Sequence: The user goes to website. Click login. User enters email address and password to login in. Navigates to the poll group in question. Removes the invited user.

3.2.17 Delete Poll

1. Description: The user will be able to delete an already existing poll. This will mean that the results will also be deleted
2. Actor(s): Registered User, Admin
3. Conditions:
   3.2.17.4.1 Pre: The user is logged in and the creator of the poll.
   3.2.17.4.2 Post: The poll is deleted.
4. Steps or Sequence: The user logs in. Goes to my polls. Click poll they want deleted. Click delete poll button.

3.2.18 Cancel Poll Early

1. Description: The user will be able to cancel a poll before it is supposed to have ended.
2. Actor(s): Registered User

3. Condition:
   3.2.18.4.1 Pre: The poll is ongoing
   3.2.18.4.2 Post: The poll has been canceled
4. Steps or Sequence: The user logs in. The user goes to their polls. The use clicks cancel.

3.2.19 Edit Account

1. Description: The user will be able to edit their personal account information
2. Actor(s): Registered User
3. Condition:
   3.2.19.4.1 Pre: The users account with be some values
   3.2.19.4.2 Post: The users account will hold different values
4. Steps or Sequence: The user logs in. The user goes to their profile. The user modifies their settings.

3.2.20 Delete Account

1. Description: The user will be able to delete their account and the system will remove their information.
2. Actors(s): Registered User, Admin
3. Condition:
   3.2.20.4.1 Pre: The user has an account
   3.2.20.4.2 Post: The user account has been deleted
4. Steps or Sequence: The user logs in. The user goes to their profile. The users clicks, "Delete My Account"

3.2.21 Export Poll Data

1. Description: The user will be able to export their poll data to a csv file
2. Actor(s): Registered User
3. Condition:
   3.2.21.4.1 Pre:
   3.2.21.4.2 Post: The user will have downloaded the data
4. Steps or Sequence: User logs in. User goes to their polls. The user selects a poll they would like to access. The user clicks "Export Poll Data".

3.2.22 Make Mobile Friendly

1. Description: The user will be able to access the service effectively on a mobile device.
2. Actor(s): All Users
3. Condition:
   3.2.22.4.1 Pre: The user access the system from a mobile device
   3.2.22.4.2 Post: The user can view and interact with the mobile site.
4. Steps or Sequence: User accesses the website from a mobile device.

3.2.23 Report Offensive Polls

1. Description: The user will be able to report offensive polls to an admin.
2. Actor(s): Registered User
3. Condition:
   3.2.23.4.1 Pre: The user has an account
   3.2.23.4.2 Post: The poll has been reported to an admin as offensive
4. Steps or Sequence: User logs in. User goes to their polls. User selects poll. User click "Report Offensive Poll".

3.2.24 Create Admin Account

1. Description: The user will be able to become an admin if they have the right secret password.
2. Actor(s): Registered User
3. Condition:
   3.2.24.4.1 Pre:
   3.2.24.4.2 Post: The user will have an admin account created for them.
4. Steps or Sequence: The user logs in. The user goes to their profile. The user modifies their settings.

3.2.25 Forgot Password Service

1. Description: The user can reset their password and be emailed a temporary password.
2. Actor(s): Registered user, Admin
3. Condition:
   3.2.25.4.1 Pre: User has an account.
   3.2.25.4.2 Post: Users password is set to a temporary password.
4. Steps or Sequence: User goes to login page. Clicks "Forgot My Password" button. System sent email with temporary password.

3.2.26 Thank You Email - Survey Response

1. Description: The system will send a thank you email to the user who voted.
2. Actor(s): Registered User and Anonymous User.
3. Condition:
   3.2.26.4.1 Pre:
   3.2.26.4.2 Post: The user has received an email.
4. Steps or Sequence: User votes in poll. An email wil automatically be sent

3.2.27 Send Newsletter

1. Description: Admin's have the ability to send an email to every member on the site.
2. Actor(s): Admin
3. Condition:
   3.2.27.4.1 Pre:
   3.2.27.4.2 Post: All registered users receive an email written by the admin.
4. Steps or Sequence: Admin has an account. Admin goes to the Admin portal. User writes out the newsletter. Newsletter is sent.

3.2.28 View Support Ticket

1. Description: Admins will be able to see what Support Tickets they have received.
2. Actor(s): Admin
3. Condition:
   3.2.28.4.1 Pre: There are Tickets stored in the DB
   3.2.28.4.2 Post: That ticket information is presented to the Admin
4. Steps or Sequence: Admin logs in. Admin goes to View Support Ticket section

3.2.29 Recommend Polls to Others

1. Description: Be able to recommend polls to others.
2. Actor(s): Registered User.
3. Condition:
   3.2.29.4.1 Pre: Poll exists.
   3.2.29.4.2 Post: Email is sent with a link to the poll.
4. Steps or Sequence: User logs in. User goes to poll. User inputs the email they would like to send to.

3.2.30 Create Poll Groups

1. Description: Registered user will be able to create poll groups
2. Actor(s): Registered User
3. Condition:
   3.2.30.4.1 Pre: No poll group exists
   3.2.30.4.2 Post: Poll group exists

4. Steps or Sequence: User logs in. User goes to their groups. User clicks create group, inputs name and poll.

3.2.31 Add Users to Group

1. Description: Registered User adds other Registered Users to an existing poll group
2. Actor(s): Registered User
3. Condition:
   3.2.31.4.1 Pre: User has an account. Poll group is already created.
   3.2.31.4.2 Post: Poll group has one more added to the group
4. Steps or Sequence: User goes to Group Manager page. Clicks on an existing poll group. Clicks on "Add Users" button. User types in username and clicks "Add User"

3.2.32 Delete Users from Group

1. Description: Remove a user from a group.
2. Actor(s): Registered User
3. Condition:
   3.2.32.4.1 Pre: User has an account. Poll group is already created
   3.2.32.4.2 Post: Poll group has one or more users removed from the group.
4. Steps or Sequence: User goes to Group Manager page. Clicks on an existing poll group. The admin of the group will click on the group member they would like to remove.

3.2.33 Delete Group

1. Description: Remove a Group
2. Actor(s): Registered User
3. Condition:
   3.2.33.4.1 Pre: User has an account. Poll group is already created
   3.2.33.4.2 Post: Poll group no longer exists
4. Steps or Sequence: User goes to Group Manager page. Clicks on existing poll group. Clicks on "Delete Group" button.

3.2.34 Send Support Ticket

1. Description: Registered user will be able to send in a ticket for help.
2. Actor(s): Registered User
3. Condition:
   3.2.34.4.1 Pre: User has an account
   3.2.34.4.2 Post: A support ticket will have been submitted for the user

4. Steps or Sequence: User logs in. User encounters a problem. User goes to their account page and click "Send Support Ticket".

3.2.35 Send Feedback

1. Description: Any User can send feedback to the developer
2. Actor(s): Any User
3. Condition:
   3.2.35.4.1 Pre:
   3.2.35.4.2 Post: A feedback ticket will be submitted

4. Steps or Sequence: Any user goes to the home page. Clicks "Send Feedback". The user will type in the feedback they would like to send. User clicks send.

3.2.36 View Reported Offensive Polls

1. Description: Admin can view polls that have been reported as offensive.
2. Actor(s): Admin
3. Condition:
   3.2.36.4.1 Pre: Admin logged in.
   3.2.36.4.2 Post:
4. Steps or Sequence: Admin logs in. Admin goes to the Admin Portal.

3.2.37 View Reported Feedback

1. Description: Admin can view feedback that have been reported.
2. Actor(s): Admin
3. Condition:
   3.2.37.4.1 Pre: Admin logged in.
   3.2.37.4.2 Post:
4. Steps or Sequence: Admin logs in. Admin goes to the Admin Portal.

3.2.38 Poll of the Day

1. Description: On the Community Polls page there will be a section for the "Poll of the Day" which will be the most popular current public poll.
2. Actors(s): System
3. Condition
   3.2.38.4.1 Pre: User goes to Community Polls
   3.2.38.4.2 Post: User can see Poll of the Day
4. Steps or Sequence: When user goes to Community Polls it should dynamically grab the Poll of the Day for them.

3.2.39 Random Poll Button

1. Description: The home page will have a random poll button which will take the user to a random current, public poll.
2. Actors(s): User
3. Condition
   3.2.40.4.1 Pre:
   3.2.40.4.2 Post:
4. Steps or Sequence: User goes to the home page. User clicks on "Vote in a Random Poll". User is taken to a random poll.

3.3 Non-Functional Requirements

3.3.1 **Performance**

- Every page in the application should load quickly
- Changes in poll results will be viewable to all users live with the results refreshing in a reasonable amount of time
- The application should be capable of supporting all active users

3.3.2 Reliability

- The application will accurately display polling results

3.3.3 Availability

- The application should be up and running a majority of the time

3.3.4 Security

- Application users must register for an account to create polls
- Application users can only view statistics of polls that they've voted in
- Only users that are invited to private poll can vote in that poll

3.3.5 Maintainability

- A new version of the application will leave all database contents unchanged

3.3.6 Portability

- The easyPoll application can be easily moved to a live server by packaging the project into a jar file, and uploading it.

3.4 Design Constraints

The entirety of our system will be available using the web for both Computers and mobile devices. Because of this, our application will be designed to respond to different screen sizes to be optimal for any device using our application.

3.5 Other Requirements

4. Design & Development

The purpose of this section is to outline how our team plans to execute the design and development of the "easyPoll" application. It will cover the software process model that we chose to follow and justify it. It will also show a diagram of this model that will provide details to our processes such as our sprint schedule and meeting frequency.

4.1 Software Process Model

4.1.1 Introduction

This section will briefly describes the process model chosen and any background information required to understand the rest of this section.

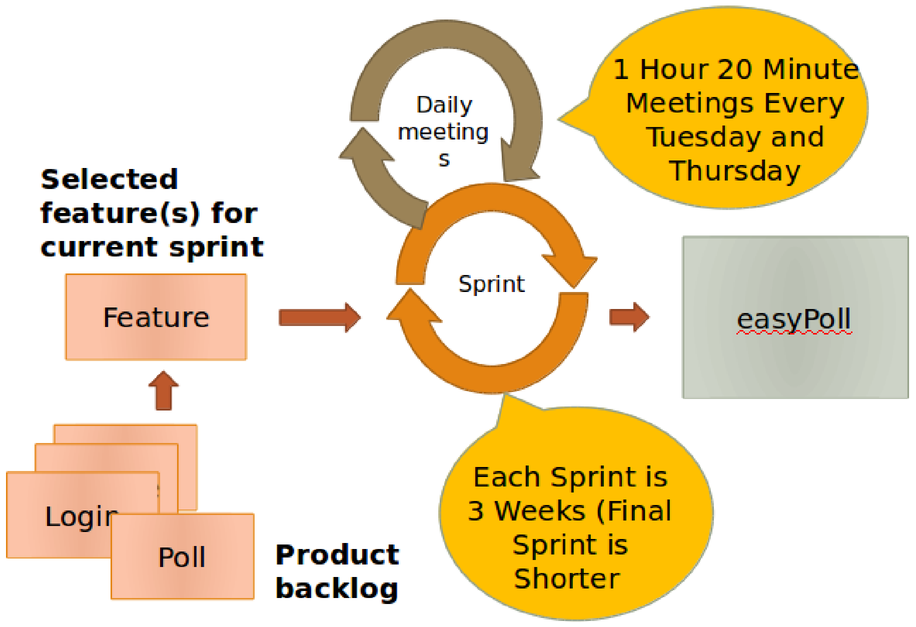
4.1.2 Process Model

4.1.2.1 Plan Description

The process model we have chosen for the development of our application is Scrum. This model will help us build out application by selecting requirements from out product backlog and completing them in our 3 week sprints. We can slowly build the project from implementing requirements from the product backlog, starting with the most risky requirements first. We have a "daily" scrum meeting in person or through messaging.

We plan to meet on Tuesdays and Thursdays, from 3:30 to 4:50 P.M, Facebook Messenger, and when class time is allotted to work on the project. At the start of the sprint delegate requirements to each team member. We started with the requirements that needed the most panning and design. Then we will begin implementing the new features into the project. We test features as we go but also will do a thorough test of the new features at the end of a sprint.

4.1.2.1 Plan Diagram


Daily meeting s
1 Hour 20 Minute Meetings Every Tuesday and Thursday
Selected feature(s) for current sprint
Feature
Sprint
easyPoll
Login
Poll
Product backlog
Each Sprint is 3 Weeks (Final Sprint is Shorter

4.2 Sprint 1

4.2.1 Description

In the first sprint, we wanted to ensure to finalize our design plans and start building the basic tools needed to make our application work and fully lay out our plan of the application. This includes the ability to login to the application, the HTML/CSS site mockups, class diagram, and database of the application.

4.2.2 Satisfying Requirements

This sprint is going to start on requirements 3.2.2 Login, 3.2.6 View Finished Polls, 3.2.9 Create Poll, 3.2.11 View Community Polls, 3.2.15 Vote in Poll, 3.2.22 Make Mobile Friendly

4.2.3 Design

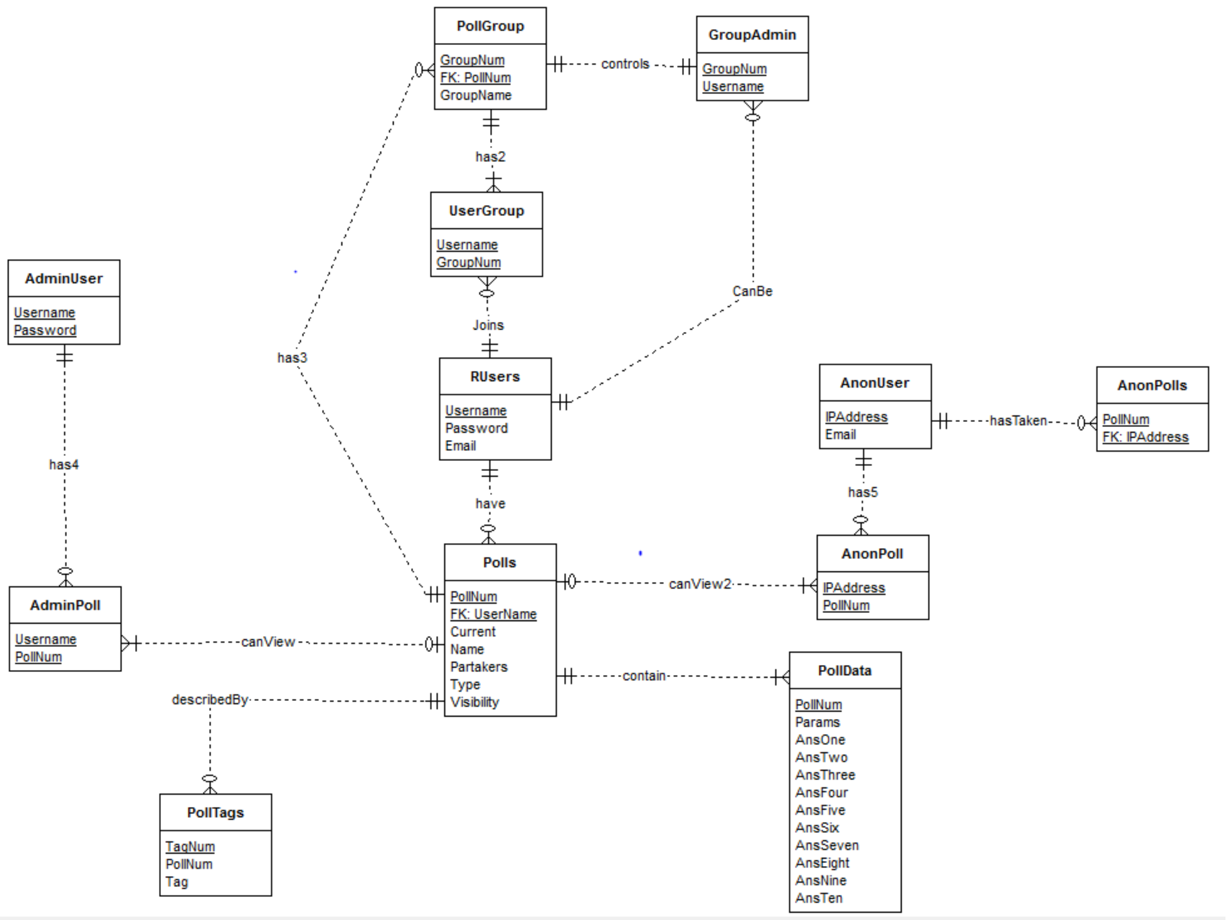
4.2.3.1 Description

We focused on a simple design for the first sprint, as we felt the features we were implementing would be much further expanded on and integrated down the line in order to work with the rest of our use cases.

4.2.3.2 Standards, Frameworks and Tools

Tools we used in the creation of the design of the sprint were modeling tools from Microsoft Visio, and a free modeling website, gliffy.com. Standards we followed in the design of features of the sprint was UML.

4.2.3.4 Database Diagram



4.2.4 Development

4.2.4.1 Description

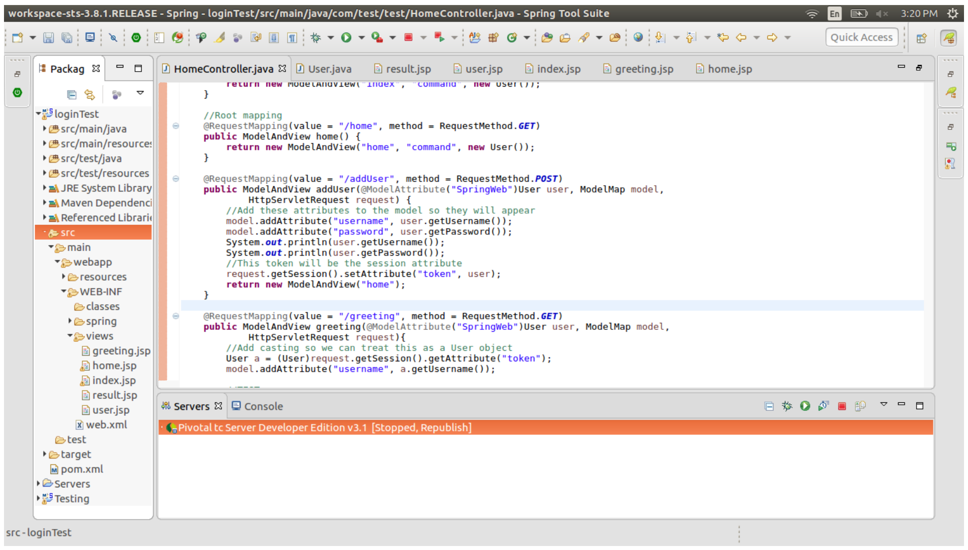
The features we developed in the first sprint was a simple class diagram and setting up the database for persistence of objects. These aspects we felt were vital to the later successful implementation of the core aspects of the application.

4.2.4.2 Standards, Frameworks and Tools

For the development aspect of the sprint, we used a Java IDE in order to create the classes we needed. We used the latest version of Spring Tools Suite version 3.8.2.

4.2.4.3 Screenshots

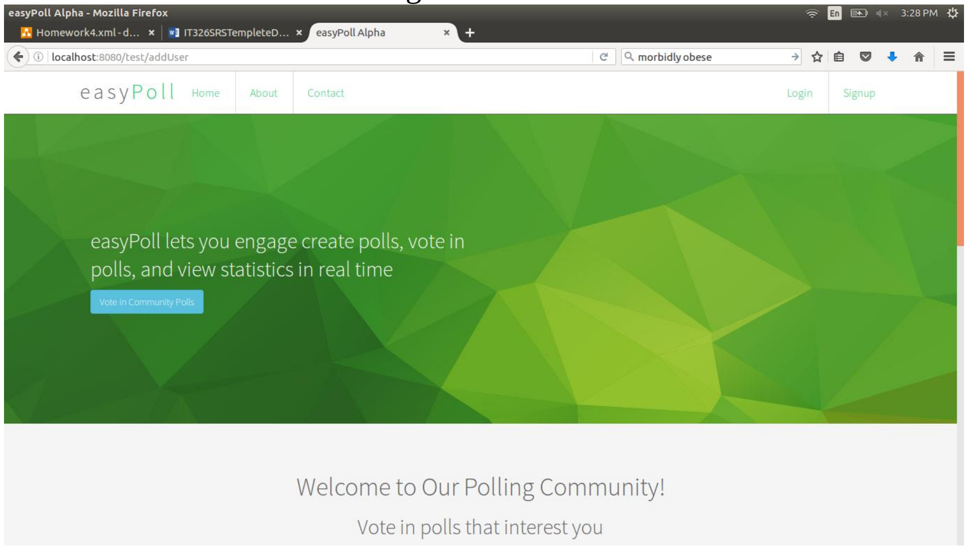
This section includes screen shots of our site mockups.



```java
        return new ModelAndView("index", "command", new User());
    }

    //Root mapping
    @RequestMapping(value = "/home", method = RequestMethod.GET)
    public ModelAndView home() {
        return new ModelAndView("home", "command", new User());
    }

    @RequestMapping(value = "/addUser", method = RequestMethod.POST)
    public ModelAndView addUser(@ModelAttribute("SpringWeb")User user, ModelMap model,
            HttpServletRequest request) {
        //Add these attributes to the model so they will appear
        model.addAttribute("username", user.getUsername());
        model.addAttribute("password", user.getPassword());
        System.out.println(user.getUsername());
        System.out.println(user.getPassword());
        //This token will be the session attribute
        request.getSession().setAttribute("token", user);
        return new ModelAndView("home");
    }

    @RequestMapping(value = "/greeting", method = RequestMethod.GET)
    public ModelAndView greeting(@ModelAttribute("SpringWeb")User user, ModelMap model,
            HttpServletRequest request){
        //Add casting so we can treat this as a User object
        User a = (User)request.getSession().getAttribute("token");
        model.addAttribute("username", a.getUsername());
```

```
Pivotal tc Server Developer Edition v3.1  [Stopped, Republish]
```

Screenshot 4.2.4.3.1 - Image of the code for the home controller

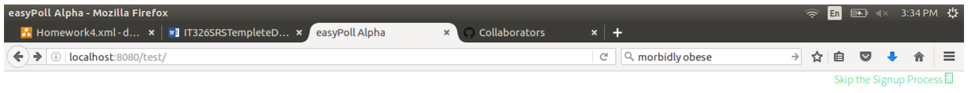
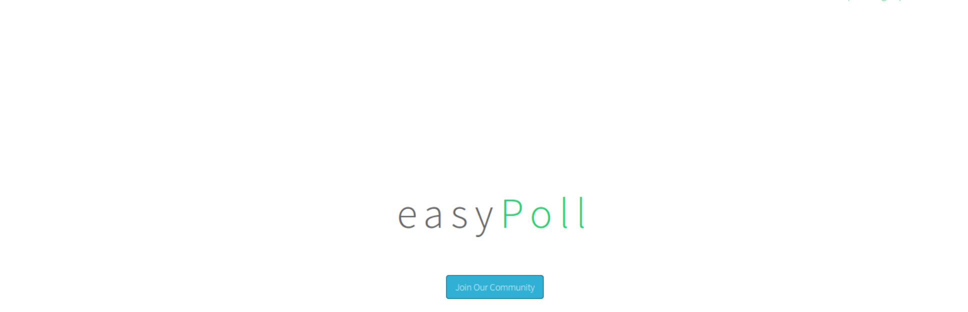


easyPoll lets you engage create polls, vote in polls, and view statistics in real time

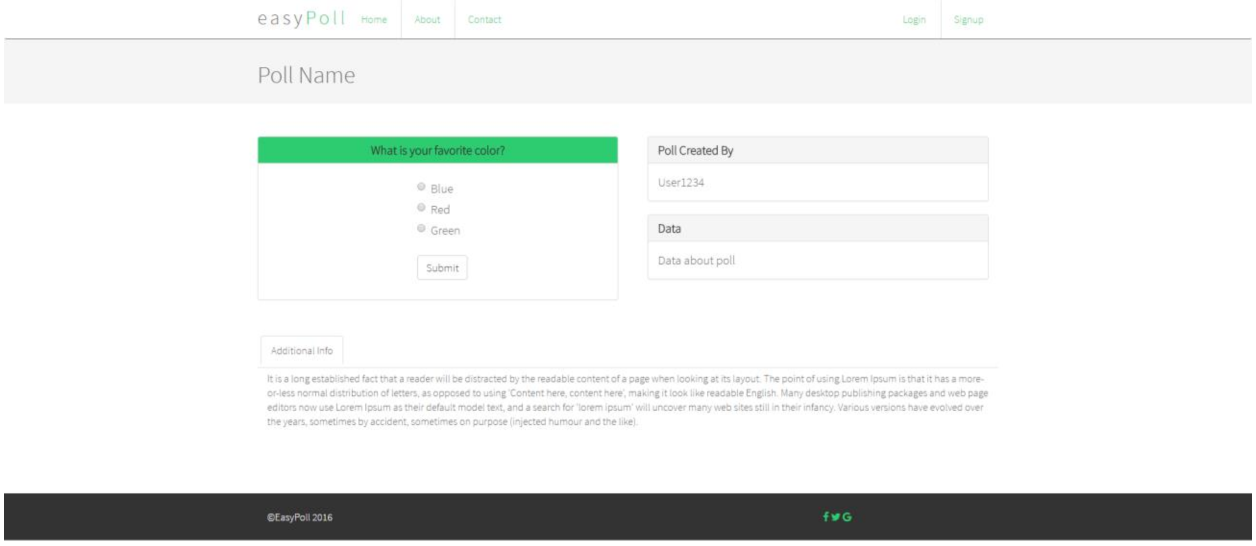
Welcome to Our Polling Community!

Vote in polls that interest you

Screenshot 4.2.4.3.2- Image of our home page

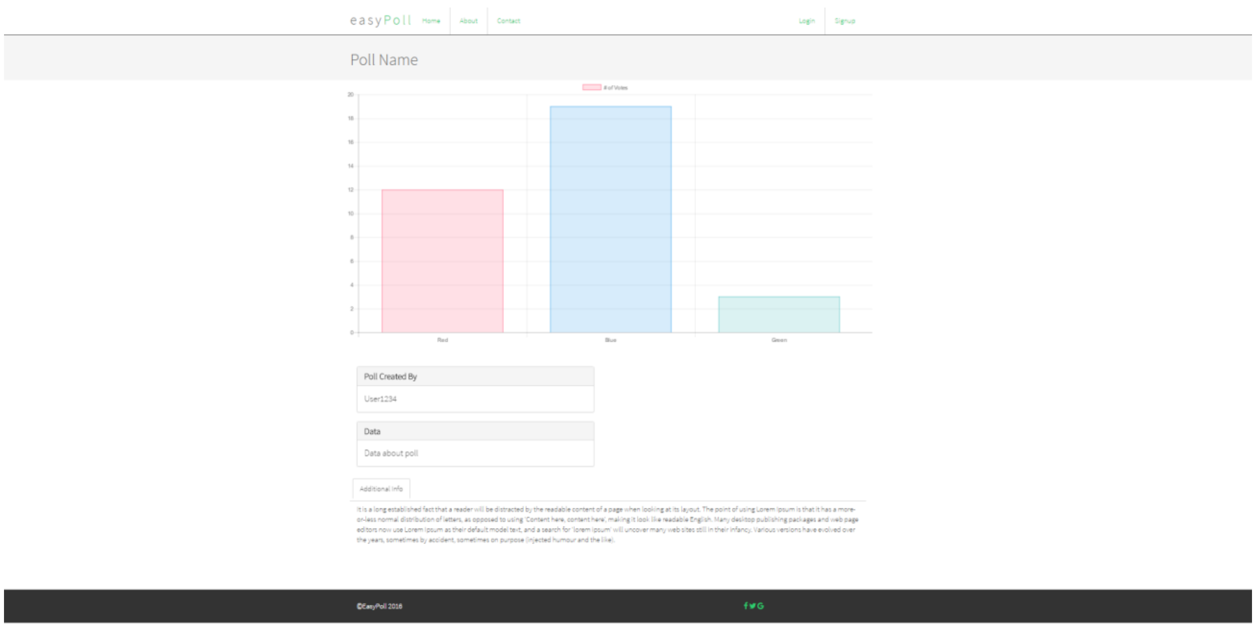



Screenshot 4.2.4.3.3 - Image of the landing page



It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Screenshot 4.2.4.3.4 - Image of the single poll before voting



Poll Created By

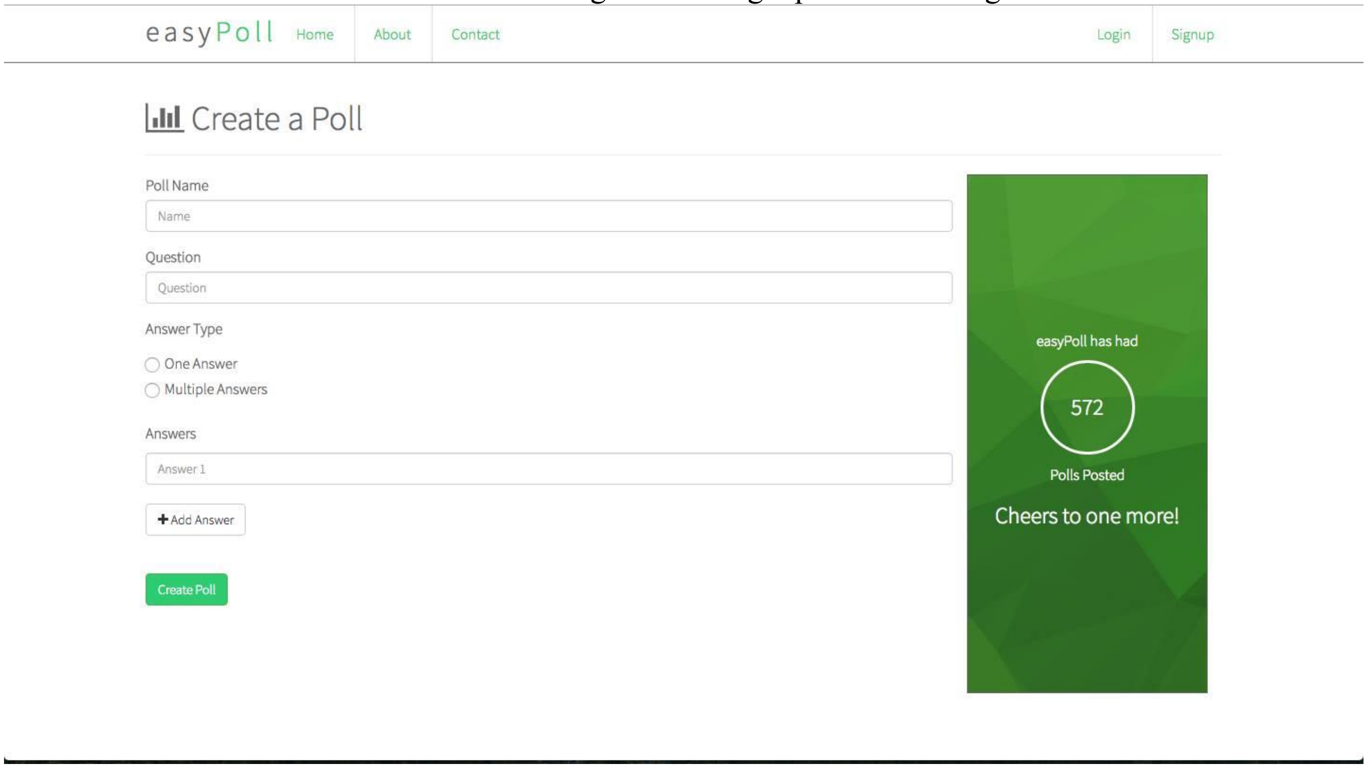
User1234

Data

Data about poll

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Screenshot 4.2.4.3.5 - Image of the single poll after voting



Poll Name

Name

Question

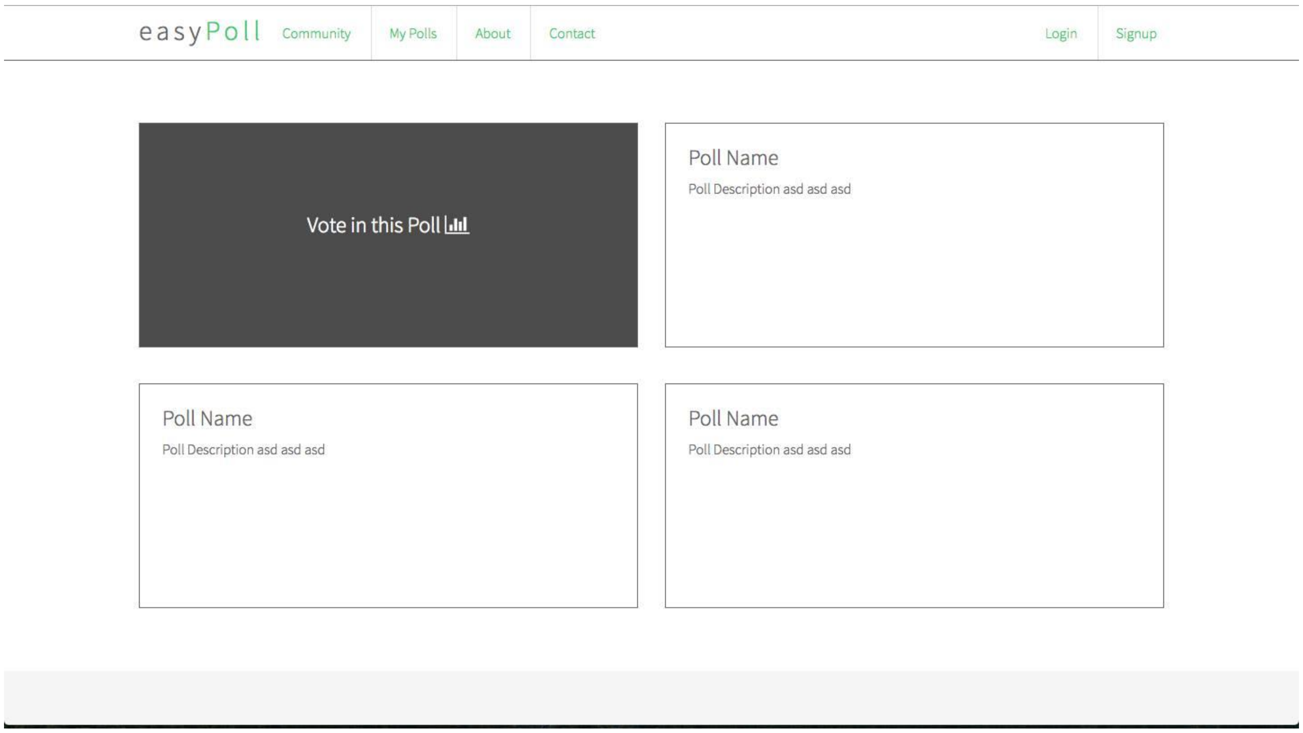
Question

Answer Type

Answers

Answer 1

Screenshot 4.2.4.3.6 - Image of create poll



Screenshot 4.2.4.3.7 - Bottom of homepage

4.3 Sprint 2

4.3.1 Description

In the second sprint we started implementing more requirements using our class diagram and making jsp (html) pages. We worked on connecting back end to front end by implementing our database design and successfully getting and adding things to the database using our application. We implemented models and adding methods to our home controller.

4.3.2 Satisfying Requirements

This sprint is going to implement requirements 3.2.2 Login, 3.2.6 View Finished Polls, 3.2.9 Create Poll, 3.2.11 View Community Polls, 3.2.15 Vote in Poll, 3.2.22 Make Mobile Friendly, 3.2.1 Create Account, 3.2.3 Vote in Anonymous Poll, 3.2.4 Vote in Private Poll, 3.2.5 View Poll Statistics, 3.2.10 View My Polls, 3.2.12 View User Statistics, 3.2.14 Limit Users to One Vote Per Poll, 3.2.25 Forgot Password Service, 3.2.36 View Reported Offensive Polls

4.3.3 Design

4.3.3.1 Description

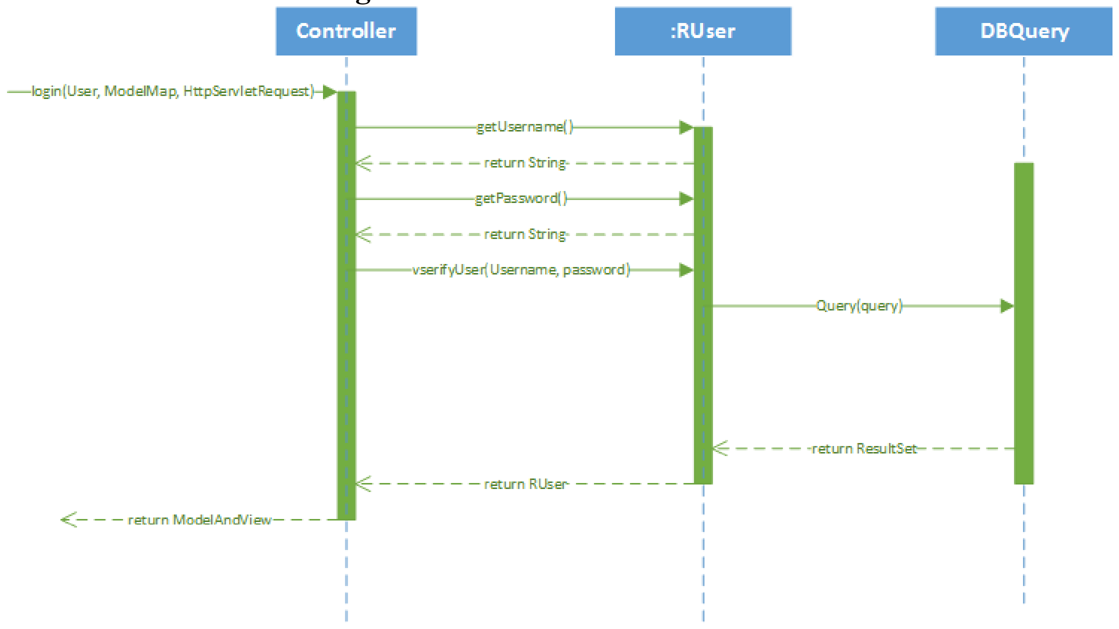
We focused on completing our class diagram and diagramming more specific actions of application.

4.3.3.2 Standards, Frameworks and Tools

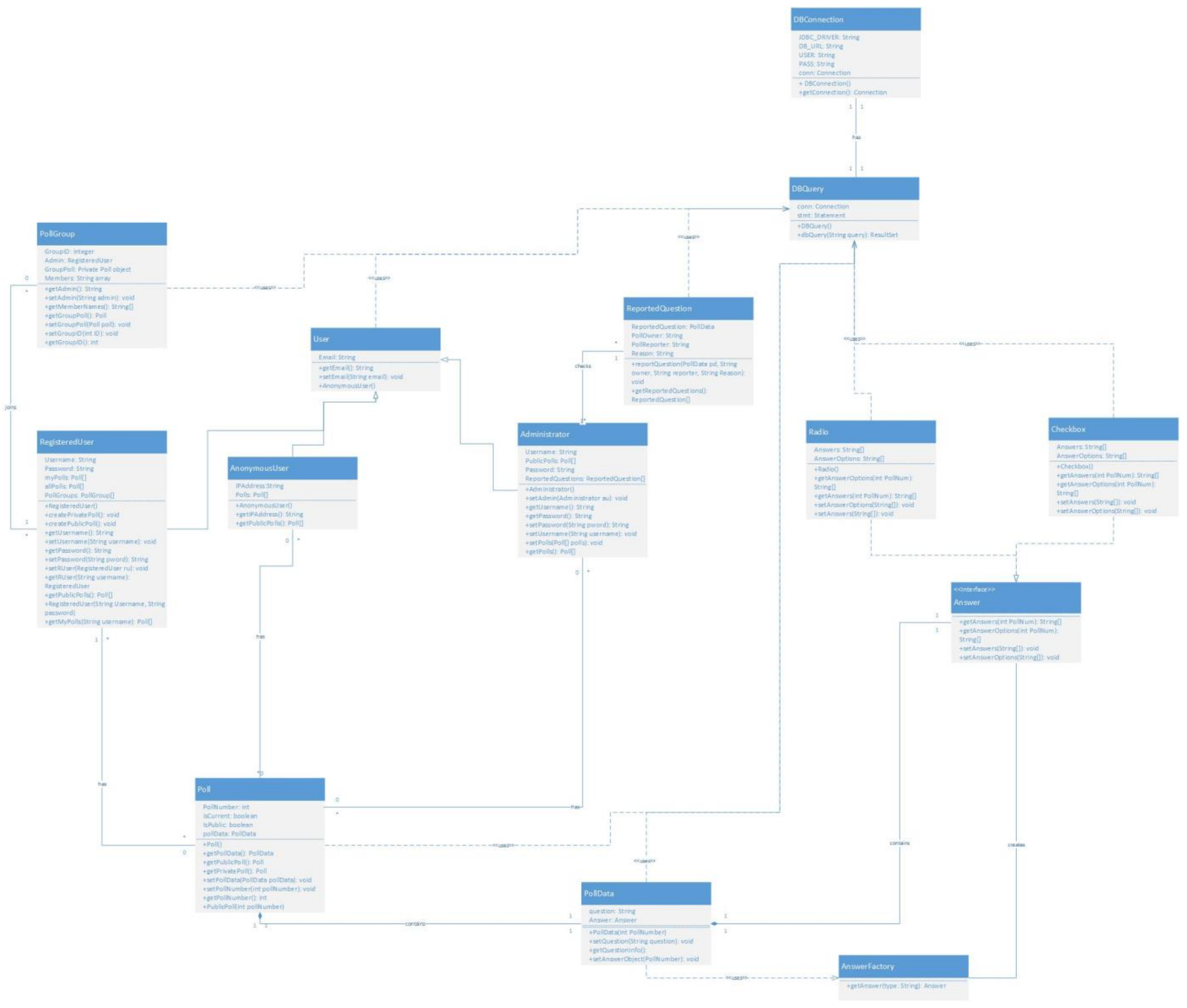
Tools we used in the creation of the design of the sprint were modeling tools from Microsoft Visio. Standards we followed in the design of features of the sprint was UML.

4.3.3.3 Sequence Diagram

4.3.3.3.3.1 User Login



4.3.3.4 Class Diagram


DBConnection
JDBC_DRIVER: String
DB_URL: String
USER: String
PASS: String
conn: Connection
+ DBConnection()
+getConnection(): Connection
has
DBQuery
conn: Connection
stmt: Statement
+DBQuery()
+dbQuery(String query): ResultSet
PollGroup
GroupID: integer
Admin: RegisteredUser
GroupPoll: Private Poll object
Members: String array
+getAdmin(): String
+setAdmin(String admin): void
+getMemberNames(): String[]
+getGroupPoll(): Poll
+setGroupPoll(Poll poll): void
+setGroupID(int ID): void
+getGroupID(): int
<<uses>>
joins
User
Email: String
+getEmail(): String
+setEmail(String email): void
+AnonymousUser()
ReportedQuestion
ReportedQuestion: PollData
PollOwner: String
PollReporter: String
Reason: String
+reportQuestion(PollData pd, String owner, String reporter, String Reason): void
+getReportedQuestions(): ReportedQuestion[]
checks
RegisteredUser
Username: String
Password: String
myPolls: Poll[]
allPolls: Poll[]
PollGroups: PollGroup[]
+RegisteredUser()
+createPrivatePoll(): void
+createPublicPoll(): void
+getUsername(): String
+setUsername(String username): void
+getPassword(): String
+setPassword(String pword): String
+setRUser(RegisteredUser ru): void
+getRUser(String username): RegisteredUser
+getPublicPolls(): Poll[]
+RegisteredUser(String Username, String password)
+getMyPolls(String username): Poll[]
AnonymousUser
IPAddress:String
Polls: Poll[]
+AnonymousUser()
+getIPAddress(): String
+getPublicPolls(): Poll[]
Administrator
Username: String
PublicPolls: Poll[]
Password: String
ReportedQuestions: ReportedQuestion[]
+Administrator()
+setAdmin(Administrator au): void
+getUsername(): String
+getPassword(): String
+setPassword(String pword): String
+setUsername(String username): void
+setPolls(Poll[] polls): void
+getPolls(): Poll[]
Radio
Answers: String[]
AnswerOptions: String[]
+Radio()
+getAnswerOptions(int PollNum): String[]
+getAnswers(int PollNum): String[]
+setAnswerOptions(String[]): void
+setAnswers(String[]): void
Checkbox
Answers: String[]
AnswerOptions: String[]
+Checkbox()
+getAnswers(int PollNum): String[]
+getAnswerOptions(int PollNum): String[]
+setAnswers(String[]): void
+setAnswerOptions(String[]): void
<<Interface>>
Answer
+getAnswers(int PollNum): String[]
+getAnswerOptions(int PollNum): String[]
+setAnswers(String[]): void
+setAnswerOptions(String[]): void
has
Poll
PollNumber: int
isCurrent: boolean
isPublic: boolean
pollData: PollData
+Poll()
+getPollData(): PollData
+getPublicPoll(): Poll
+getPrivatePoll(): Poll
+setPollData(PollData pollData): void
+setPollNumber(int pollNumber): void
+getPollNumber(): int
+PublicPoll(int pollNumber)
contains
PollData
question: String
Answer: Answer
+PollData(int PollNumber)
+setQuestion(String question): void
+getQuestionInfo():
+setAnswerObject(PollNumber): void
contains
creates
AnswerFactory
+getAnswer(type: String): Answer

4.3.4 Development

4.3.4.1 Description

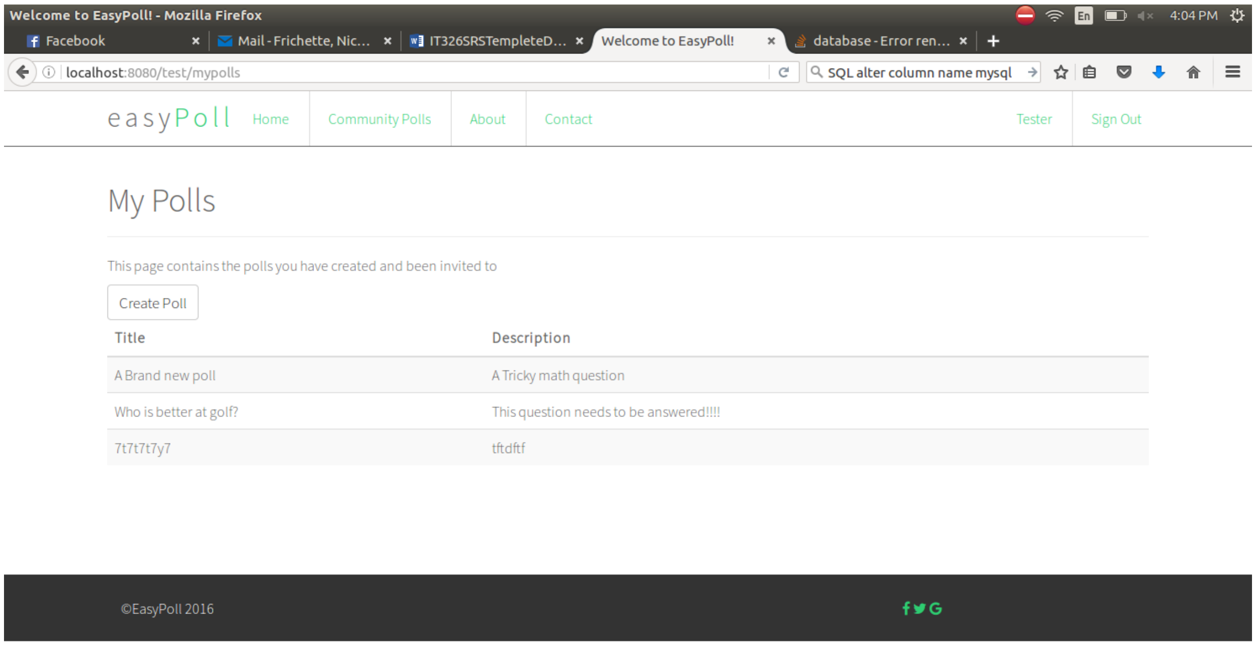
We focused on implementing our design so we could complete user login and voting system. We implemented out database design and successfully se out application to get and put information into the database.

4.3.4.2 Standards, Frameworks and Tools

For the development aspect of the sprint, we used a Java IDE in order to create the classes we needed. We used the latest version of Spring Tools Suite version 3.8.2.

4.3.4.3 Screenshots

This section includes screen shots of out application.

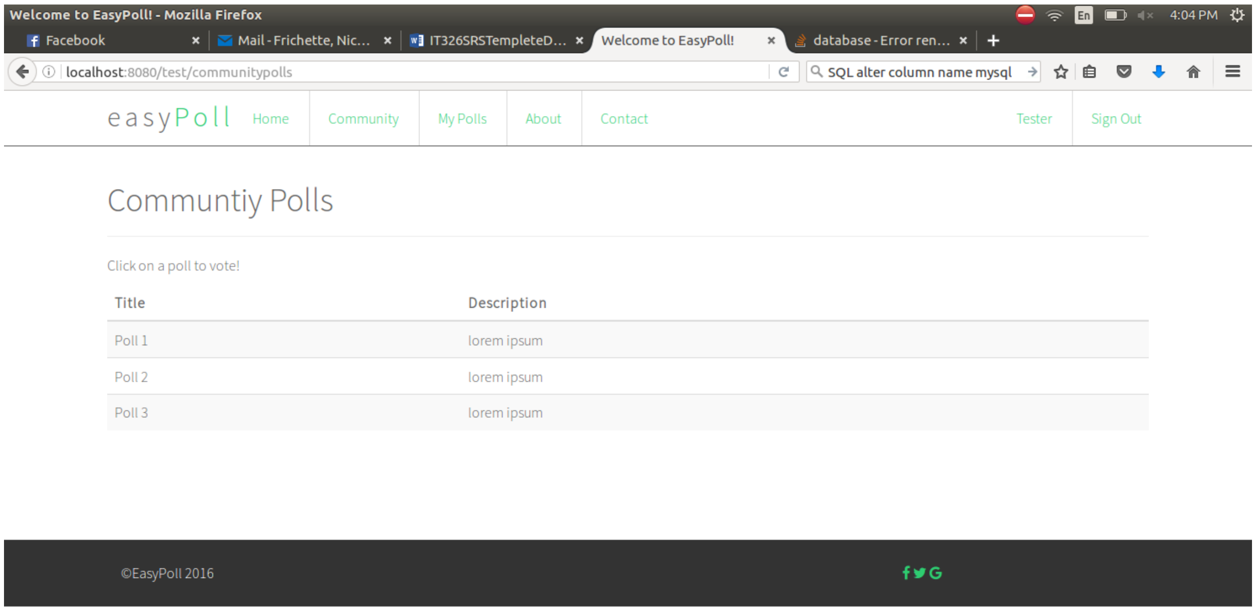


My Polls

This page contains the polls you have created and been invited to

Title	Description
A Brand new poll	A Tricky math question
Who is better at golf?	This question needs to be answered!!!!
7t7t7t7y7	tftdftf

Screenshot 4.3.4.3.1 - Image my polls page

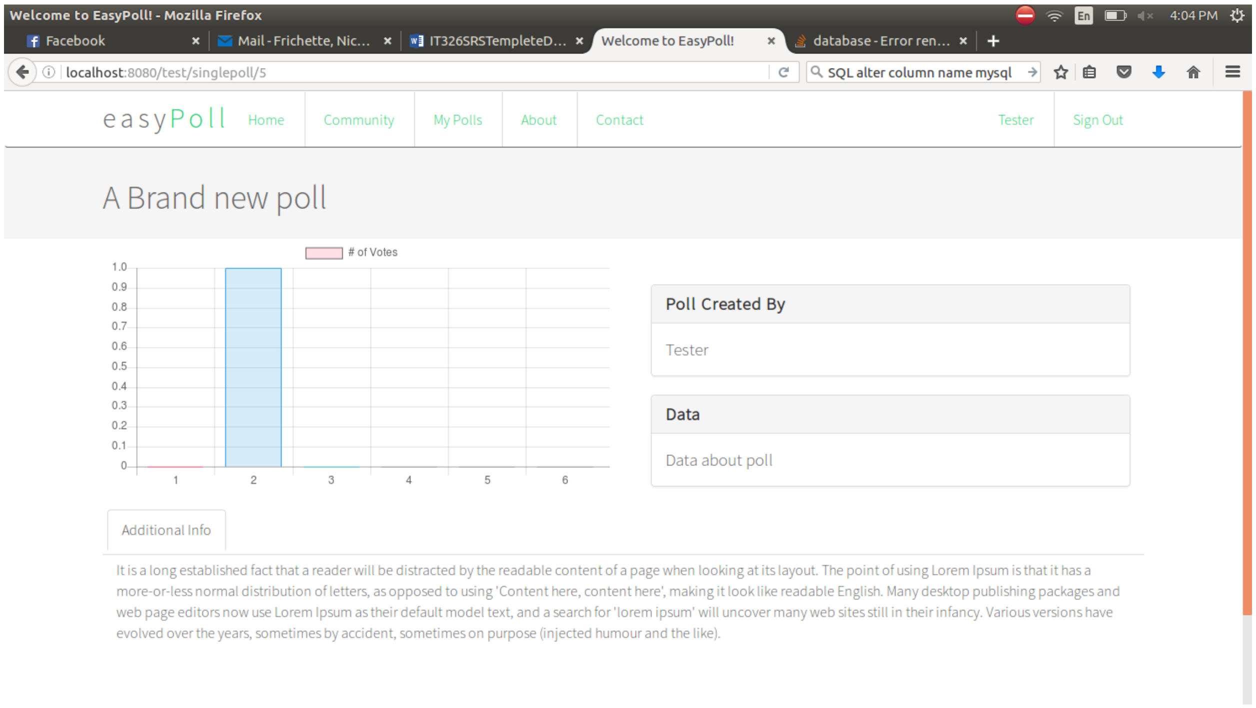


Communtiy Polls

Click on a poll to vote!

Title	Description
Poll 1	lorem ipsum
Poll 2	lorem ipsum
Poll 3	lorem ipsum

Screenshot 4.3.4.3.2 - Image of community polls page


Welcome to EasyPoll! - Mozilla Firefox
localhost:8080/test/singlepoll/5
easyPoll Home Community My Polls About Contact Tester Sign Out
# of Votes
Additional Info

A Brand new poll

Poll Created By

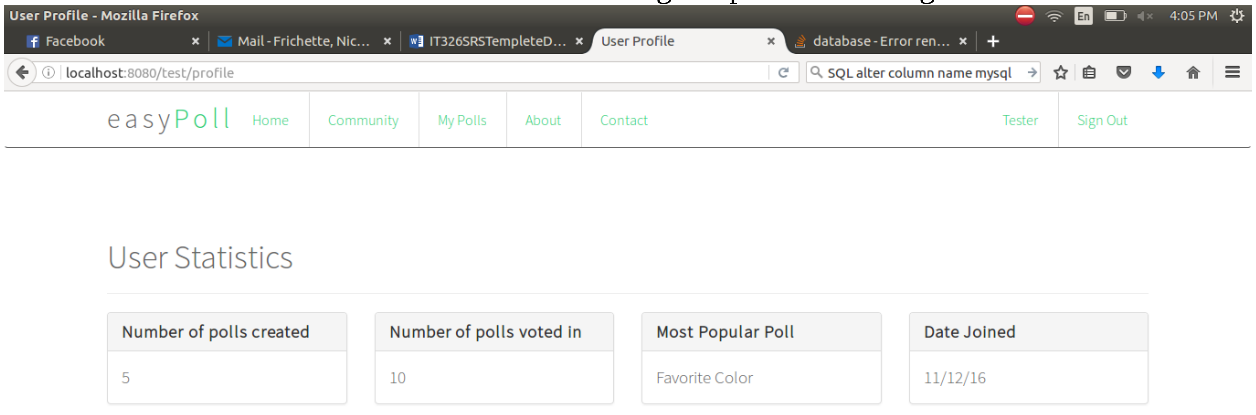
Tester

Data

Data about poll

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

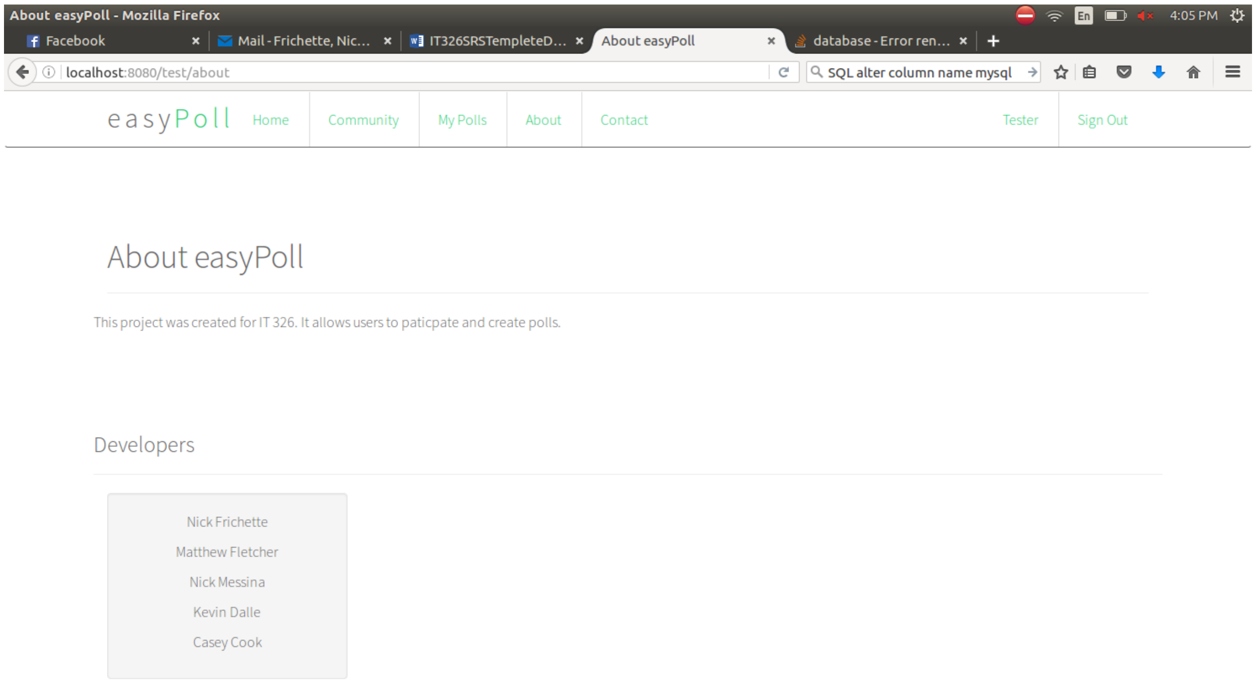
Screenshot 4.3.4.3.3 - Image of poll after voting


User Profile - Mozilla Firefox
localhost:8080/test/profile
easyPoll Home Community My Polls About Contact Tester Sign Out

User Statistics

Number of polls created	Number of polls voted in	Most Popular Poll	Date Joined
5	10	Favorite Color	11/12/16

Screenshot 4.3.4.3.4 - Image of user statistics page



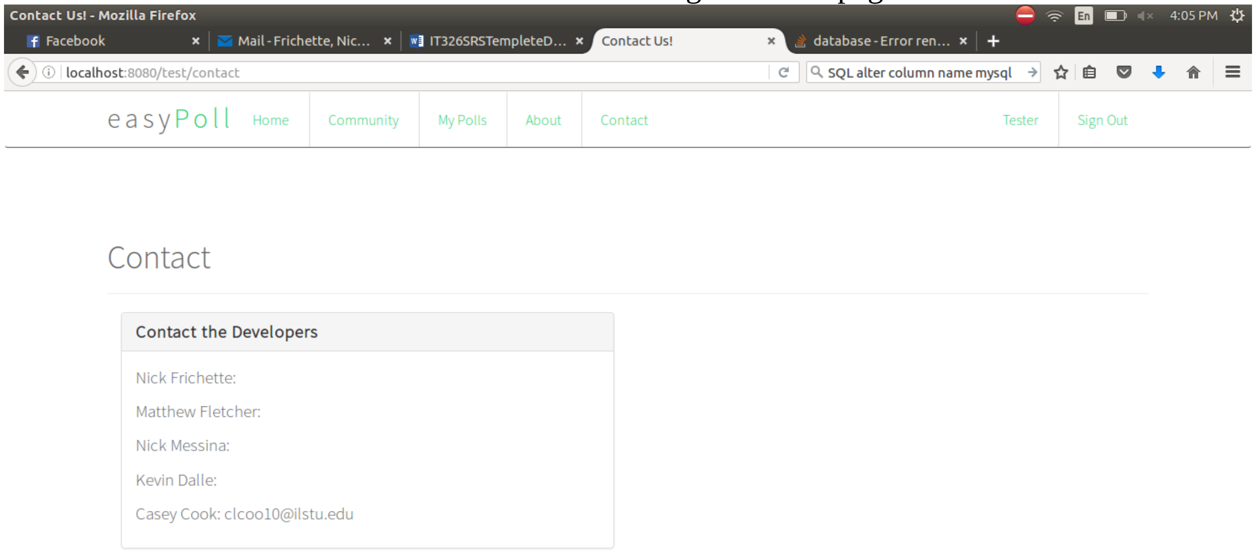
About easyPoll

This project was created for IT 326. It allows users to paticpate and create polls.

Developers

Nick Frichette
Matthew Fletcher
Nick Messina
Kevin Dalle
Casey Cook

Screenshot 4.3.4.3.5- Image of about page

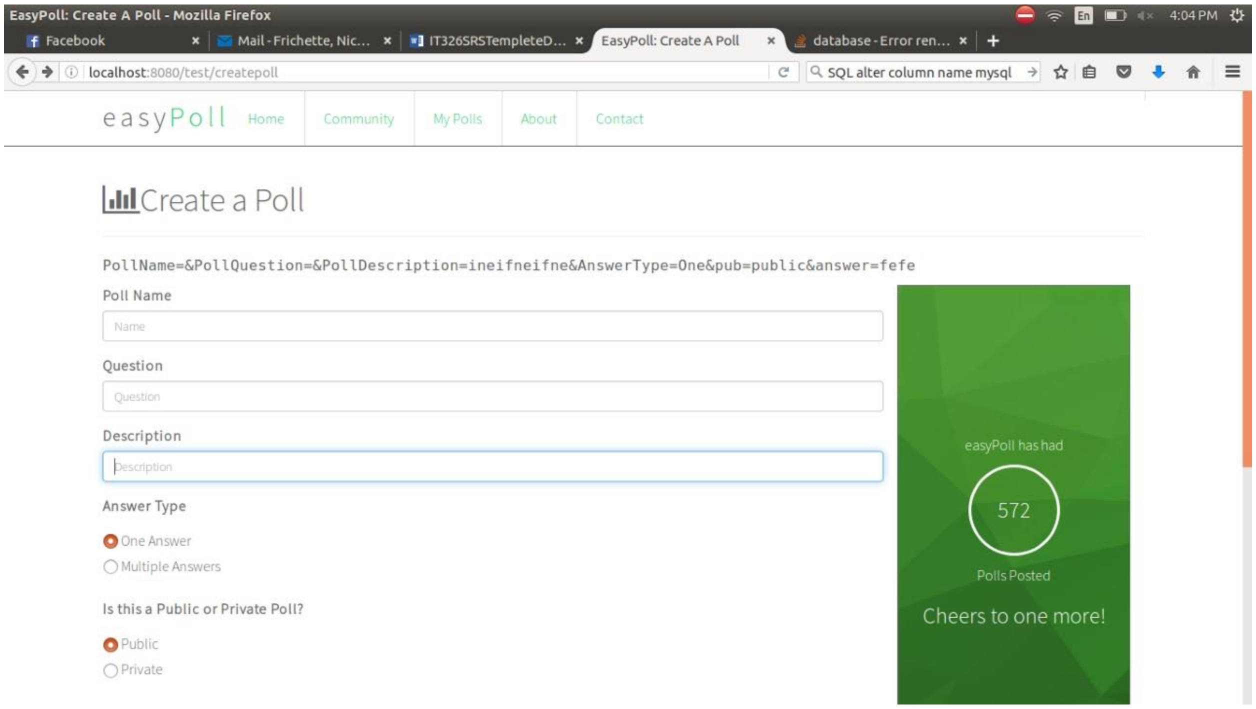


Contact

Contact the Developers

Nick Frichette:
Matthew Fletcher:
Nick Messina:
Kevin Dalle:
Casey Cook: clcoo10@ilstu.edu

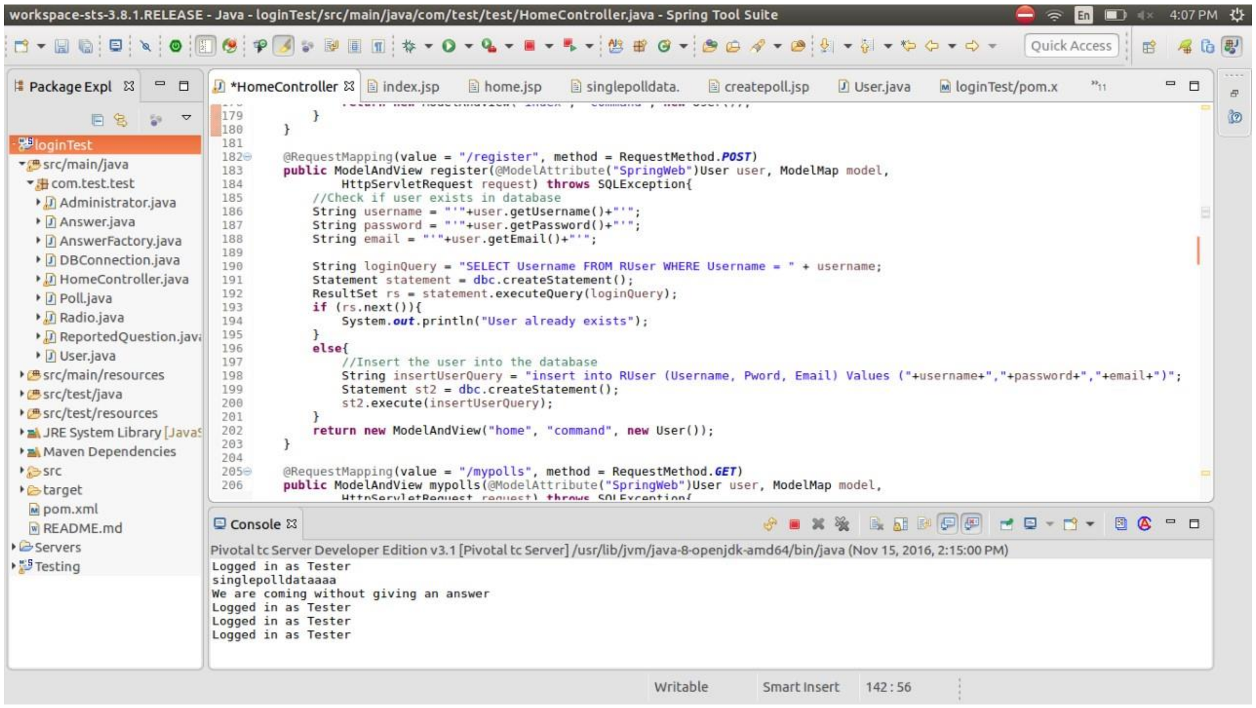
Screenshot 4.3.4.3.6 - Image of contact page



```
PollName=&PollQuestion=&PollDescription=ineifneifne&AnswerType=One&pub=public&answer=fefe
```

Poll Name: Name
Question: Question
Description: Description
Answer Type: [✓ One Answer] Multiple Answers
Is this a Public or Private Poll? [✓ Public] Private

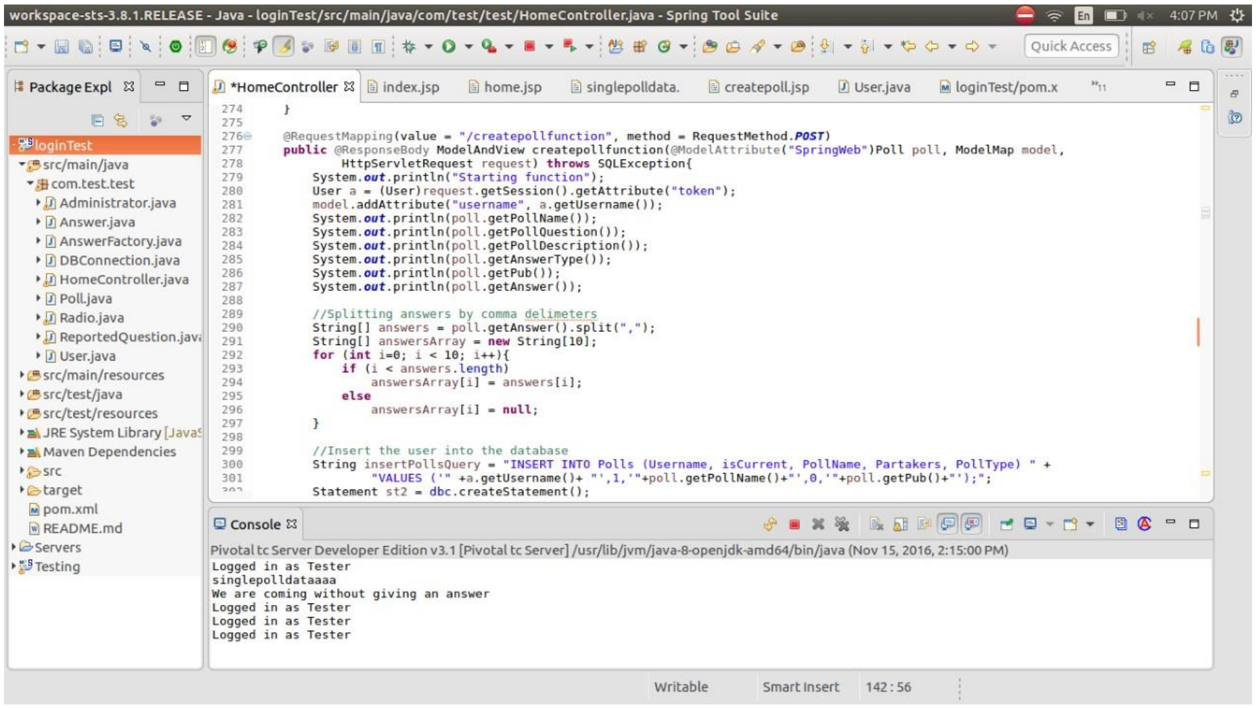
Screenshot 4.3.4.3.7 - Image of create poll page



```java
        }
    }

    @RequestMapping(value = "/register", method = RequestMethod.POST)
    public ModelAndView register(@ModelAttribute("SpringWeb")User user, ModelMap model,
            HttpServletRequest request) throws SQLException{
        //Check if user exists in database
        String username = "'"+user.getUsername()+"'";
        String password = "'"+user.getPassword()+"'";
        String email = "'"+user.getEmail()+"'";

        String loginQuery = "SELECT Username FROM RUser WHERE Username = " + username;
        Statement statement = dbc.createStatement();
        ResultSet rs = statement.executeQuery(loginQuery);
        if (rs.next()){
            System.out.println("User already exists");
        }
        else{
            //Insert the user into the database
            String insertUserQuery = "insert into RUser (Username, Pword, Email) Values ("+username+","+password+","+email+")";
            Statement st2 = dbc.createStatement();
            st2.execute(insertUserQuery);
        }
        return new ModelAndView("home", "command", new User());
    }

    @RequestMapping(value = "/mypolls", method = RequestMethod.GET)
    public ModelAndView mypolls(@ModelAttribute("SpringWeb")User user, ModelMap model,
```

```
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Nov 15, 2016, 2:15:00 PM)
Logged in as Tester
singlepolldataaaa
We are coming without giving an answer
Logged in as Tester
Logged in as Tester
Logged in as Tester
```

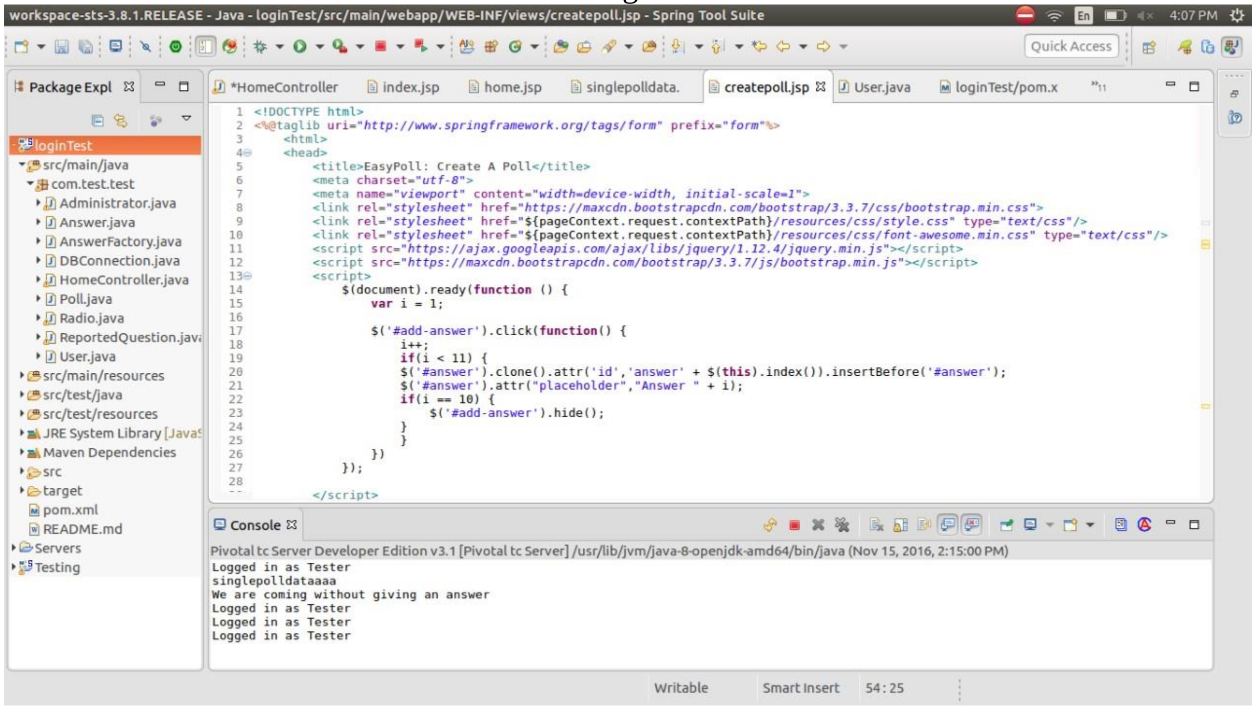
Screenshot 4.3.4.3.8 - Image of home controller


workspace-sts-3.8.1.RELEASE - Java - loginTest/src/main/java/com/test/test/HomeController.java - Spring Tool Suite

```
    }

    @RequestMapping(value = "/createpollfunction", method = RequestMethod.POST)
    public @ResponseBody ModelAndView createpollfunction(@ModelAttribute("SpringWeb")Poll poll, ModelMap model,
            HttpServletRequest request) throws SQLException{
        System.out.println("Starting function");
        User a = (User)request.getSession().getAttribute("token");
        model.addAttribute("username", a.getUsername());
        System.out.println(poll.getPollName());
        System.out.println(poll.getPollQuestion());
        System.out.println(poll.getPollDescription());
        System.out.println(poll.getAnswerType());
        System.out.println(poll.getPub());
        System.out.println(poll.getAnswer());

        //Splitting answers by comma delimeters
        String[] answers = poll.getAnswer().split(",");
        String[] answersArray = new String[10];
        for (int i=0; i < 10; i++){
            if (i < answers.length)
                answersArray[i] = answers[i];
            else
                answersArray[i] = null;
        }

        //Insert the user into the database
        String insertPollsQuery = "INSERT INTO Polls (Username, isCurrent, PollName, Partakers, PollType) " +
                "VALUES ('" +a.getUsername()+ "',1,'"+poll.getPollName()+"',0,'"+poll.getPub()+"');";
        Statement st2 = dbc.createStatement();
```

```
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Nov 15, 2016, 2:15:00 PM)
Logged in as Tester
singlepolldataaaa
We are coming without giving an answer
Logged in as Tester
Logged in as Tester
Logged in as Tester
```

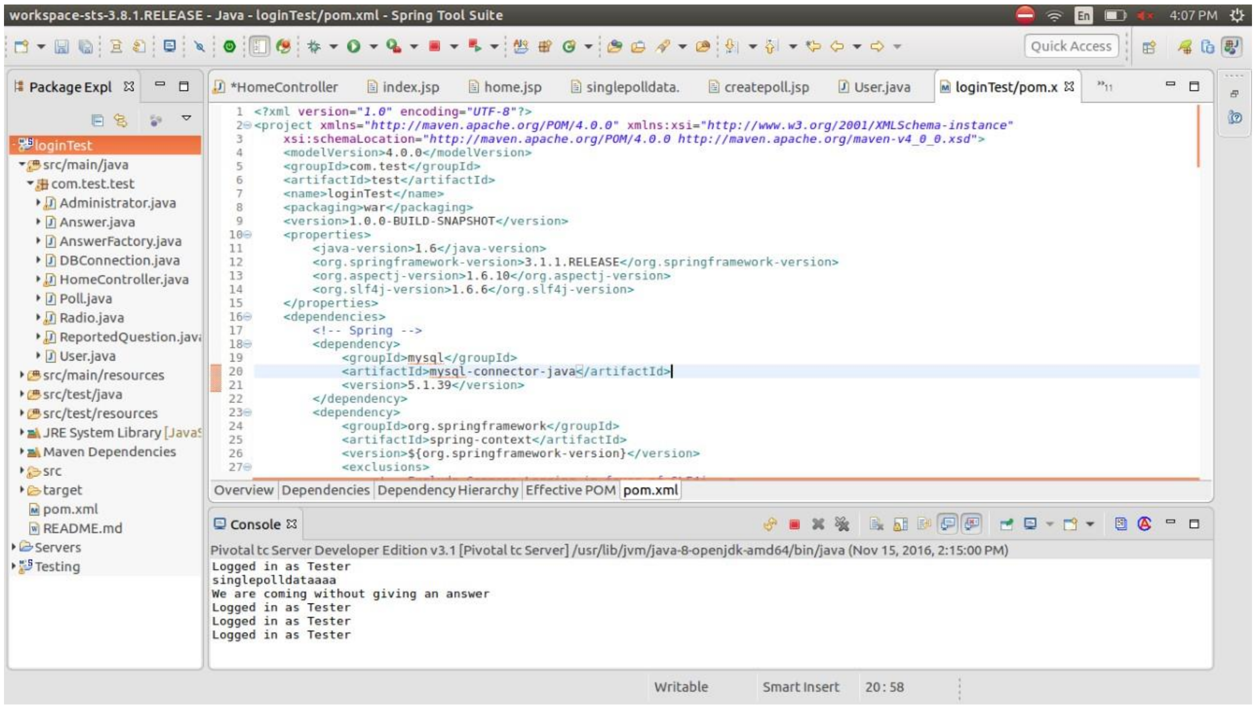
Screenshot 4.3.4.3.9 - Image of home controller cont.


workspace-sts-3.8.1.RELEASE - Java - loginTest/src/main/webapp/WEB-INF/views/createpoll.jsp - Spring Tool Suite

```
<!DOCTYPE html>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
    <html>
    <head>
        <title>EasyPoll: Create A Poll</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/style.css" type="text/css"/>
        <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/font-awesome.min.css" type="text/css"/>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
        <script>
            $(document).ready(function () {
                var i = 1;

                $('#add-answer').click(function() {
                    i++;
                    if(i < 11) {
                    $('#answer').clone().attr('id','answer' + $(this).index()).insertBefore('#answer');
                    $('#answer').attr("placeholder","Answer " + i);
                    if(i == 10) {
                        $('#add-answer').hide();
                    }
                    }
                })
            });

        </script>
```

```
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Nov 15, 2016, 2:15:00 PM)
Logged in as Tester
singlepolldataaaa
We are coming without giving an answer
Logged in as Tester
Logged in as Tester
Logged in as Tester
```

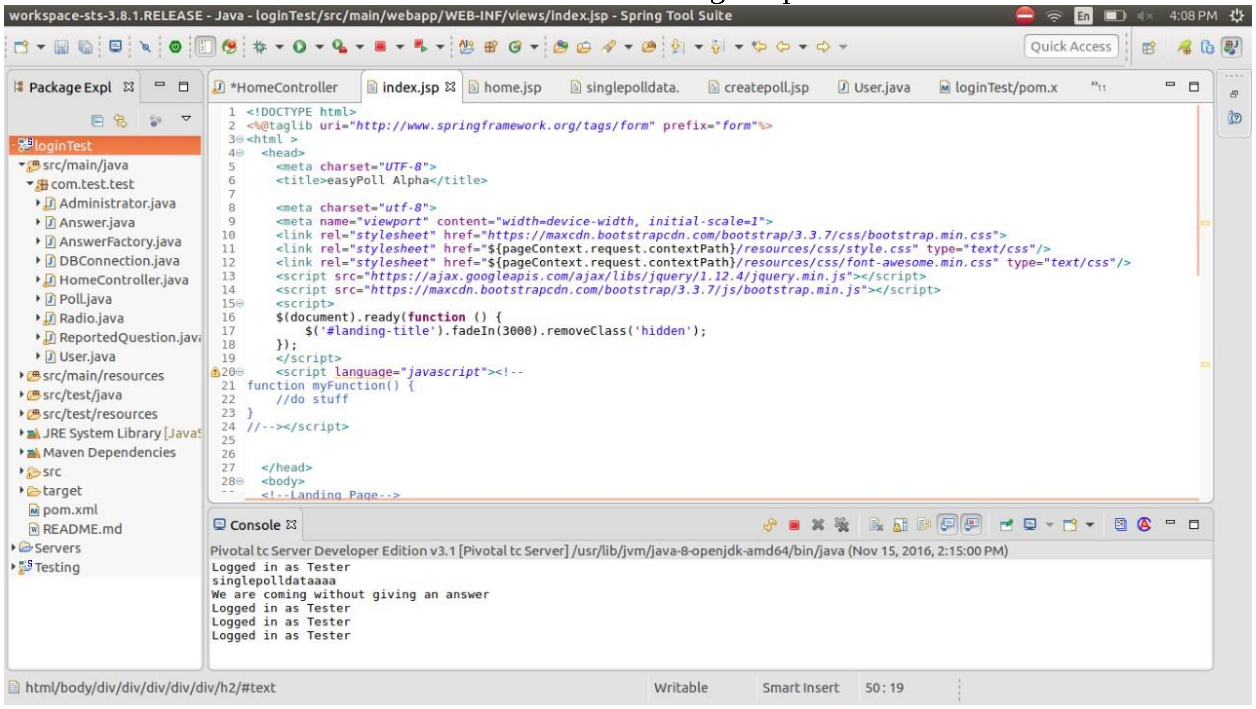
Screenshot 4.3.4.3.10 - Image of createpoll.jsp page



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test</groupId>
    <artifactId>test</artifactId>
    <name>loginTest</name>
    <packaging>war</packaging>
    <version>1.0.0-BUILD-SNAPSHOT</version>
    <properties>
        <java-version>1.6</java-version>
        <org.springframework-version>3.1.1.RELEASE</org.springframework-version>
        <org.aspectj-version>1.6.10</org.aspectj-version>
        <org.slf4j-version>1.6.6</org.slf4j-version>
    </properties>
    <dependencies>
        <!-- Spring -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.39</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${org.springframework-version}</version>
            <exclusions>
```

```
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Nov 15, 2016, 2:15:00 PM)
Logged in as Tester
singlepolldataaaa
We are coming without giving an answer
Logged in as Tester
Logged in as Tester
Logged in as Tester
```

Screenshot 4.3.4.3.11 - Image of pom.xml



```
<!DOCTYPE html>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html >
  <head>
    <meta charset="UTF-8">
    <title>easyPoll Alpha</title>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/style.css" type="text/css"/>
    <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/font-awesome.min.css" type="text/css"/>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <script>
    $(document).ready(function () {
        $('#landing-title').fadeIn(3000).removeClass('hidden');
    });
    </script>
    <script language="javascript"><!--
function myFunction() {
    //do stuff
}
//--></script>


  </head>
  <body>
  <!--Landing Page-->
```

```
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Nov 15, 2016, 2:15:00 PM)
Logged in as Tester
singlepolldataaaa
We are coming without giving an answer
Logged in as Tester
Logged in as Tester
Logged in as Tester
```

Screenshot 4.3.4.3.11 - Image of index.jsp

4.4 Sprint 3

4.4.1 Description

During Sprint 3, we implemented the majority of features as well as did a significant amount of bug testing.

4.4.2 Satisfying Requirements

This sprint is going to implement requirements 3.2.2 Login, 3.2.6 View Finished Polls, 3.2.9 Create Poll, 3.2.11 View Community Polls, 3.2.15 Vote in Poll, 3.2.22 Make Mobile Friendly, 3.2.1 Create Account, 3.2.4 Vote in Private Poll, 3.2.5 View Poll Statistics, 3.2.10 View My Polls, 3.2.12 View User Statistics, 3.2.14 Limit Users to One Vote Per Poll, 3.2.25 Forgot Password Service, 3.2.36 View Reported Offensive Polls, Vote in poll anonymously {3.2.3}, End Poll {3.2.7}, Edit Poll {3.2.8}, Invite User to Private Polls {3.2.13}, Delete Invited Users {3.2.16}, Delete Poll {3.2.17}, Cancel Poll Early {3.2.18}, Edit Account {3.2.19}, Delete Account {3.2.20}, Export Poll Data as CSV {3.2.21}, Report Offensive Polls {3.2.23}, Create Admin Account {3.2.24}, Thank you Email Survey Response {3.2.26}, Send Newsletter {3.2.27}, Recommend Polls to Others {3.2.29}, Create Poll Groups {3.2.30}, Add Users to Group {3.2.31}, Delete Users from Group {3.2.32}, Delete Group {3.2.33}, Send Support Ticket {3.2.34}, Send Feedback {3.2.35}, View Reported Feedback {3.2.37},

4.4.3 Design

4.4.3.1 Description

We focused on completing our class diagram and diagramming more specific actions of application.

4.4.3.2 Standards, Frameworks and Tools

Tools we used in the creation of the design of the sprint were modeling tools from Microsoft Visio. Standards we followed in the design of features of the sprint was UML.

4.4.4 Development

4.4.4.1 Description

We focused on implementing our design so we could complete user login and voting system. We implemented out database design and successfully se out application to get and put information into the database.

4.4.4.2 Standards, Frameworks and Tools

For the development aspect of the sprint, we used a Java IDE in order to create the classes we needed. We used the latest version of Spring Tools Suite version 3.8.2.

4.4.4.4 Testing

Integration Testing: Over the course of the project we have experienced integration testing regularly. We have been using Git as our version control software, and as a result we have to merge every time someone commits to the repository. We used a Big Bang approach every time there was a merge. After each merge we would test the core functionality to ensure that each feature is working.

The biggest issue we ran into was that we didn't use source control to track our database creation script. Every time we would make a change to the database structure we would have to modify the creation script. If someone fell behind they would have regular crashes in the application until they updated. Once we had nailed down the exact database design this issue was resolved.

1) Requirement: 3.2.34 Send Feedback – Nick Frichette
Objective: To perform black box testing of the Send Feedback functional requirement.
Testing: I will perform Black Box testing and try the following three test cases. No input, a normal amount of input, and a very large input. My results are in the table below.

Action	**Result**
No Input	Error handling kicked in. Pop-up stopped the user and said "Please fill out this field".
Normal Input	The input string is properly stored in the database.
Very Large Input	Even with input of over 7000 characters, the feedback input is still properly stored in the database.

Based on this information, the functional requirement is performing as required.

2) Requirement: 3.2.8 Edit Poll – Nick Frichette
Objective: To perform black box testing of the Edit Poll functional requirement
Testing: I will perform Black Box testing to test the following two use cases. No difference in input, and different input.

Action	**Result**
No Difference In Input	The function returned properly and did not modify any values in the.
Different Input	The function successfully changed the values of the poll.

My results show that our functional requirement is performing as planned.

3) Requirement 3.2.17 Delete Poll - Casey Cook
Objective: To perform Black Box testing of the Delete Poll functional requirement
Testing: I will perform Black Box testing to test if after Delete Poll button is clicked if poll disappears from My Polls Page and Community Polls Page for creator and other users. I will also test if user who did not create poll can see delete button (they shouldn't).

Action	**Result**
Creator Click Delete Button	Poll is deleted from Community Polls and My Polls page
Non-Creator doesn't see Delete Button	Non-Creator cannot see Delete poll button on Poll page
Non-Creator cannot see Deleted Poll	Non-Creator cannot see Deleted Poll on Community Polls page

My results show that our functional requirement is performing as planned.

4) Requirement 3.2.18 Cancel Poll Early - Casey Cook

Objective: To perform Black Box Testing of the Cancel Poll Early functional requirement.

Testing: I will perform Black Box testing to test if a Creator can cancel a poll early so users can only see the results of the poll and can no longer vote in it. I will also test to see if non creator can not see the cancel poll button (the shouldn't see it).

Action	Result
Creator Click Cancel Poll Button	Poll only shows results and can no longer be voted in
Non-Creator doesn't see Cancel Poll button	Non-Creator cannot see Cancel poll button on Poll page
Non-Creator cannot vote in canceled poll	Non-Creator just sees results of poll and cannot vote in it

My results show that our functional requirement is performing as planned.

5) Requirement 3.2.1 Create Account – Kevin Dalle

Objective: To perform Black Box Testing of the Create Account functional requirement.

Testing: I will perform black box testing on creating a new account. I will test whether someone can create an account that already exists as well.

Action	Result
Creator new Account	New user was successfully logged in and the user information was entered into the database
Try to create an account that already exists	User was logged in using their credentials instead of creating a new account

My results show that our functional requirement is performing as planned.

6) Requirement 3.2.2 Login - Kevin Dalle

Objective: To perform Black Box Testing of the Login functional requirement.

Testing: I will perform Black Box testing to test if a user can login with valid credentials. I will also test if they can login with completely incorrect credentials, and with correct username but wrong password.

Action	Result
Valid User logins	Was able to successfully login with the credentials of a valid user.
Invalid User logins	No user is logged in and the webpage reverts back to the home page. This is as expected.

Wrong Password	User was not logged in and the website returned to the homepage as expected.

7) Requirement 3.2.3 Anonymously Vote in Poll —– Nick Messina
Objective: Successfully vote in poll while not logged in as defined user. Make sure one vote increases vote count by one.
Testing: Perform Black Box testing to see if a an anonymous user can vote in poll, and the vote increases total votes by 1

Action	Result
Clic a Poll to vote in on the community polls page	Brings user to the specified poll's age
Cast "Yes" Vote in poll	Poll Results Displayed. "Yes" vote is incremented. Vote count increased by one

My results show that our functional requirement is performing as planned.
8) Requirement 3.2.5 View Poll Statistics– Nick Messina
Objective: go to poll page of poll previously voted in. Successfully view vote count, poll author, and additional info.
Testing: Perform Black Box testing to see if I can view all applicable information when viewing the page of a poll previously voted in by the respective user.

Action	Result
Click Poll on Community page that user previously voted in	Brings user to the specified poll's age
Check to see if Poll Creator, Poll Count, Additional info is displayed	"Poll Created by: Noah Data: 1/5 Votes Cast, Poll is Ongoing lditional Info: "I really need help with my math omework. If you could explain how you got it that would be great! "

My results show that our functional requirement is performing as planned.
9) Requirement 3.2.23 Report Offensive Poll– Matthew Fletcher

Objective: To perform Black Box testing of the Report Offensive Poll functional requirement
Testing: I will perform Black Box testing to see if our app responds correctly to a user reporting an offensive poll. There will be three parts to this test that will be shown sequentially in the chart below. To perform the **first part** of this test, the user will need to log in as an admin an be on the page that shows the list of reported polls. Look to see that the poll that is going to be reported in the second part either doesn't exists or check to see it's frequency. To perform the **second part** of the test, the tester will need to be logged in as a registered user and on the webpage for any particular poll. For the **third part** of the test, the tester needs to log in as an admin and check the reported polls.

Action	Result

Part 1: Log in as an admin.	The list of reported polls should no include the poll to be reported in part 2. If it does, keep track of the number of times it's already been reported.
Part 2: Click the "Report" button on the page where a registered user can take a poll.	The poll page should refresh and look the same as before the "Report" button was clicked.
Part 3: Refresh the admin page with reported polls or relogin as an admin.	If the poll reported in part 2 hadn't already been reported in part 1, the poll should now appear in the list of reported polls with a frequency of '1'. If the poll reported in part 2 had already been reported in part 1, the frequency of reports for that poll should have increased by 1.

Based on this information, the functional requirement is performing as required.

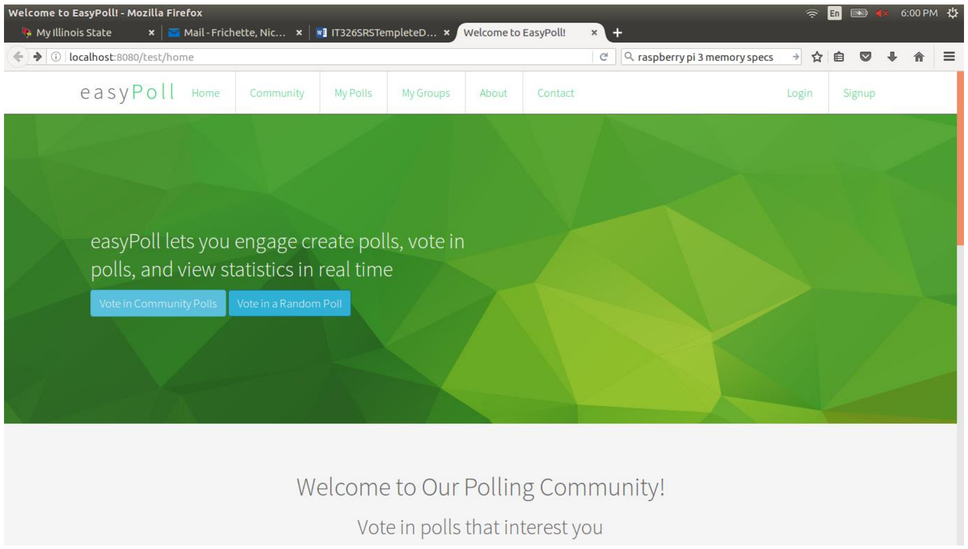
10) Requirement 3.2.31 Add Users to Group– Matthew Fletcher

Objective: To perform Black Box testing of the Report Offensive Poll functional requirement

Testing: To perform the Black Box testing, the tester will need to log in as a user that has at least one poll group already to complete **part 1**. There will be two parts to this test. The **first part** will be a user adding a user to a group. The **second part** will be logging in as the newly added user and checking to see if they were successfully added to the group.

Action	Result
Part 1: INside one of the user's groups, enter in a username that exists in the system into the "Username to Add" field next to the "Add User" button. Proceed by clicking the "Add User" button.	The user that submitted the username should immediately see username added to a list of users that are part of the group, displayed directly below the "Username to Add" field.
Part 2: After logging in as the newly added user, click the "My Groups" tab.	In the list of groups that they are a part of should be the corresponding group for the user that just added them.

4.4.4.5 Screenshots

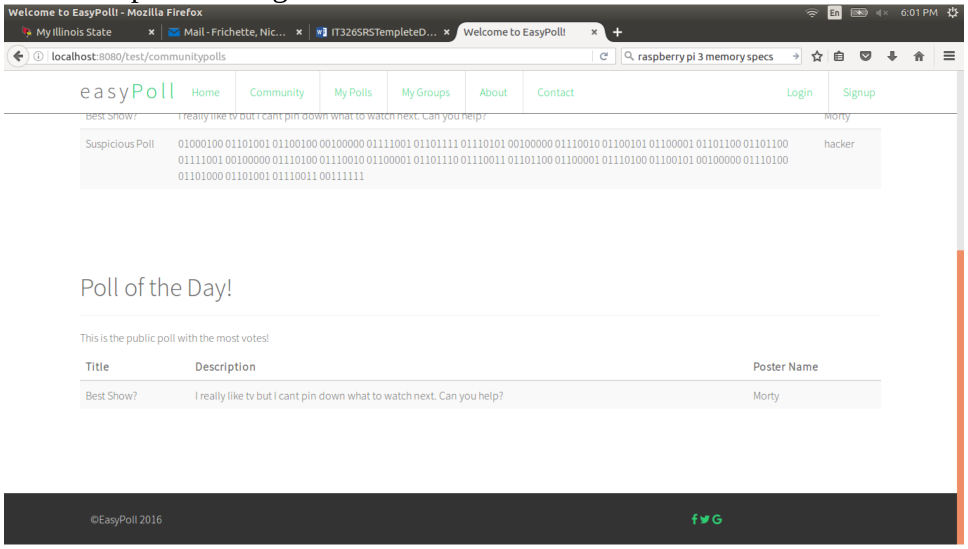


easyPoll lets you engage create polls, vote in polls, and view statistics in real time

Welcome to Our Polling Community!

Vote in polls that interest you

Sprint 3 Image 1: Home Screen With Random Poll button



Best Show?	I really like tv but I cant pin down what to watch next. Can you help?	Morty
Suspicious Poll	01000100 01101001 01100100 00100000 01111001 01101111 01110101 00100000 01110010 01100101 01100001 01101100 01101100 01111001 00100000 01110100 01110010 01100001 01101110 01110011 01101100 01100001 01110100 01100101 00100000 01110100 01101000 01101001 01110011 00111111	hacker

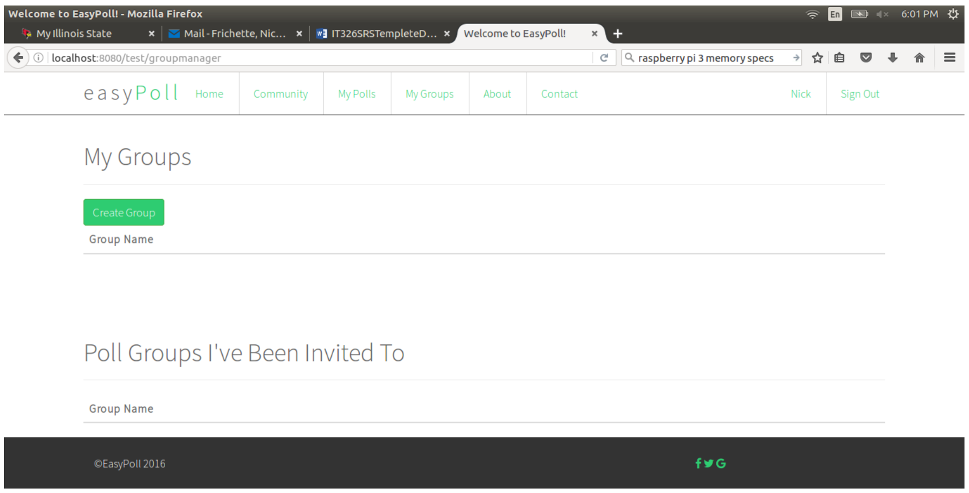
Poll of the Day!

This is the public poll with the most votes!

Title	Description	Poster Name
Best Show?	I really like tv but I cant pin down what to watch next. Can you help?	Morty

©EasyPoll 2016

Sprint 3 Image 2: Poll of the Day In Community Polls



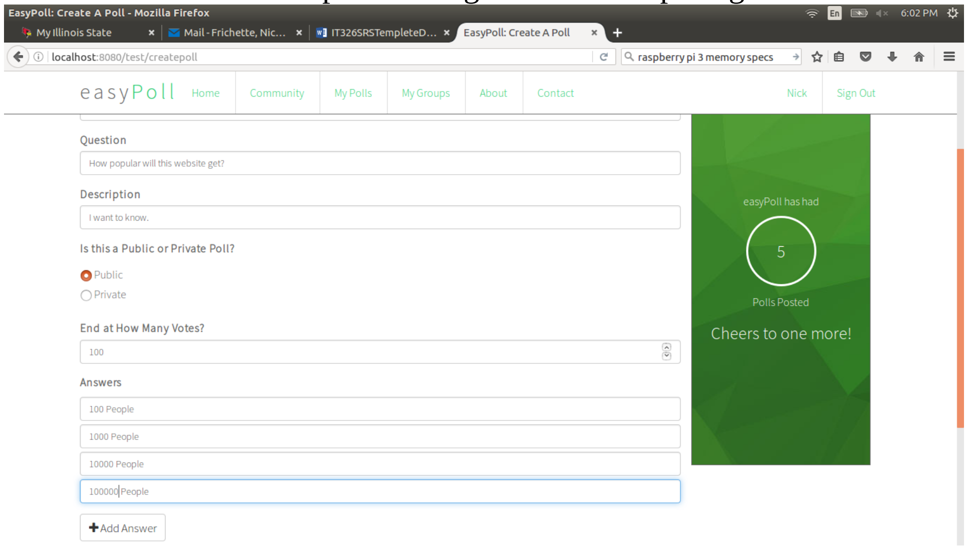
My Groups

Group Name

Poll Groups I've Been Invited To

Group Name

Sprint 3 Image 3: Poll Groups Page



Question: How popular will this website get?

Description: I want to know.

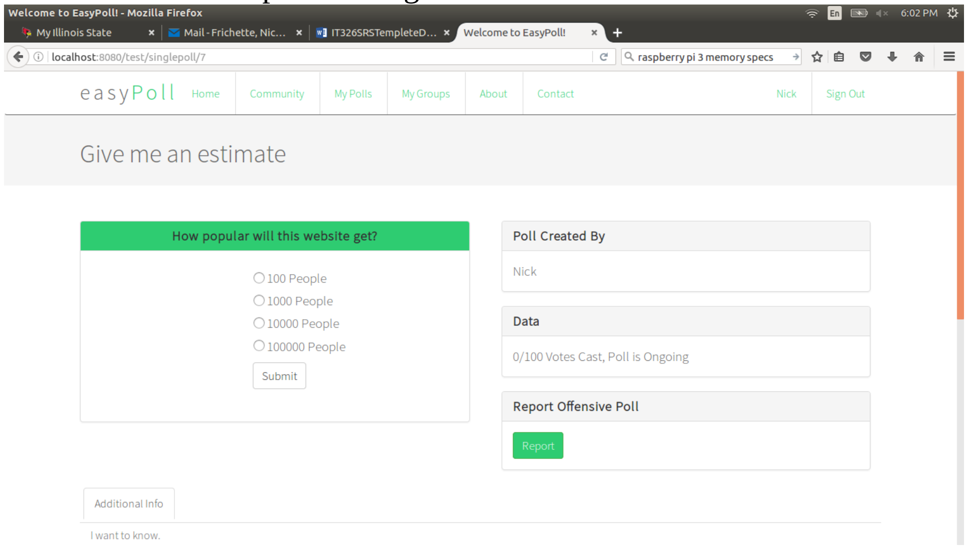
Is this a Public or Private Poll?

[✓ Public]
Private

End at How Many Votes? 100

Answers
100 People
1000 People
10000 People
100000 People

Sprint 3 Image 4: Create Poll With End Goal



Give me an estimate

How popular will this website get?

100 People
1000 People
10000 People
100000 People

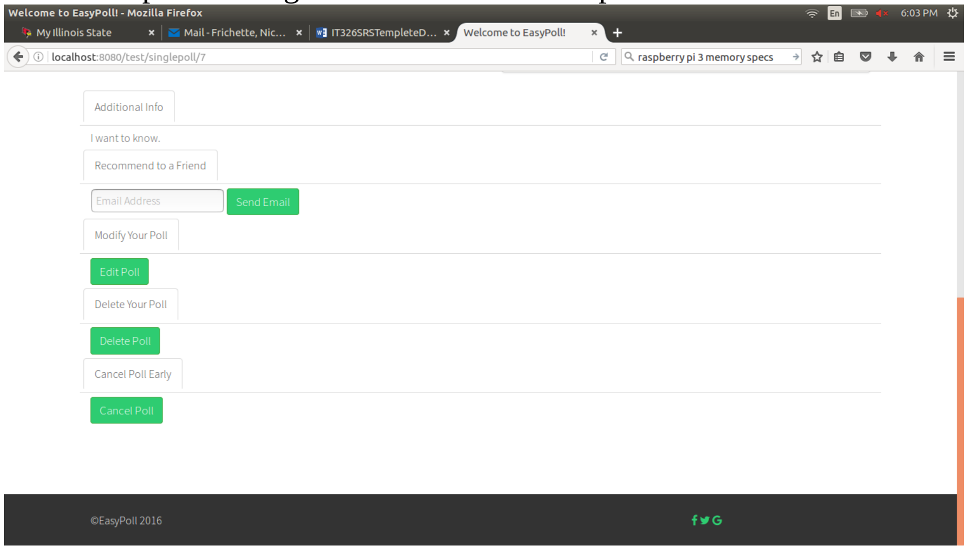
Poll Created By: Nick

Data: 0/100 Votes Cast, Poll is Ongoing

Report Offensive Poll

I want to know.

Sprint 3 Image 5: View Poll With Report Offensive Poll Button


Welcome to EasyPoll! - Mozilla Firefox
6:03 PM
My Illinois State
Mail - Frichette, Nic...
IT326SRSTempleteD...
Welcome to EasyPoll!
localhost:8080/test/singlepoll/7
raspberry pi 3 memory specs

Additional Info

I want to know.

Recommend to a Friend

Email Address | Send Email

Modify Your Poll

Edit Poll

Delete Your Poll

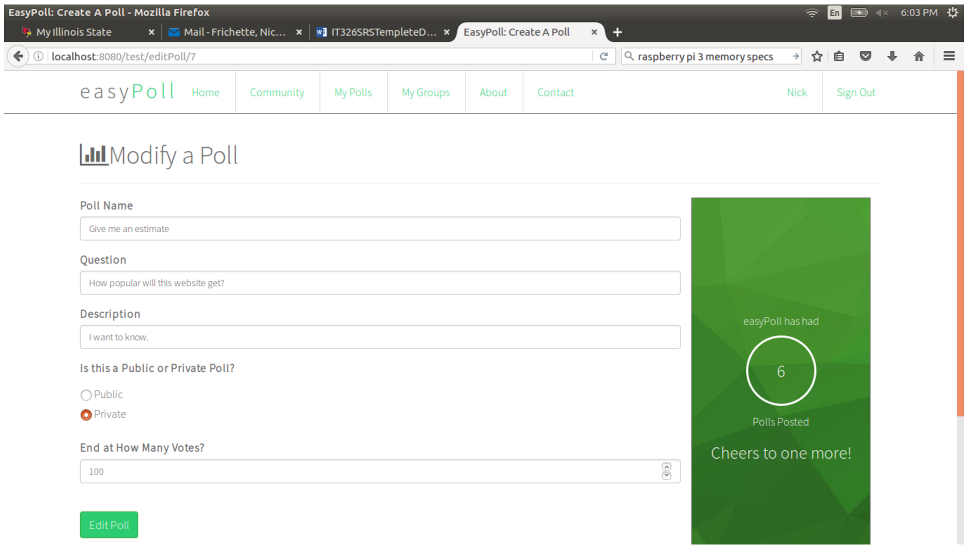
Delete Poll

Cancel Poll Early

Cancel Poll

©EasyPoll 2016

Sprint 3 Image 6: View Poll With Edit Poll, Delete Poll, Cancel Poll, and Recommend Poll



Modify a Poll

Poll Name: Give me an estimate

Question: How popular will this website get?

Description: I want to know.

Is this a Public or Private Poll?

Public [✓ Private]

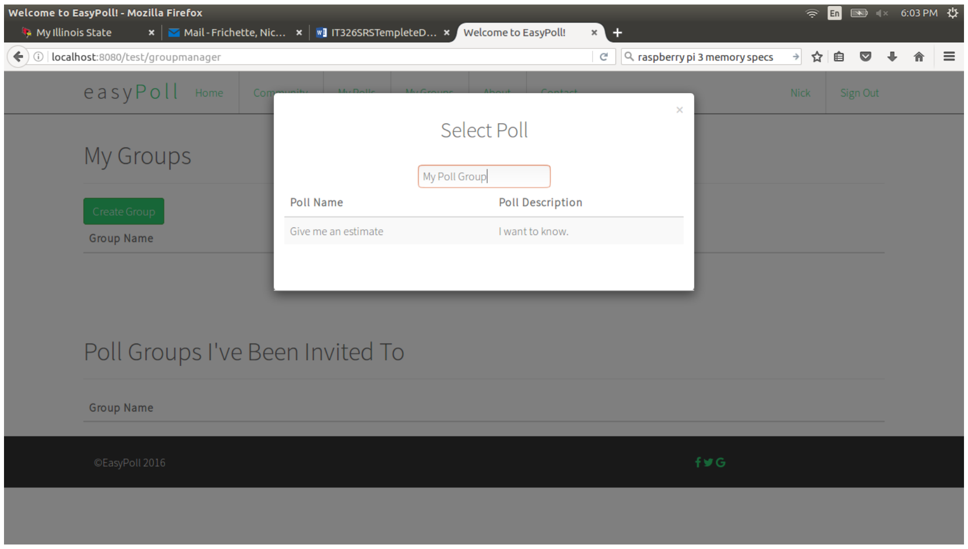
End at How Many Votes? 100

Edit Poll

easyPoll has had 6 Polls Posted

Cheers to one more!

Sprint 3 Image 7: Edit Poll Page

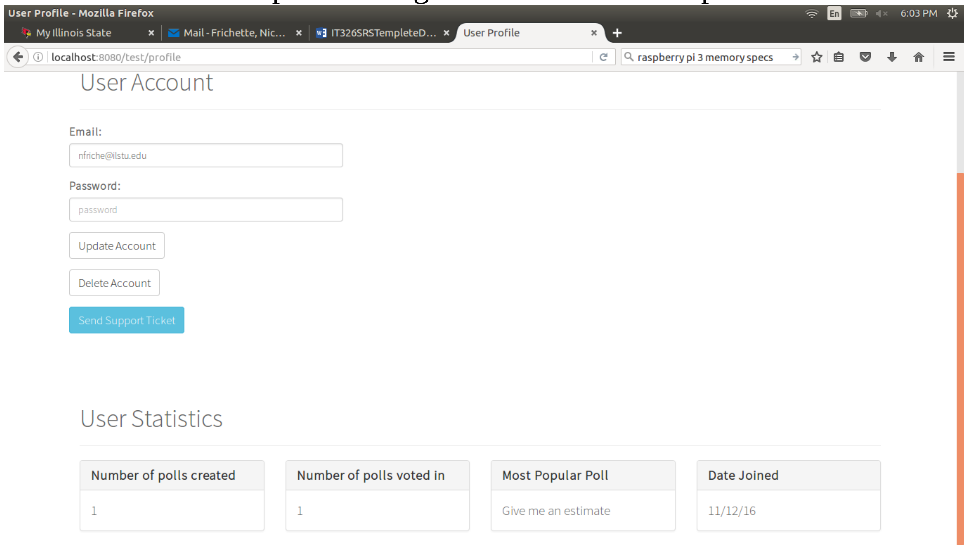

Welcome to EasyPoll! - Mozilla Firefox
localhost:8080/test/groupmanager
easyPoll
Home
Nick
Sign Out
My Groups
Create Group
Group Name
Poll Groups I've Been Invited To
Group Name
©EasyPoll 2016

Select Poll

My Poll Group

Poll Name	Poll Description
Give me an estimate	I want to know.

Sprint 3 Image 8: Create Poll Group Modal



User Account

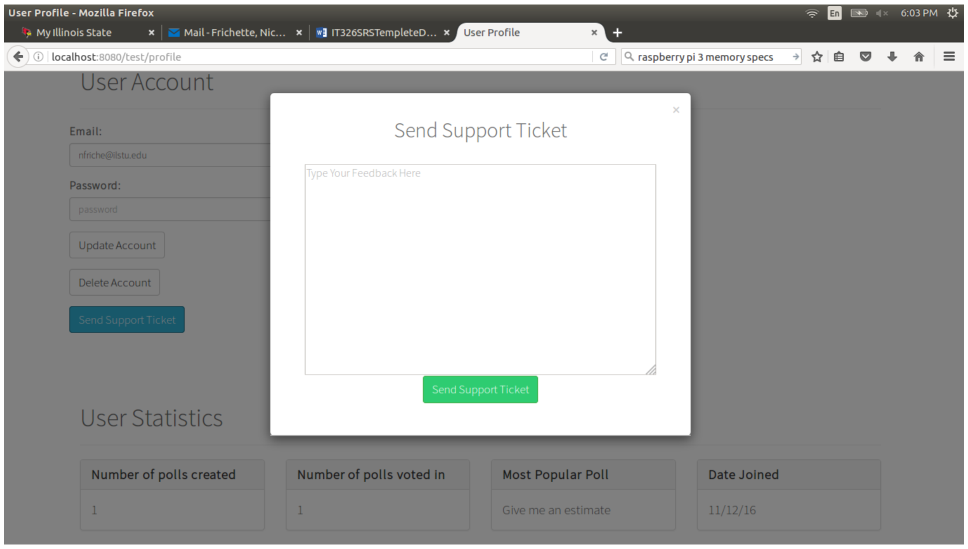
Email: nfriche@ilstu.edu

Password: password

User Statistics

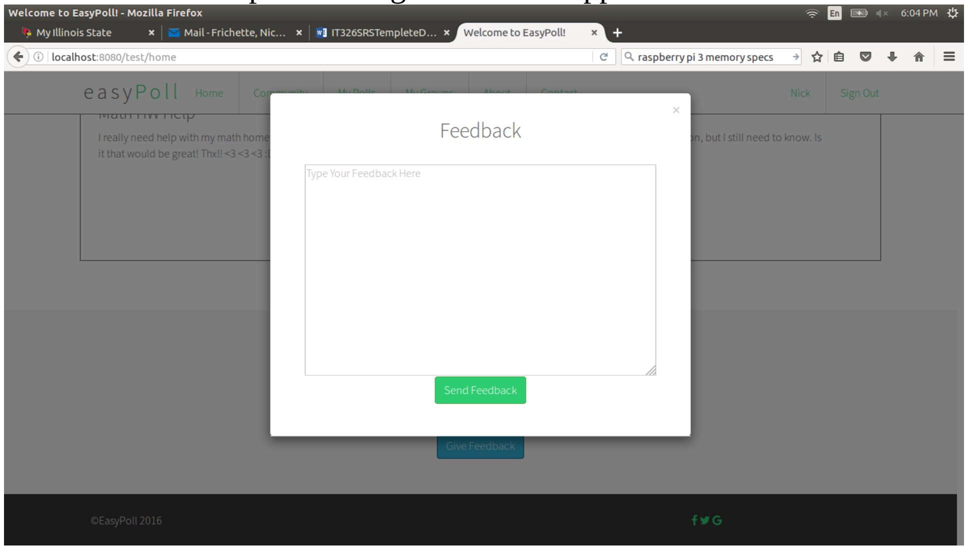
Number of polls created	Number of polls voted in	Most Popular Poll	Date Joined
1	1	Give me an estimate	11/12/16

Sprint 3 Image 9: Profile Page with User Statistics, Edit Account, and Send Support Ticket Buton

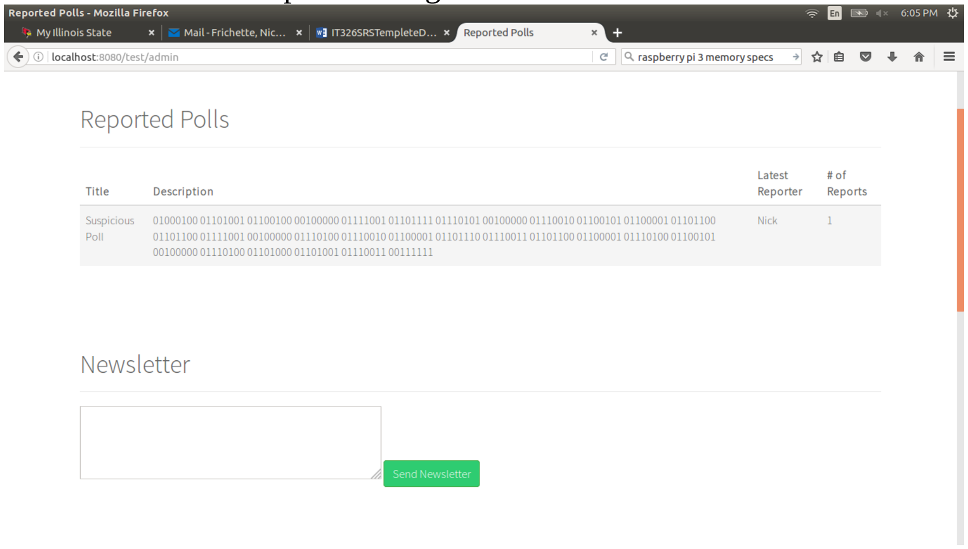


Number of polls created	Number of polls voted in	Most Popular Poll	Date Joined
1	1	Give me an estimate	11/12/16

Sprint 3 Image 10: Send Support Ticket Modal



Sprint 3 Image 11: Send Feedback Modal

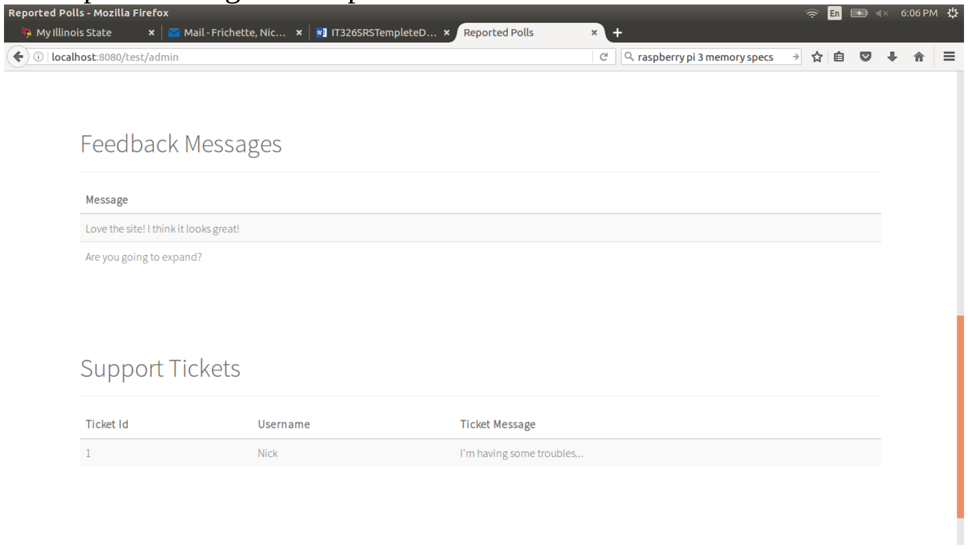


Reported Polls

Title	Description	Latest Reporter	# of Reports
Suspicious Poll	01000100 01101001 01100100 00100000 01111001 01101111 01110101 00100000 01110010 01100101 01100001 01101100 01101100 01111001 00100000 01110100 01110010 01100001 01101110 01110011 01101100 01100001 01110100 01100101 00100000 01110100 01101000 01101001 01110011 00111111	Nick	1

Newsletter

Sprint 3 Image 12: Reported Polls and Newsletter Build In Admin Portal


Reported Polls - Mozilla Firefox
My Illinois State
Mail - Frichette, Nic...
IT326SRSTempleteD...
Reported Polls
localhost:8080/test/admin
raspberry pi 3 memory specs
6:06 PM

Feedback Messages

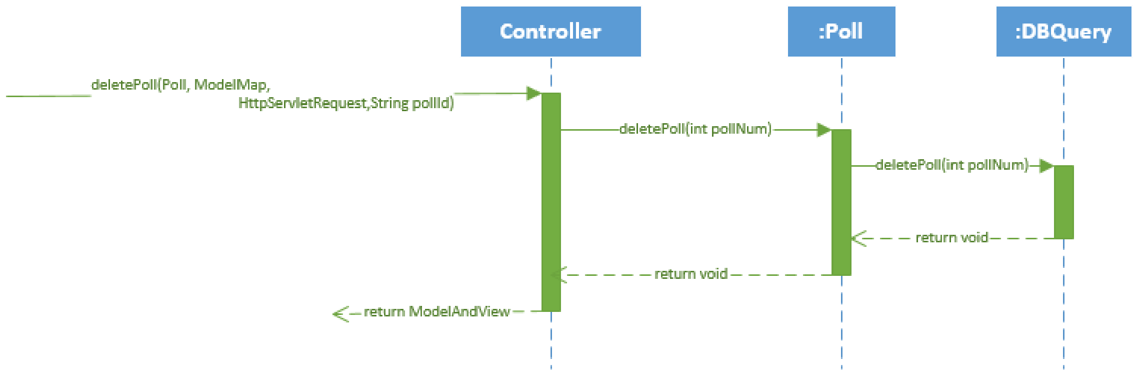
Message
Love the site! I think it looks great!
Are you going to expand?

Support Tickets

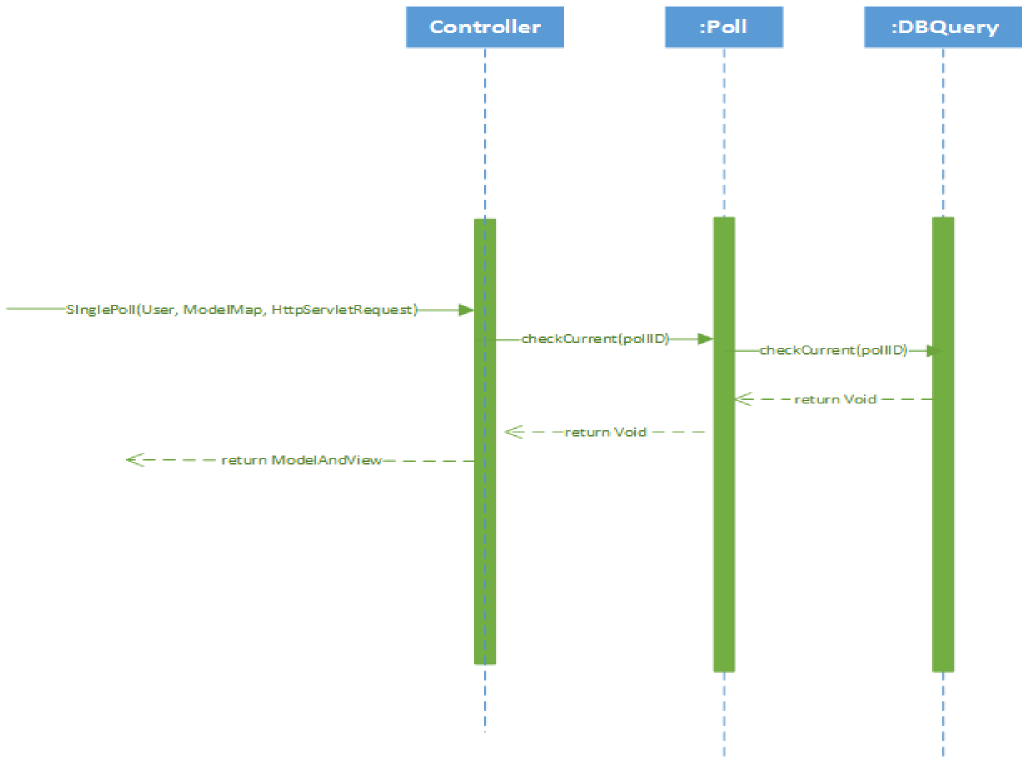
Ticket Id	Username	Ticket Message
1	Nick	I'm having some troubles...

Sprint 3 Image 13: Feedback Messages and Support Tickets in Admin Portal

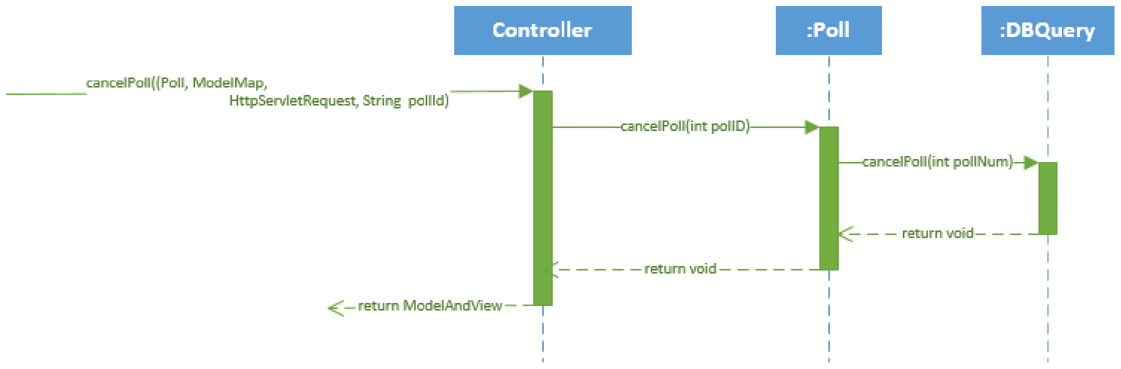
4.4.4.6 Sequence Diagrams



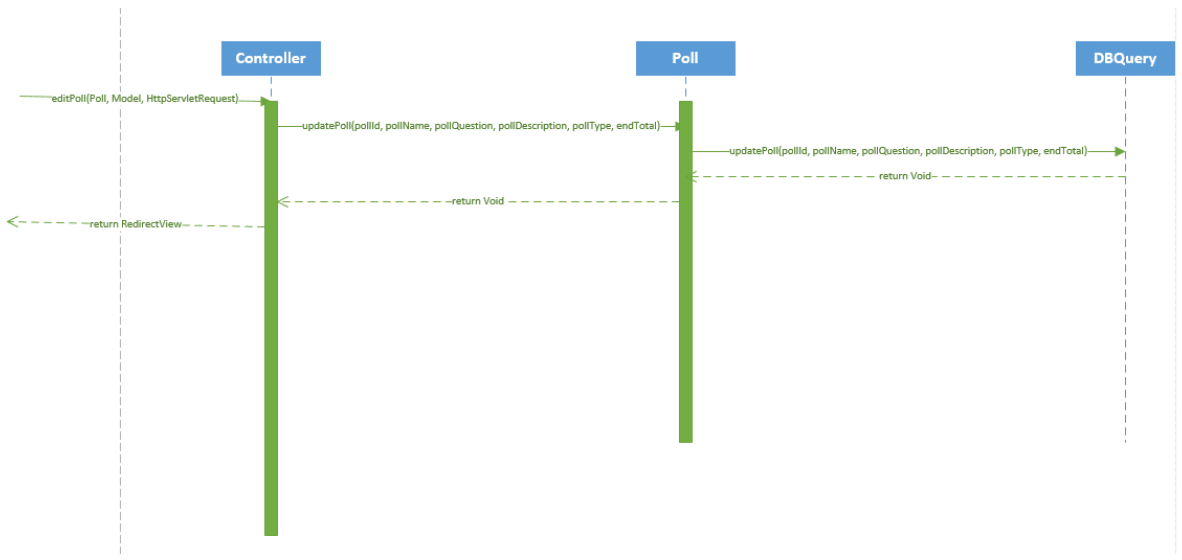
Sequence Diagram 1. Requirement 3.2.17 Delete Poll



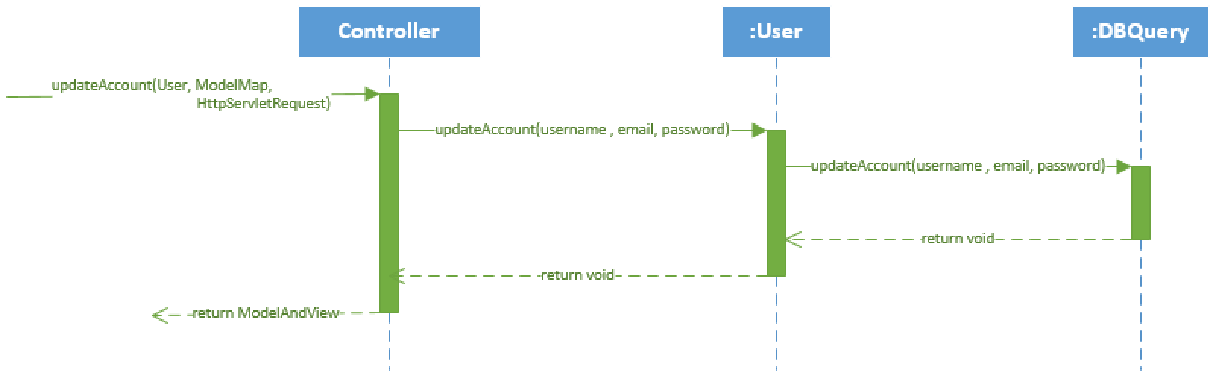
Sequence Diagram 2. Requirement 3.2.7 End Poll



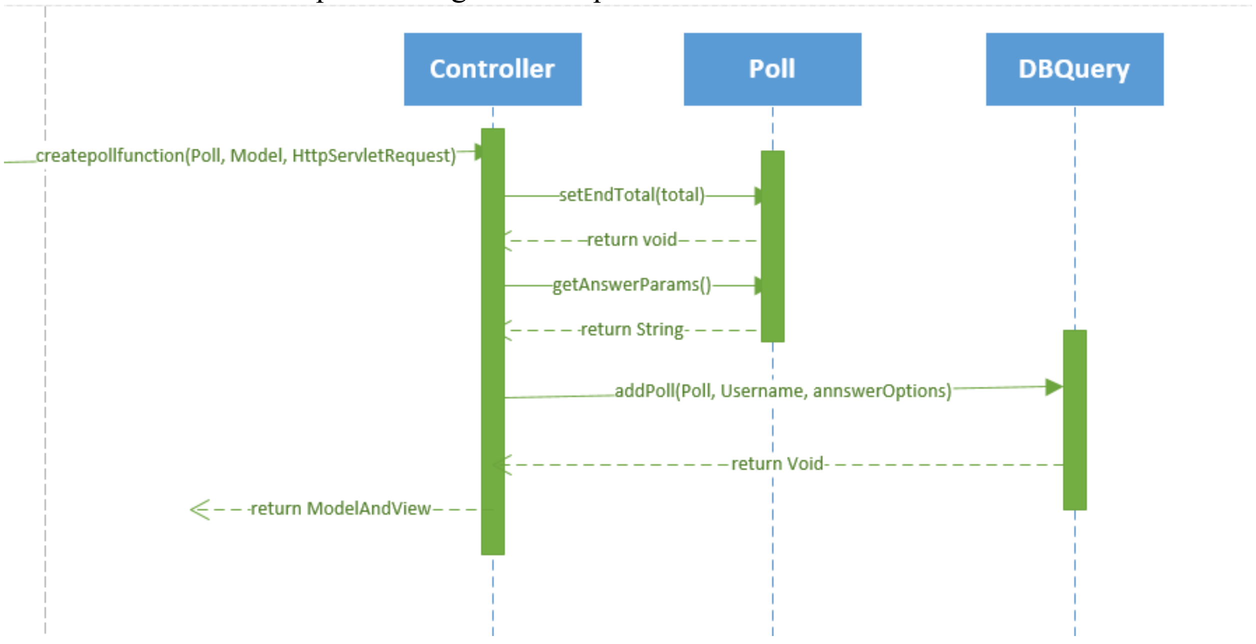
Sequence Diagram 3. Requirement 3.2.18 Cancel Poll Early


Controller
Poll
DBQuery
editPoll(Poll, Model, HttpServletRequest)
updatePoll(pollId, pollName, pollQuestion, pollDescription, pollType, endTotal)
updatePoll(pollId, pollName, pollQuestion, pollDescription, pollType, endTotal)
return Void
return Void
return RedirectView

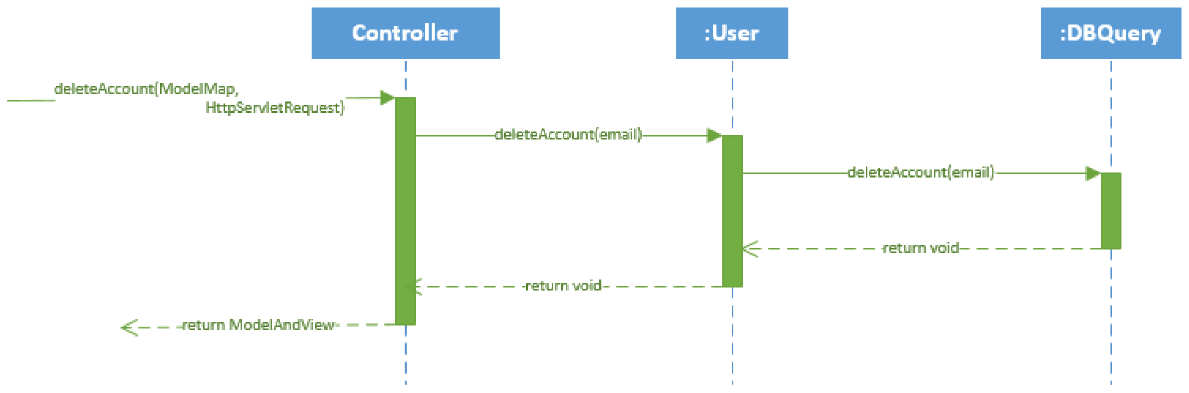
Sequence Diagram 4 Requirement 3.2.8 Edit Poll


Controller
:User
:DBQuery
updateAccount(User, ModelMap, HttpServletRequest)
updateAccount(username , email, password)
updateAccount(username , email, password)
return void
return void
return ModelAndView

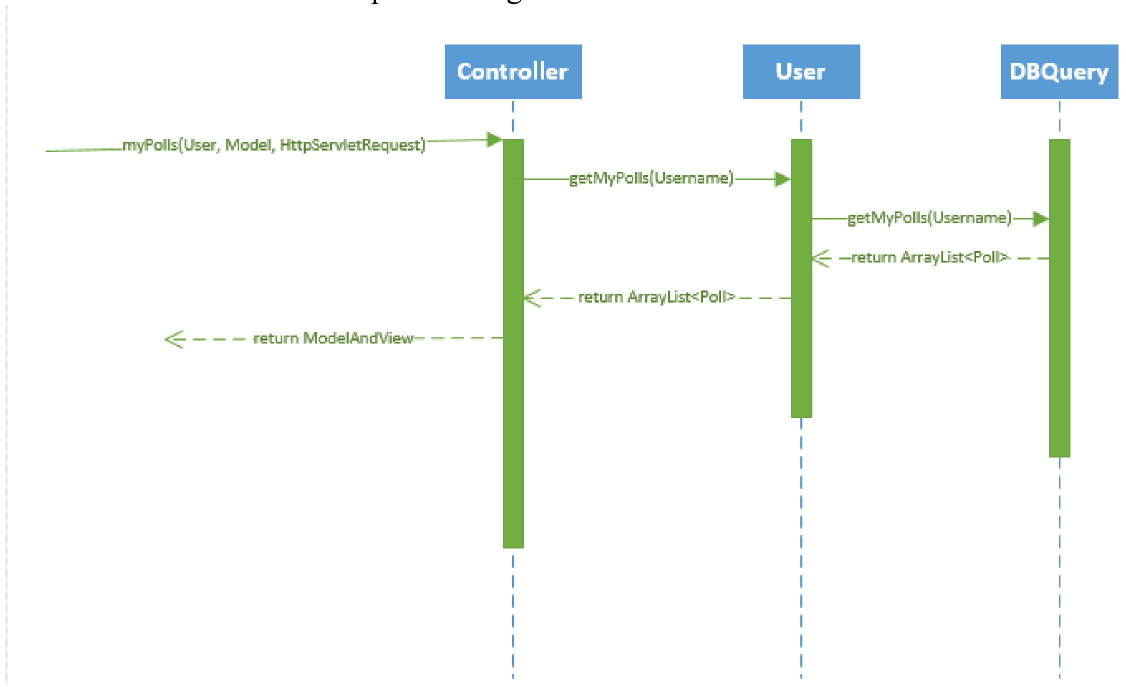
Sequence Diagram 5 Requirement 3.2.19 Edit Account



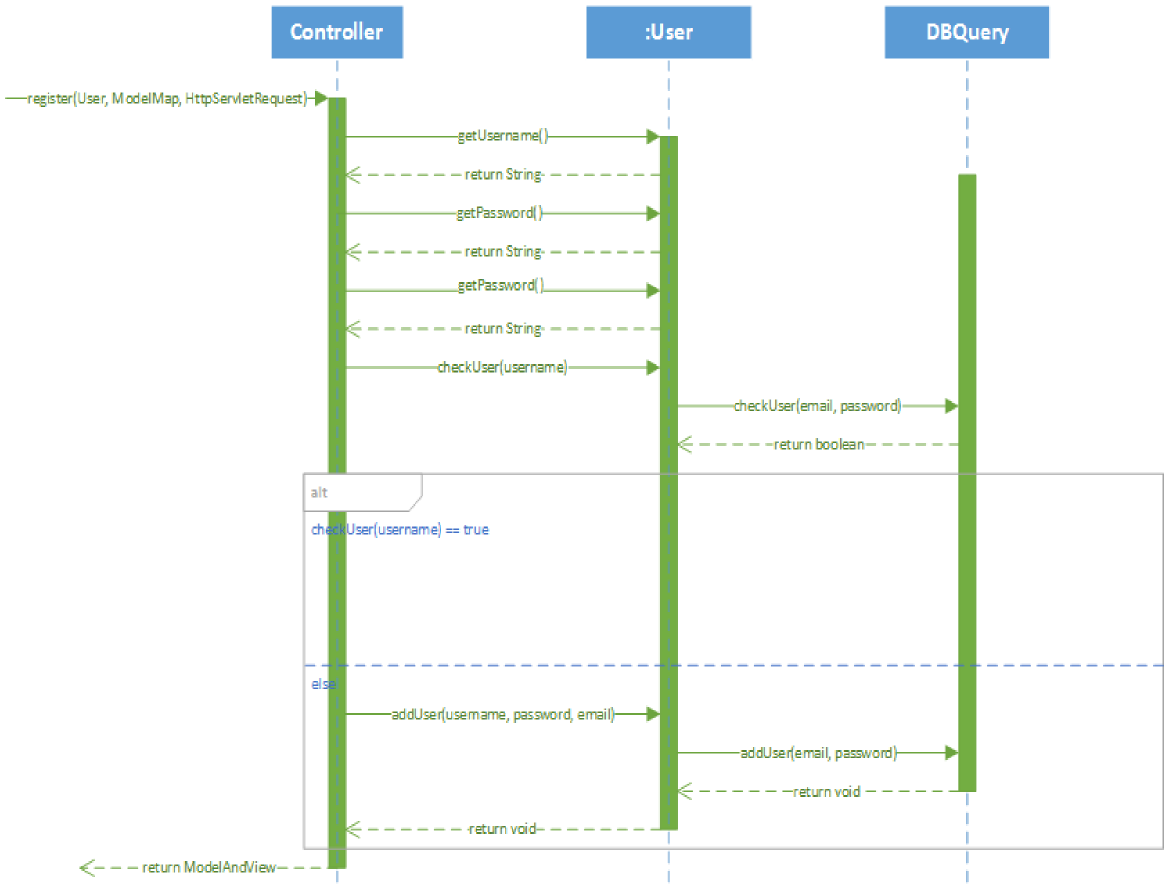
Sequence Diagram 6 3.2.9 Create Poll



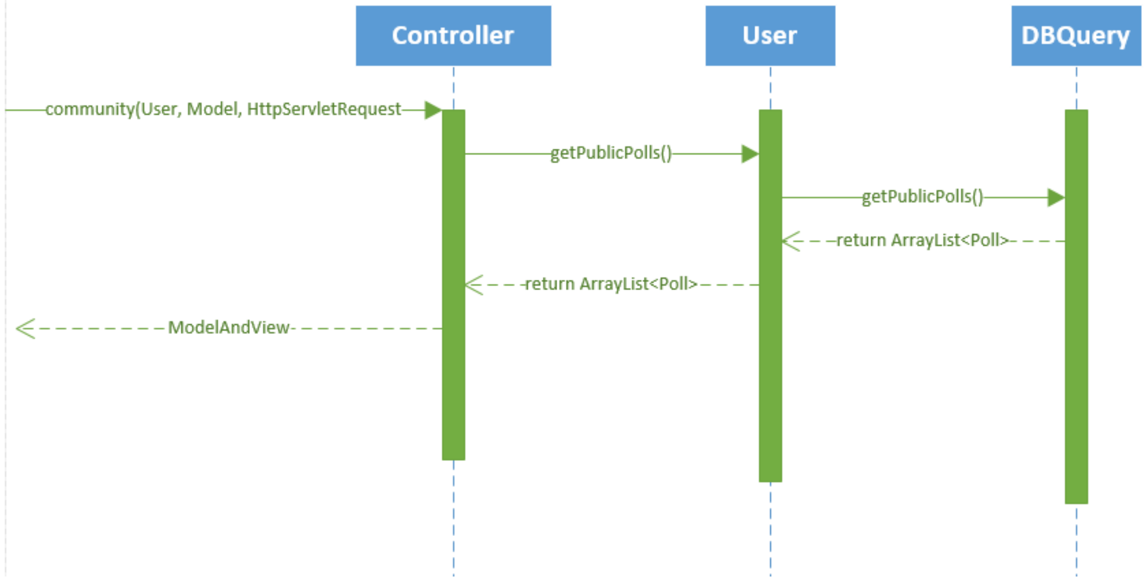
Sequence Diagram 7 3.2.20 Delete Account



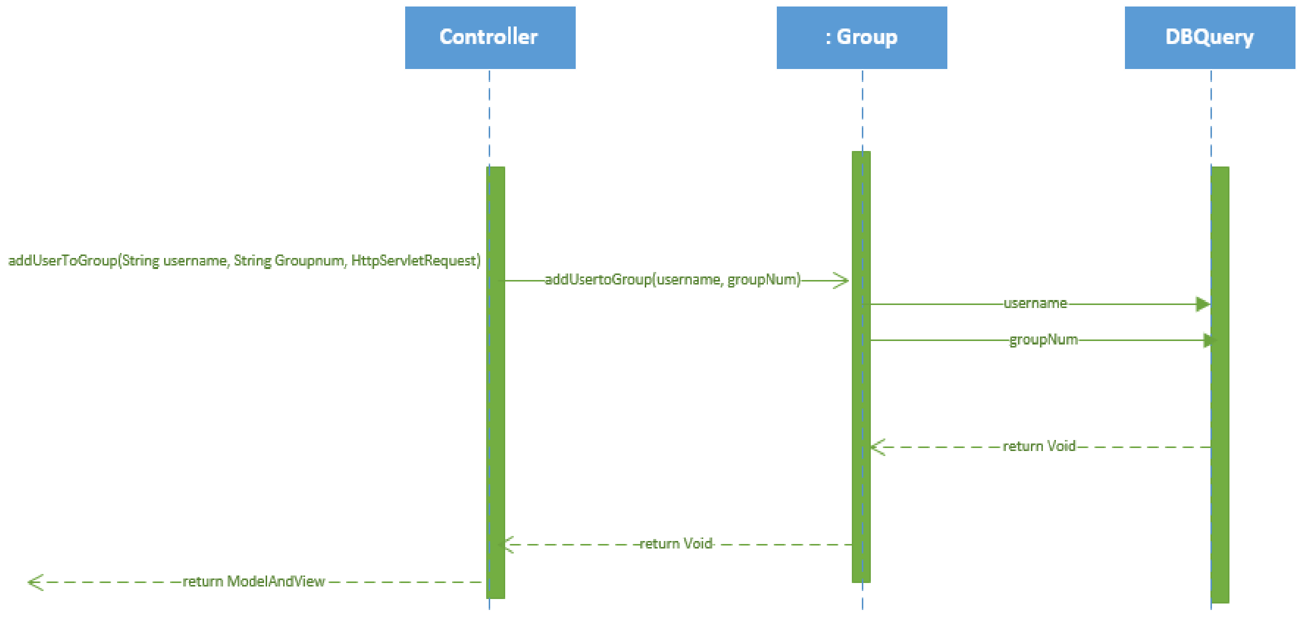
Sequence Diagram 8 3.2.10 View My Polls



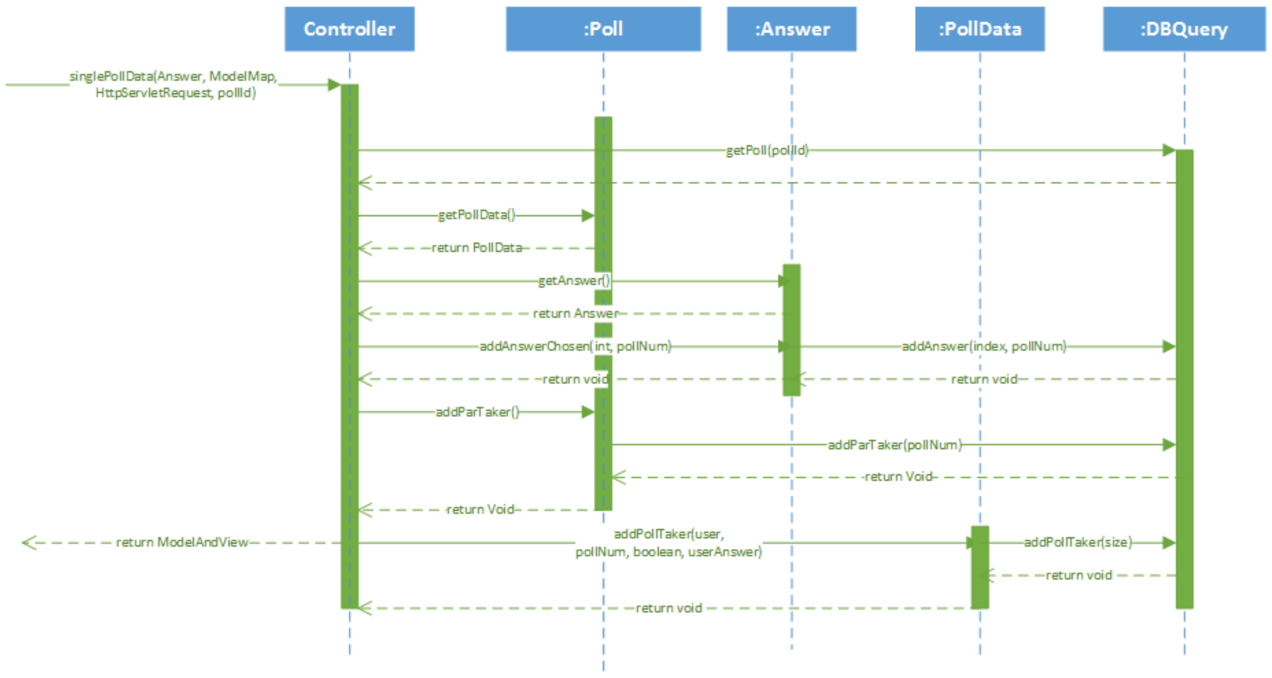
Sequence Diagram 9 3.2.1 Create Account



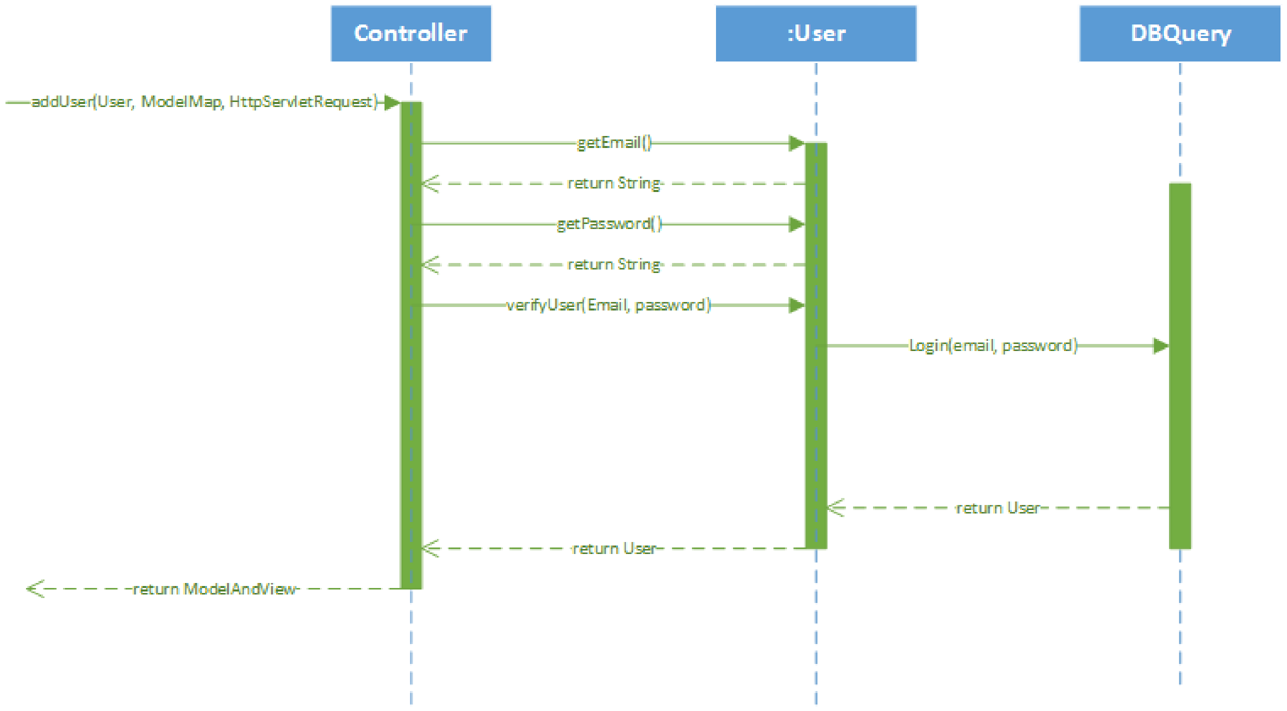
Sequence Diagram 10 3.2.11 View Community Polls


Controller
: Group
DBQuery
addUserToGroup(String username, String Groupnum, HttpServletRequest)
addUsertoGroup(username, groupNum)
username
groupNum
return Void
return Void
return ModelAndView

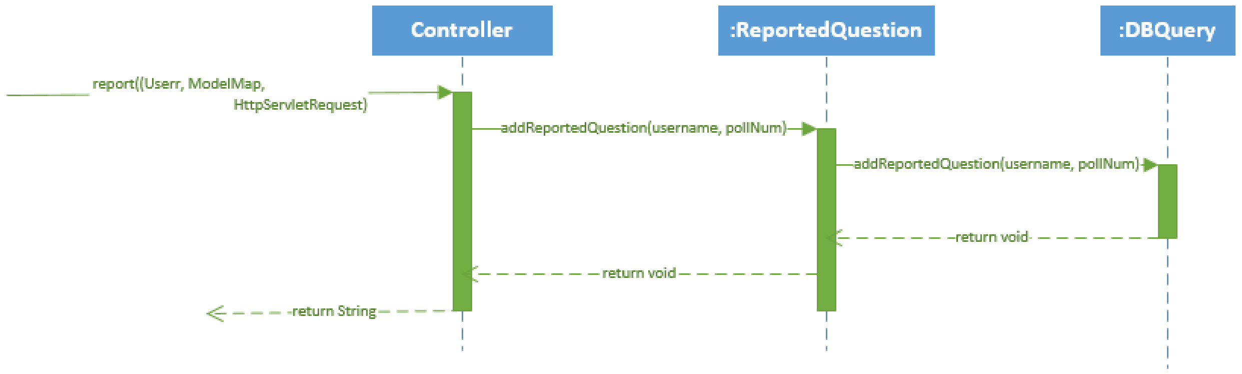
Sequence Diagram 11 3.2.31 Add User to Group



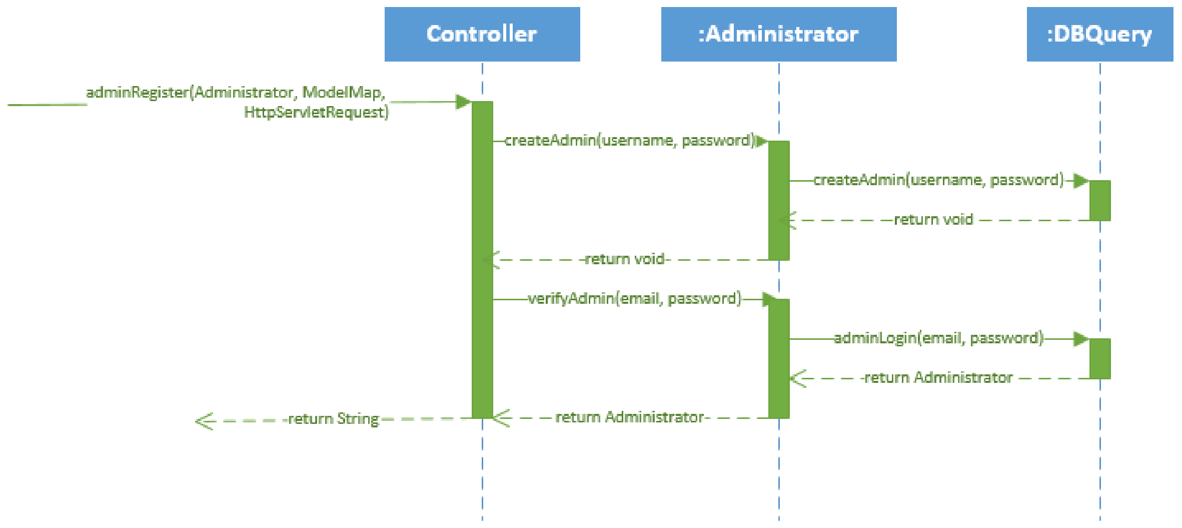
Sequence Diagram 12 3.2.15 Vote in Poll



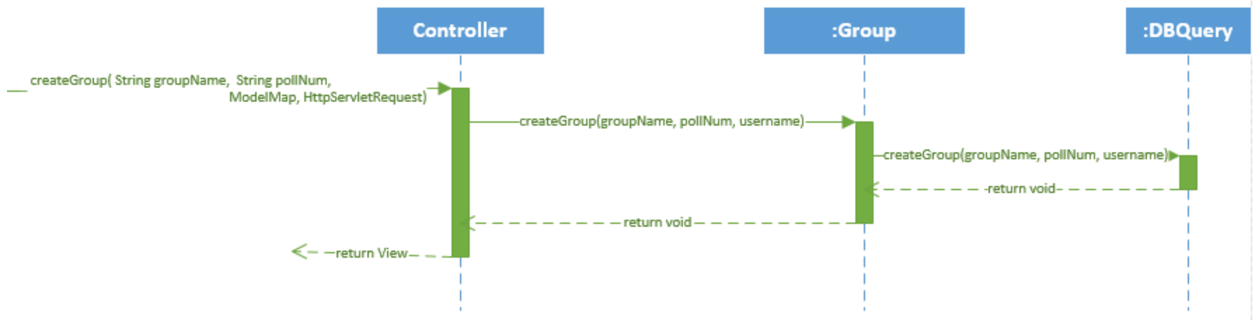
Sequence Diagram 13 3.2.2 Login


Controller
:ReportedQuestion
:DBQuery
report((Userr, ModelMap, HttpServletRequest)
addReportedQuestion(username, pollNum)
addReportedQuestion(username, pollNum)
return void
return void
return String

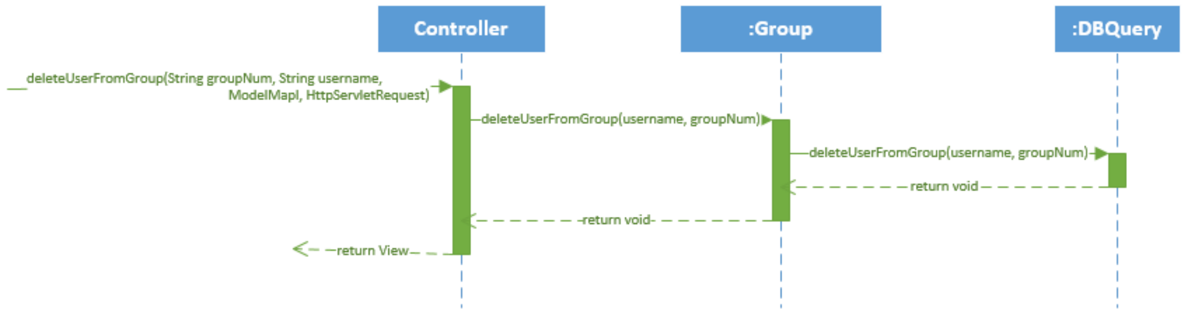
Sequence Diagram 14 3.2.23 Report Offensive Poll


Controller
:Administrator
:DBQuery
adminRegister(Administrator, ModelMap, HttpServletRequest)
createAdmin(username, password)
createAdmin(username, password)
return void
return void
verifyAdmin(email, password)
adminLogin(email, password)
return Administrator
return Administrator
return String

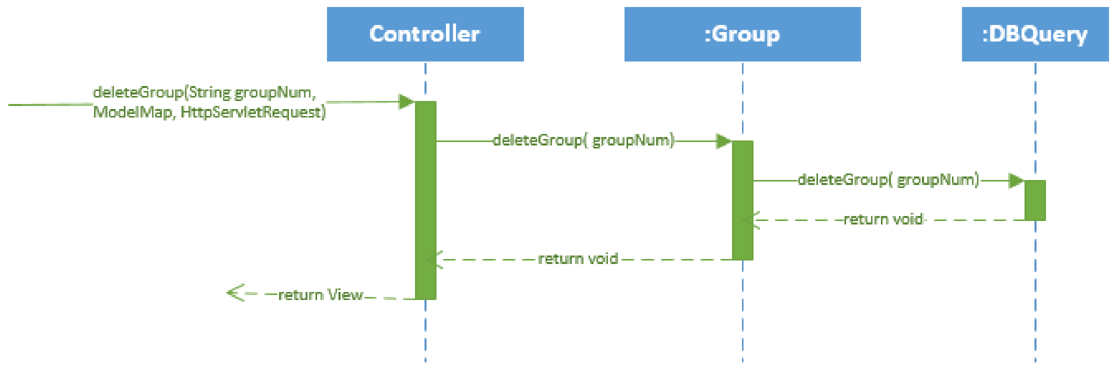
Sequence Diagram 15 3.2.24 Create Admin Account


Controller
:Group
:DBQuery
createGroup(String groupName, String pollNum, ModelMap, HttpServletRequest)
createGroup(groupName, pollNum, username)
createGroup(groupName, pollNum, username)
return void
return void
return View

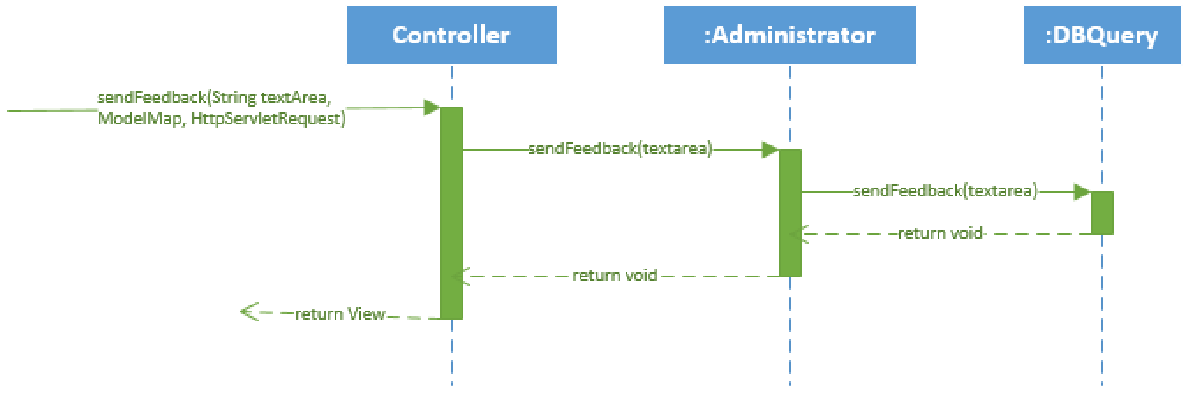
Sequence Diagram 16 3.2.30 Create Poll Groups


Controller
:Group
:DBQuery
deleteUserFromGroup(String groupNum, String username, ModelMapl, HttpServletRequest)
deleteUserFromGroup(username, groupNum)
deleteUserFromGroup(username, groupNum)
return void
return void
return View

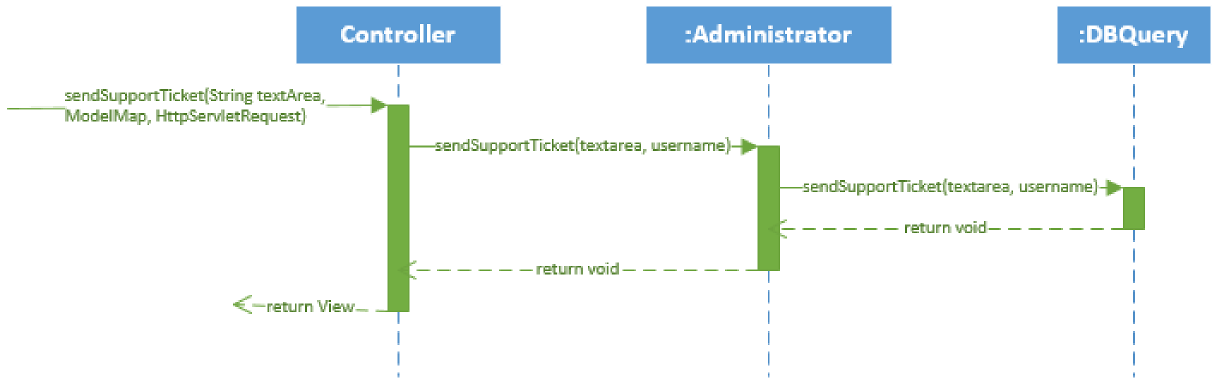
Sequence Diagram 17 3.2.32 Delete Users from Group



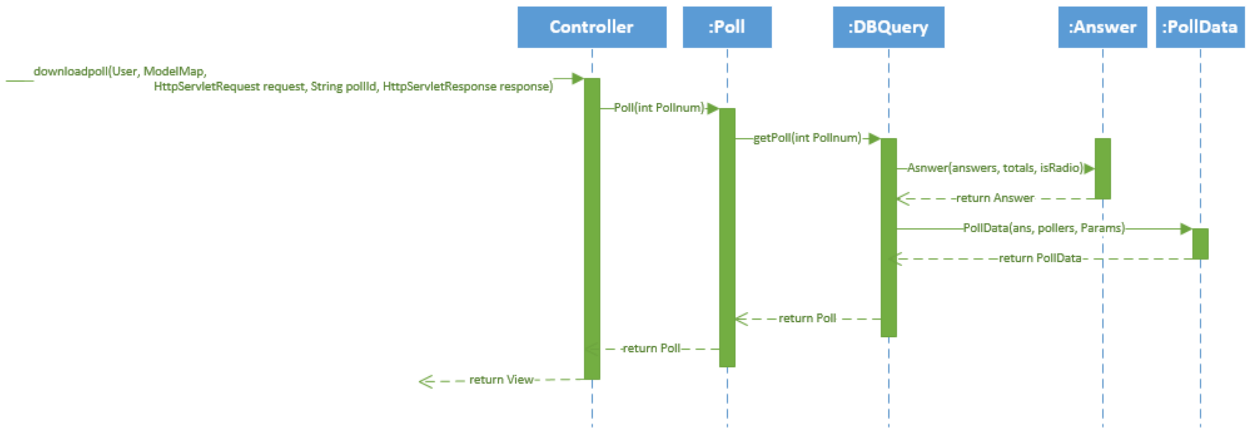
Sequence Diagram 18 3.2.33 Delete Group



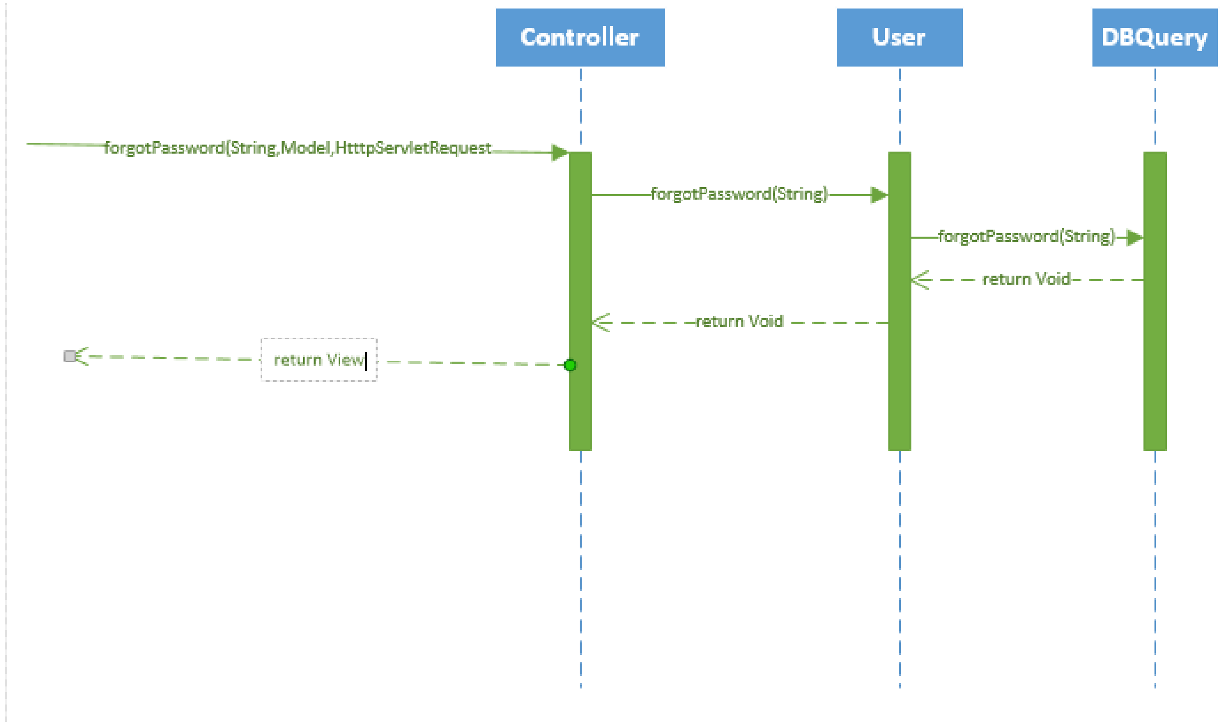
Sequence Diagram 19 3.2.35 Send Feedback


Controller
:Administrator
:DBQuery
sendSupportTicket(String textArea, ModelMap, HttpServletRequest)
sendSupportTicket(textarea, username)
sendSupportTicket(textarea, username)
return void
return void
return View

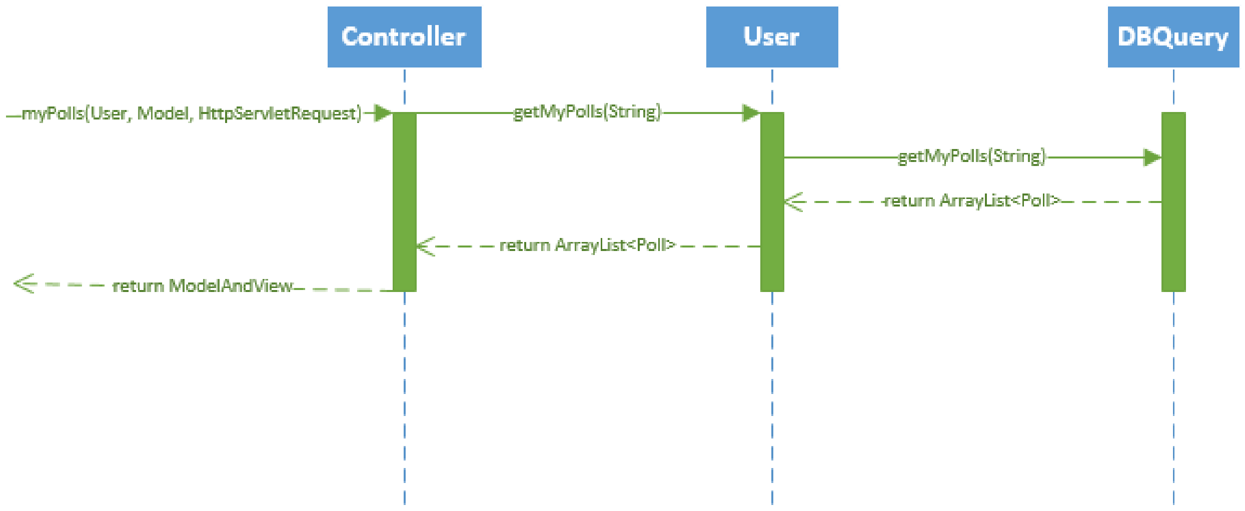
Sequence Diagram 20 3.2.34 Send Support Ticket


Controller
:Poll
:DBQuery
:Answer
:PollData
downloadpoll(User, ModelMap, HttpServletRequest request, String pollId, HttpServletResponse response)
Poll(int Pollnum)
getPoll(int Pollnum)
Asnwer(answers, totals, isRadio)
return Answer
PollData(ans, pollers, Params)
return PollData
return Poll
return Poll
return View

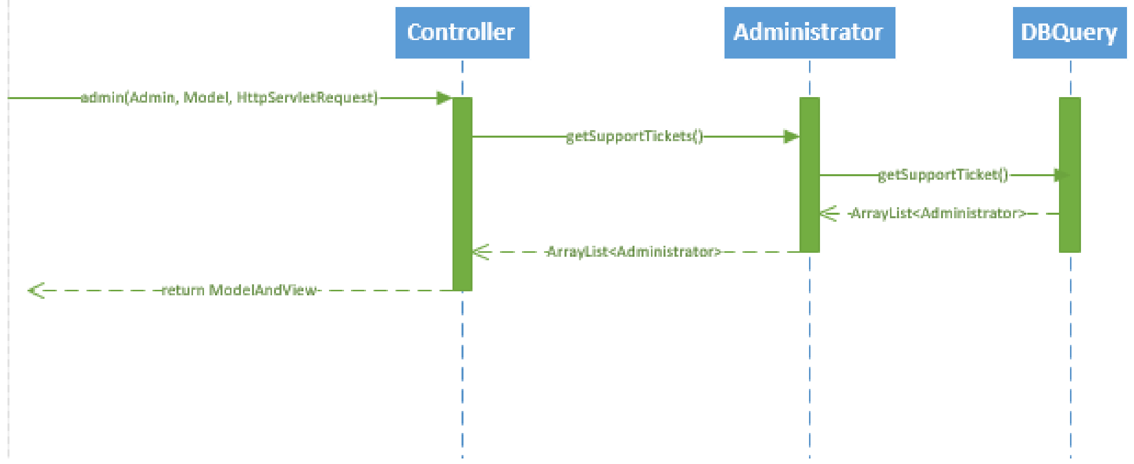
Sequence Diagram 21 3.2.21 Export Poll Data



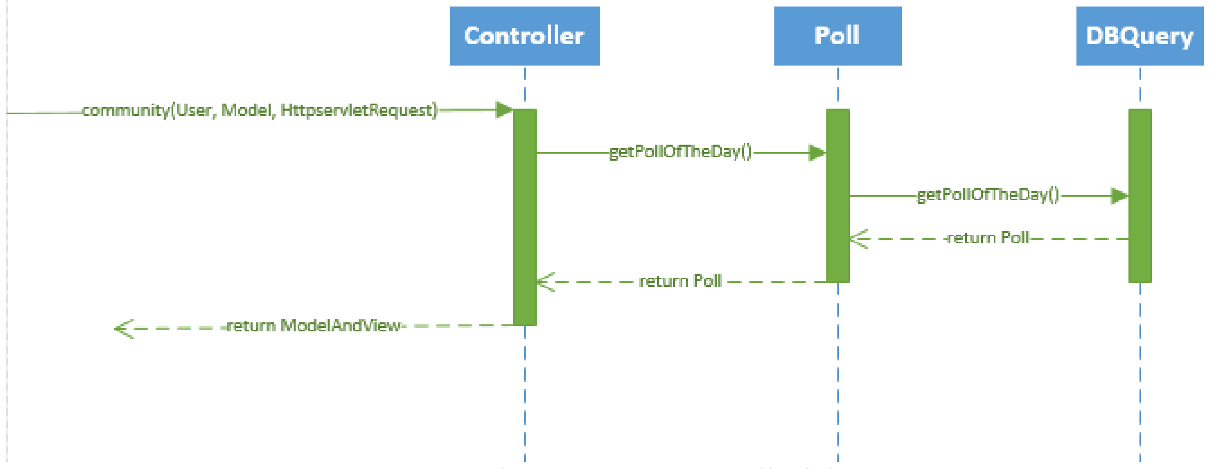
Sequence Diagram 22 3.2.25 Forgot Password Service



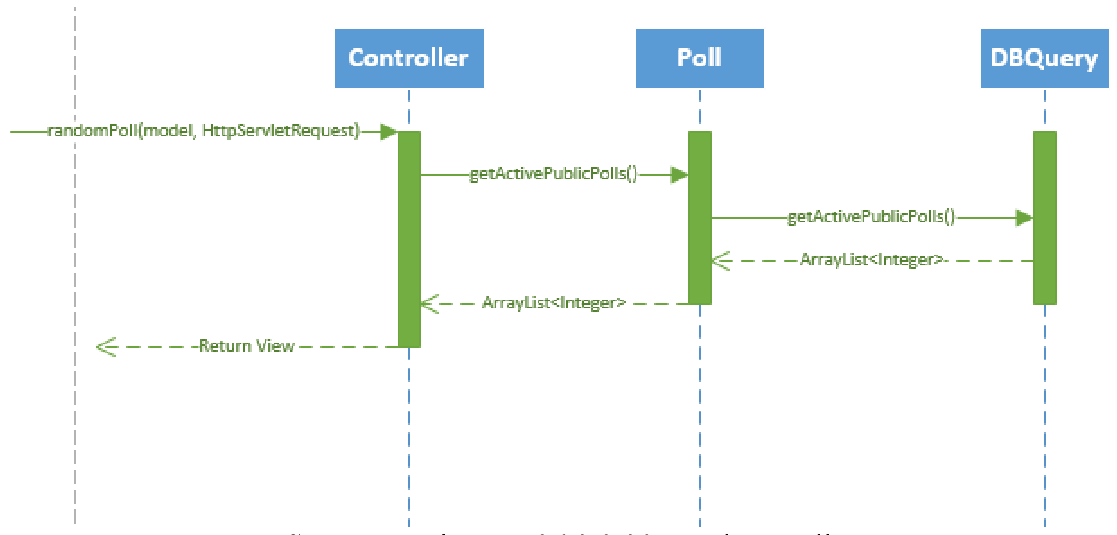
Sequence Diagram 23 3.2.12 View User Statistics



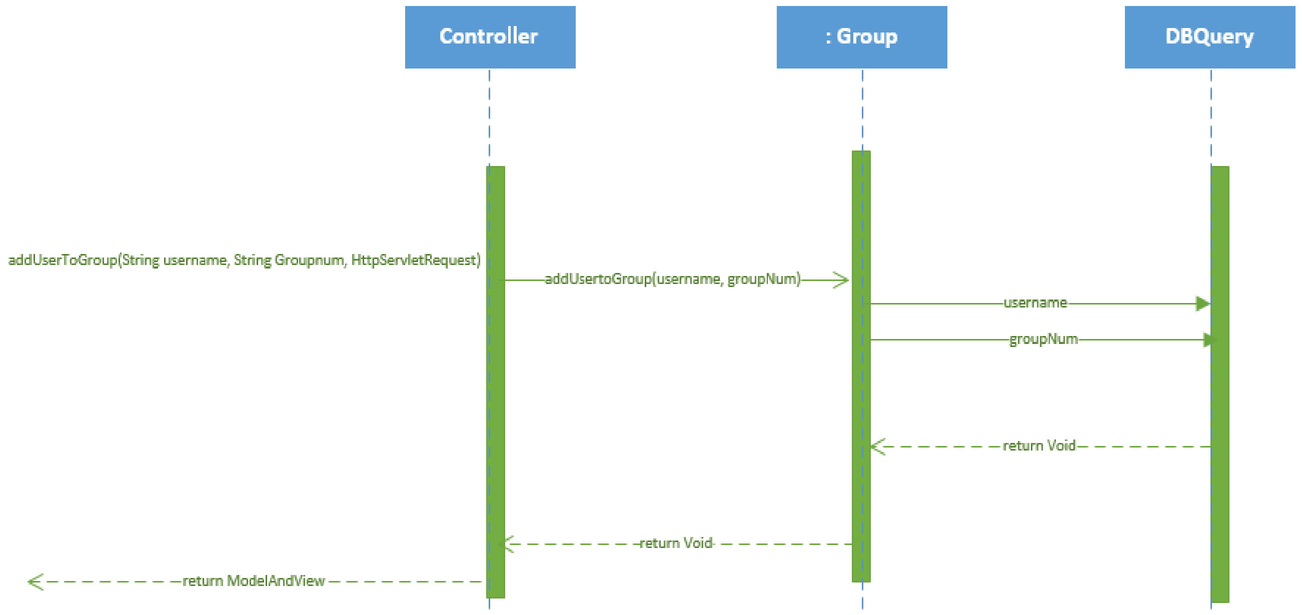
Sequence Diagram 24 3.2.37 View Reported Feedback



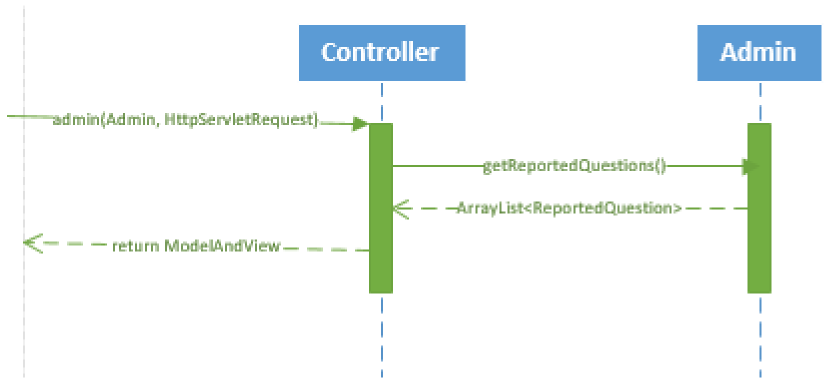
Sequence Diagram 25 3.2.38 Poll of the Day



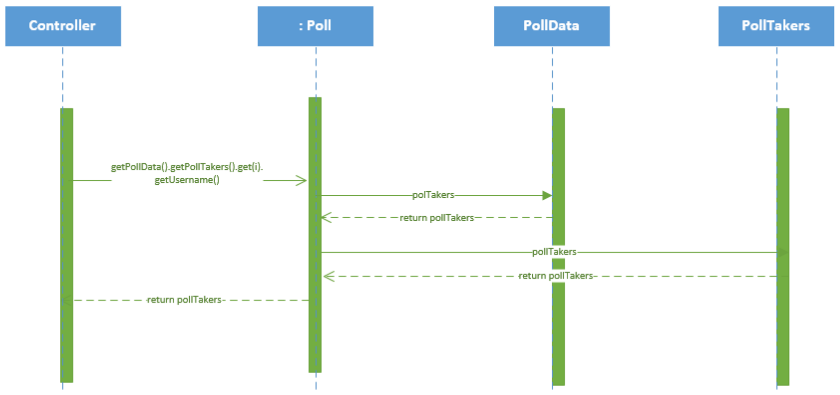
Sequence Diagram 26 3.2.39 Random Pull


Controller
: Group
DBQuery
addUserToGroup(String username, String Groupnum, HttpServletRequest)
addUsertoGroup(username, groupNum)
username
groupNum
return Void
return Void
return ModelAndView

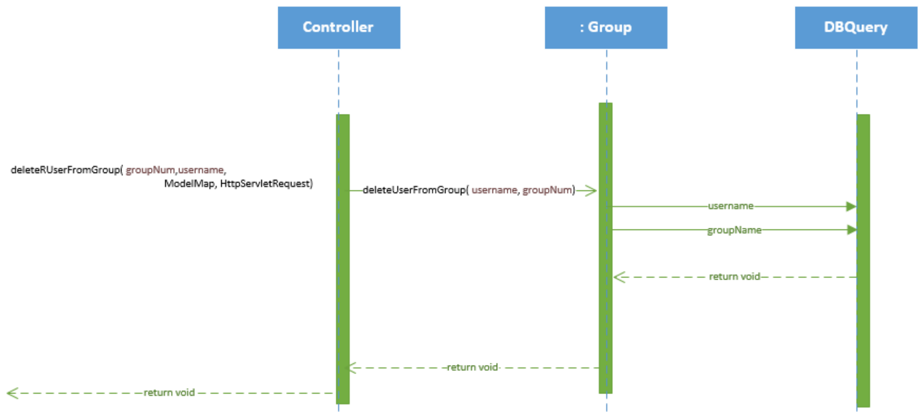
Sequence Diagram 27 3.2.13 Invite User to Private Poll



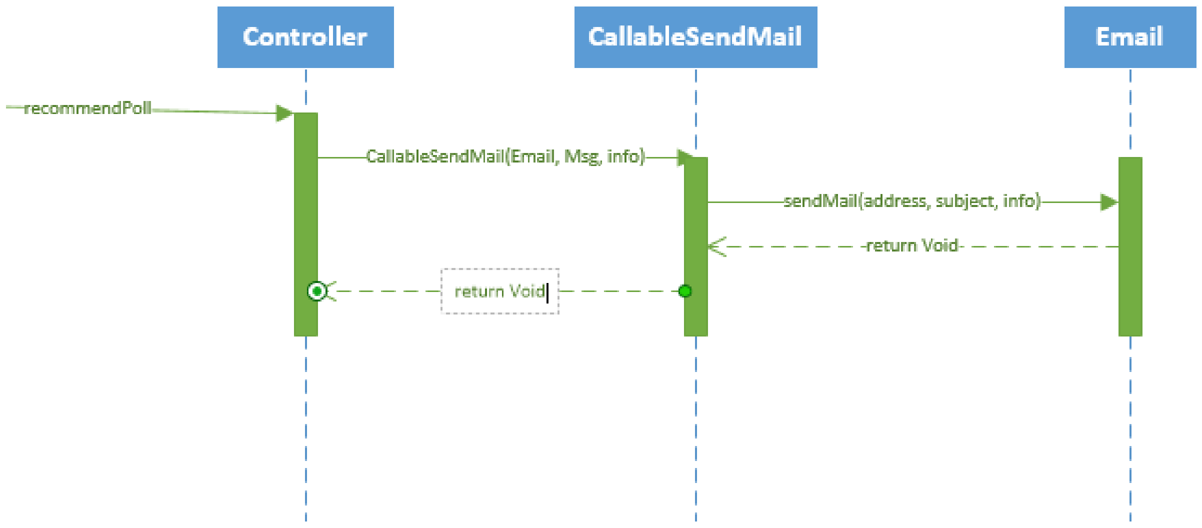
Sequence Diagram 28 3.2.36 View Reported Offensive Poll



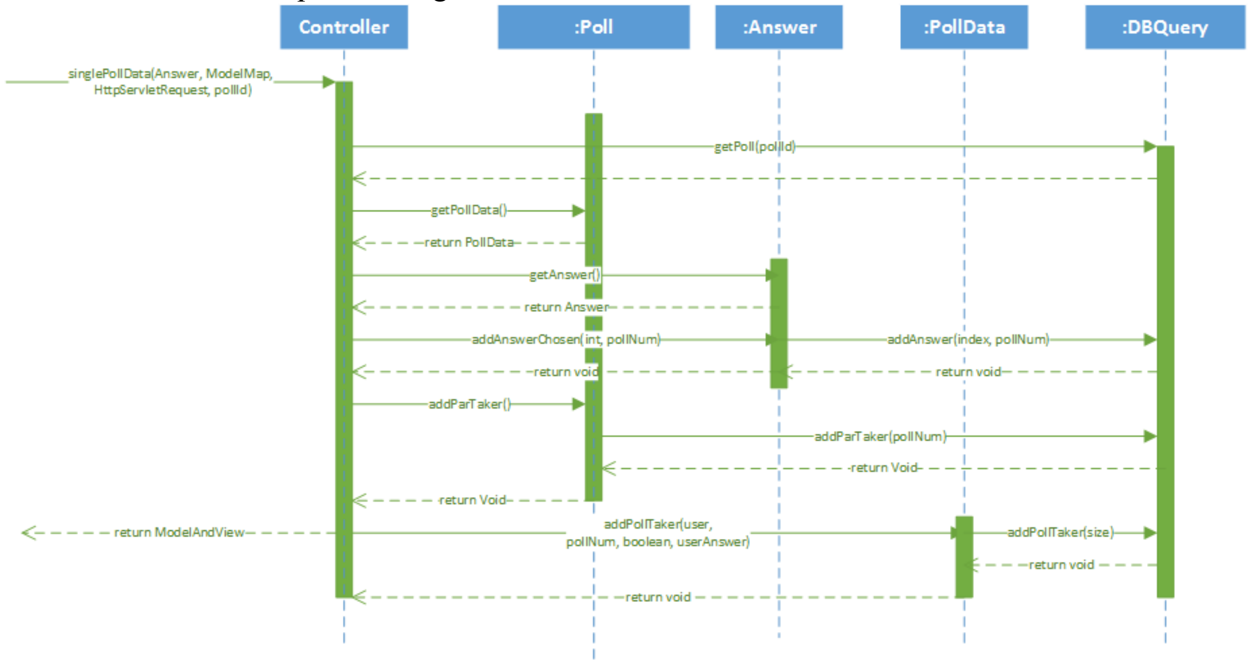
Sequence Diagram 29 3.2.14 Limit Users to One Vote Per Poll


Controller
: Group
DBQuery
deleteRUserFromGroup(groupNum,username,
ModelMap, HttpServletRequest)
deleteUserFromGroup(username, groupNum)
username
groupName
return void
return void
return void

Sequence Diagram 30 3.2.16 Delete Invited Users

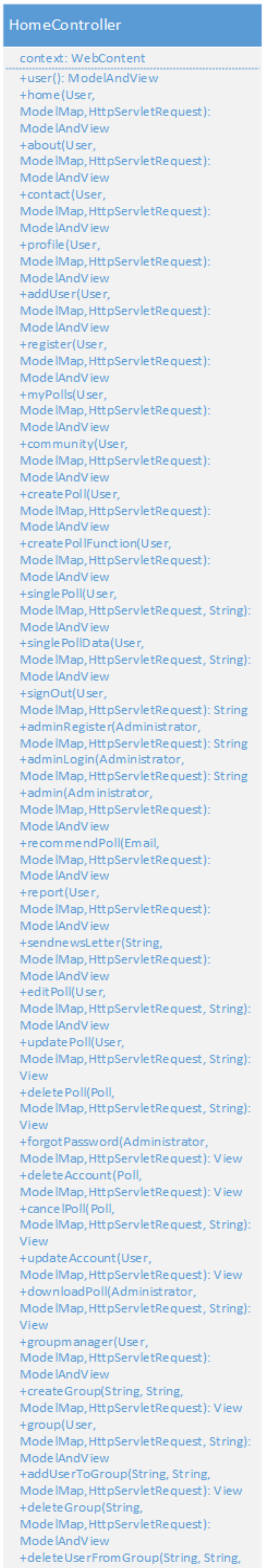

Controller
CallableSendMail
Email
recommendPoll
CallableSendMail(Email, Msg, info)
sendMail(address, subject, info)
return Void
return Void

Sequence Diagram 32 3.2.29 Recommend Polls to Others



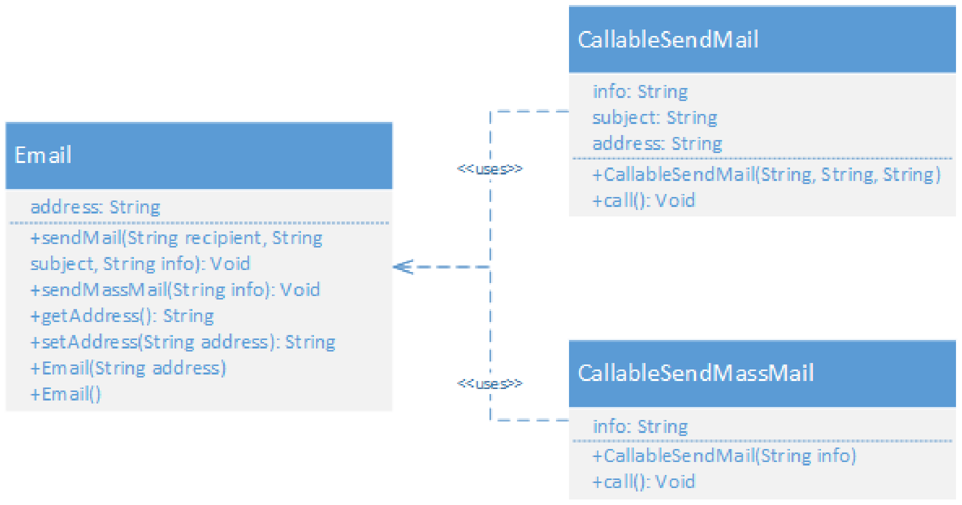
Sequence Diagram 33 3.2.4 Vote in Private Poll

4.4.4.6 Class Diagrams

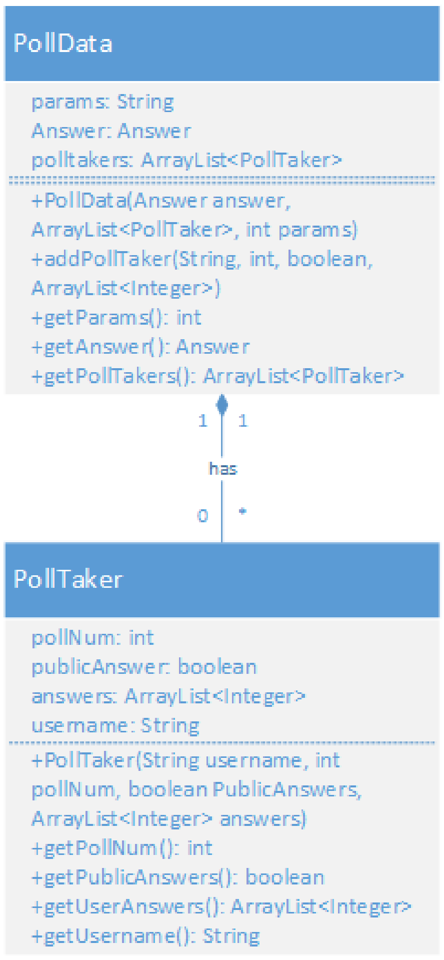


```
HomeController
context: WebContent
+user(): ModelAndView
+home(User,
ModelMap,HttpServletRequest):
ModelAndView
+about(User,
ModelMap,HttpServletRequest):
ModelAndView
+contact(User,
ModelMap,HttpServletRequest):
ModelAndView
+profile(User,
ModelMap,HttpServletRequest):
ModelAndView
+addUser(User,
ModelMap,HttpServletRequest):
ModelAndView
+register(User,
ModelMap,HttpServletRequest):
ModelAndView
+myPolls(User,
ModelMap,HttpServletRequest):
ModelAndView
+community(User,
ModelMap,HttpServletRequest):
ModelAndView
+createPoll(User,
ModelMap,HttpServletRequest):
ModelAndView
+createPollFunction(User,
ModelMap,HttpServletRequest):
ModelAndView
+singlePoll(User,
ModelMap,HttpServletRequest, String):
ModelAndView
+singlePollData(User,
ModelMap,HttpServletRequest, String):
ModelAndView
+signOut(User,
ModelMap,HttpServletRequest): String
+adminRegister(Administrator,
ModelMap,HttpServletRequest): String
+adminLogin(Administrator,
ModelMap,HttpServletRequest): String
+admin(Administrator,
ModelMap,HttpServletRequest):
ModelAndView
+recommendPoll(Email,
ModelMap,HttpServletRequest):
ModelAndView
+report(User,
ModelMap,HttpServletRequest):
ModelAndView
+sendnewsLetter(String,
ModelMap,HttpServletRequest):
ModelAndView
+editPoll(User,
ModelMap,HttpServletRequest, String):
ModelAndView
+updatePoll(User,
ModelMap,HttpServletRequest, String):
View
+deletePoll(Poll,
ModelMap,HttpServletRequest, String):
View
+forgotPassword(Administrator,
ModelMap,HttpServletRequest): View
+deleteAccount(Poll,
ModelMap,HttpServletRequest): View
+cancelPoll( Poll,
ModelMap,HttpServletRequest, String):
View
+updateAccount(User,
ModelMap,HttpServletRequest): View
+downloadPoll(Administrator,
ModelMap,HttpServletRequest, String):
View
+groupmanager(User,
ModelMap,HttpServletRequest):
ModelAndView
+createGroup(String, String,
ModelMap,HttpServletRequest): View
+group(User,
ModelMap,HttpServletRequest, String):
ModelAndView
+addUserToGroup(String, String,
ModelMap,HttpServletRequest): View
+deleteGroup(String,
ModelMap,HttpServletRequest):
ModelAndView
+deleteUserFromGroup(String, String,
```

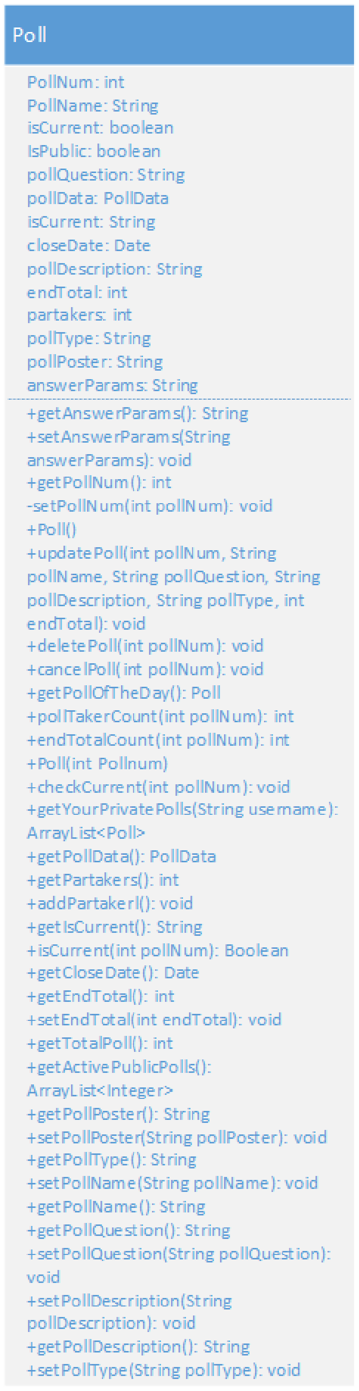
Class Diagram 1: This diagram is the central part of our application, but it's so large that it's displayed by itself. In all proceeding diagrams, we'll refer to it without all the methods for viewability.



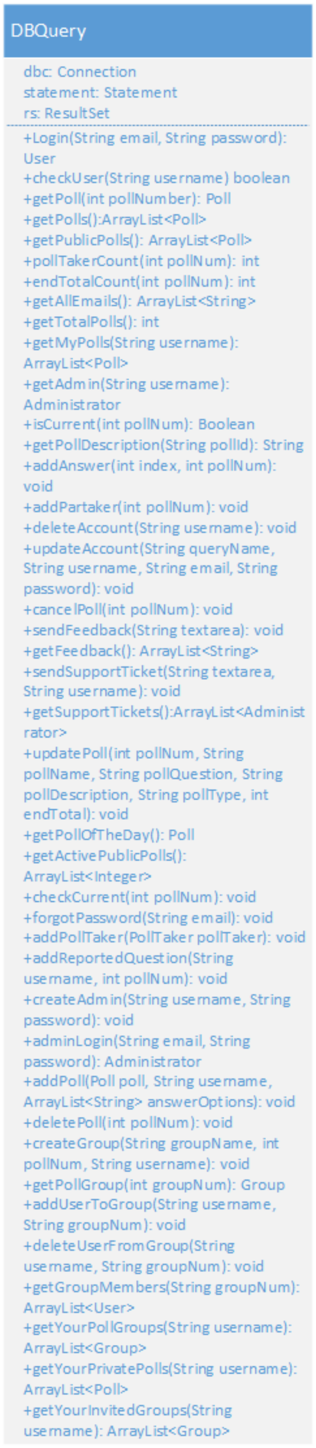
Class Diagram 2: This diagram shows how we send emails. The HomeController creates threads that will use the CallableSendMail and CallableSendMassMail classes. Both of those classes run a method from class Email.


PollData
params: String
Answer: Answer
polltakers: ArrayList<PollTaker>
+PollData(Answer answer, ArrayList<PollTaker>, int params)
+addPollTaker(String, int, boolean, ArrayList<Integer>)
+getParams(): int
+getAnswer(): Answer
+getPollTakers(): ArrayList<PollTaker>
1
1
has
0
*
PollTaker
pollNum: int
publicAnswer: boolean
answers: ArrayList<Integer>
username: String
+PollTaker(String username, int pollNum, boolean PublicAnswers, ArrayList<Integer> answers)
+getPollNum(): int
+getPublicAnswers(): boolean
+getUserAnswers(): ArrayList<Integer>
+getUsername(): String

Class Diagram 3: This diagram's emphasis is on PollTaker. PollTaker only exists if PollData exists. PollData can have zero to many PollTakers.

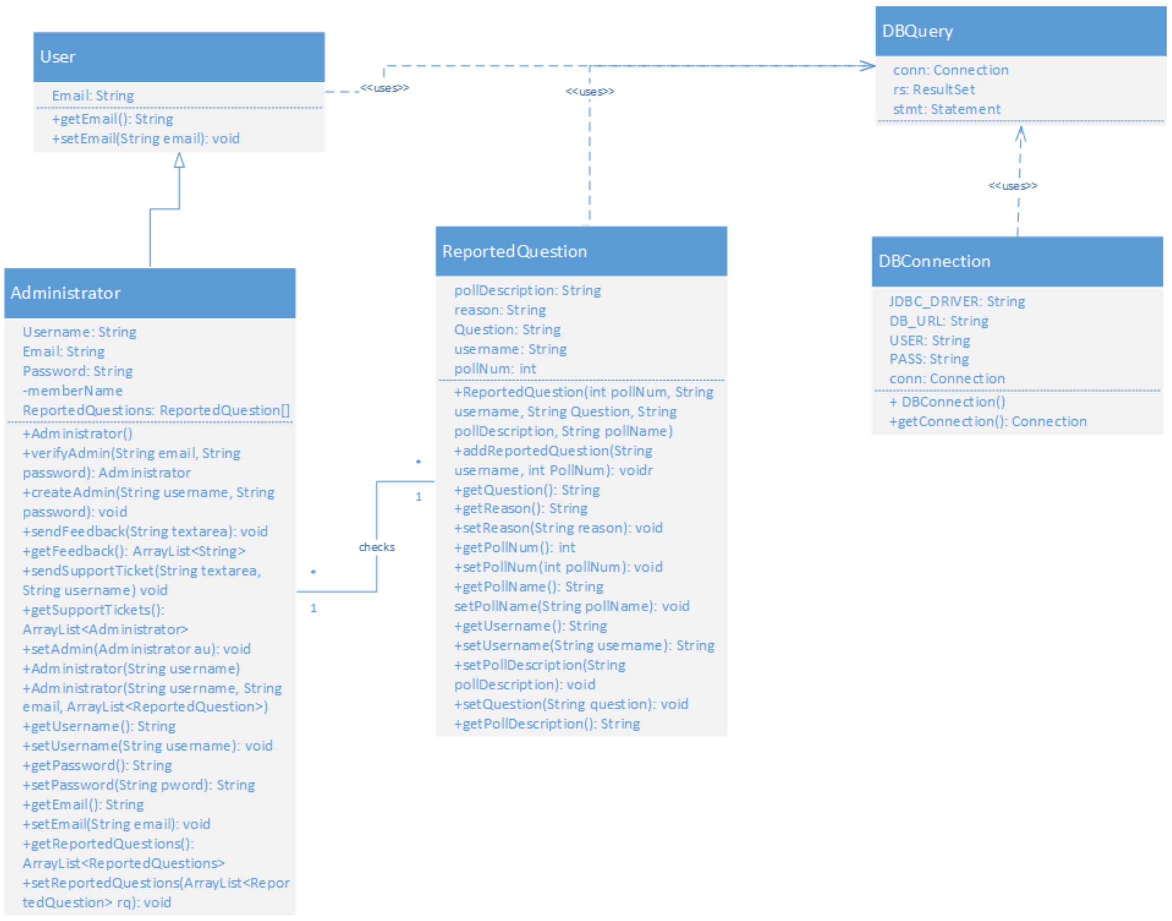

Poll
PollNum: int
PollName: String
isCurrent: boolean
IsPublic: boolean
pollQuestion: String
pollData: PollData
isCurrent: String
closeDate: Date
pollDescription: String
endTotal: int
partakers: int
pollType: String
pollPoster: String
answerParams: String
+getAnswerParams(): String
+setAnswerParams(String answerParams): void
+getPollNum(): int
-setPollNum(int pollNum): void
+Poll()
+updatePoll(int pollNum, String pollName, String pollQuestion, String pollDescription, String pollType, int endTotal): void
+deletePoll(int pollNum): void
+cancelPoll(int pollNum): void
+getPollOfTheDay(): Poll
+pollTakerCount(int pollNum): int
+endTotalCount(int pollNum): int
+Poll(int Pollnum)
+checkCurrent(int pollNum): void
+getYourPrivatePolls(String username): ArrayList<Poll>
+getPollData(): PollData
+getPartakers(): int
+addPartakerl(): void
+getIsCurrent(): String
+isCurrent(int pollNum): Boolean
+getCloseDate(): Date
+getEndTotal(): int
+setEndTotal(int endTotal): void
+getTotalPoll(): int
+getActivePublicPolls(): ArrayList<Integer>
+getPollPoster(): String
+setPollPoster(String pollPoster): void
+getPollType(): String
+setPollName(String pollName): void
+getPollName(): String
+getPollQuestion(): String
+setPollQuestion(String pollQuestion): void
+setPollDescription(String pollDescription): void
+getPollDescription(): String
+setPollType(String pollType): void

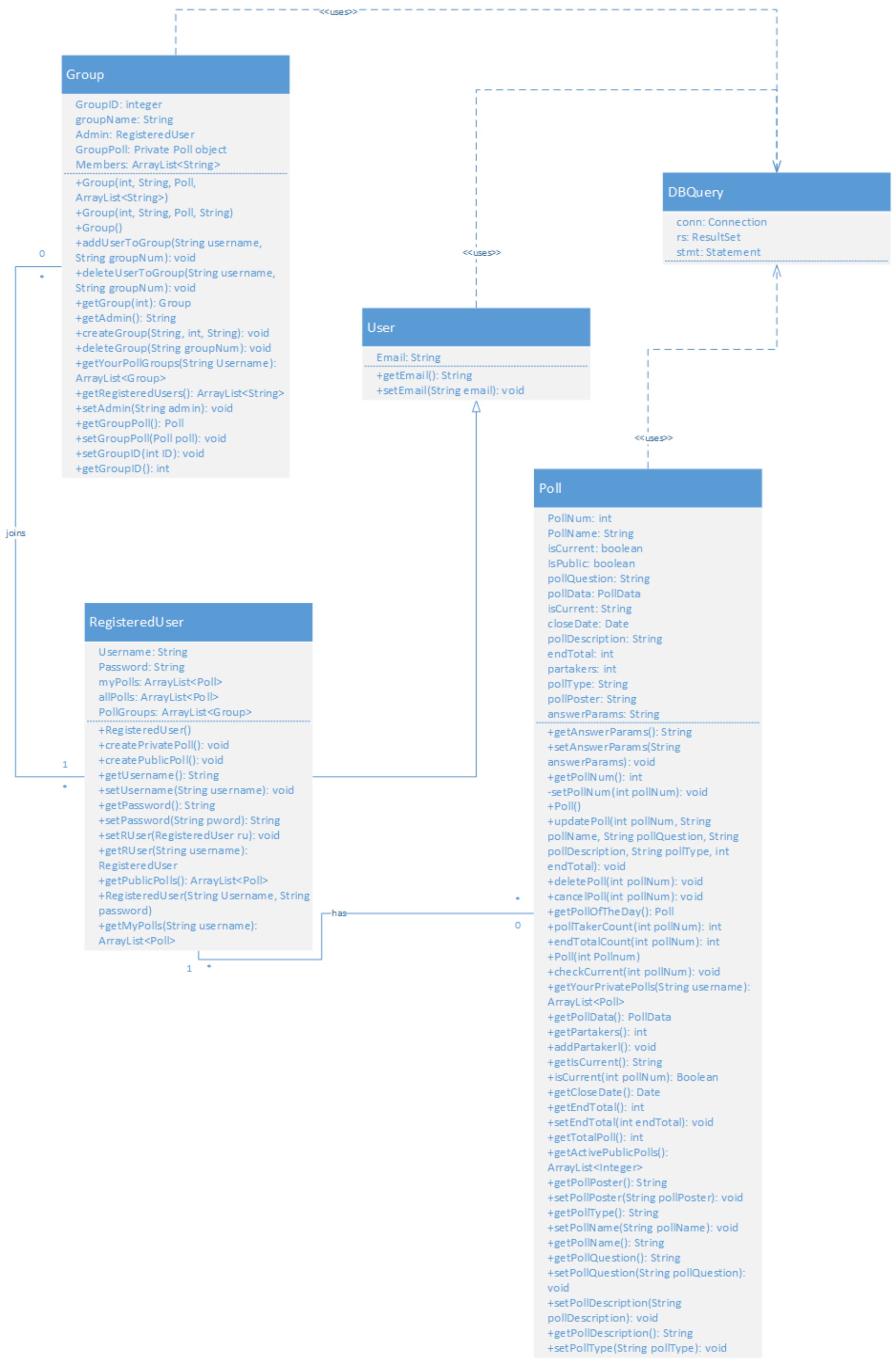
Class Diagram 4: This diagram displays the Poll class. Since it is so large, we are displaying it as a standalone diagram. All proceeding diagrams will refer to it without all the methods.



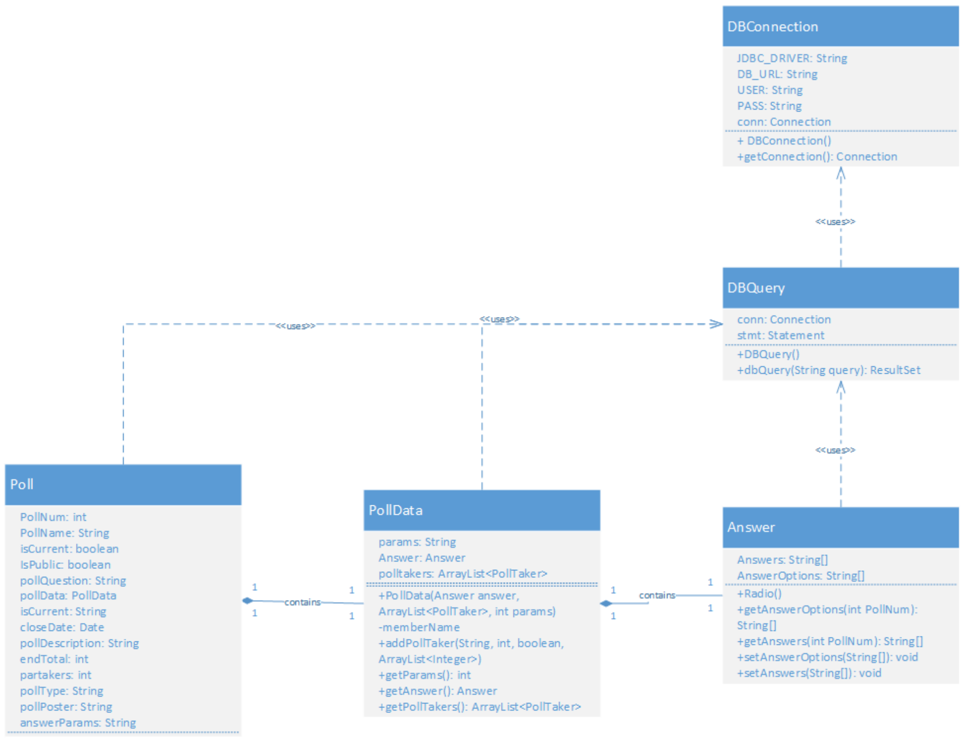
```
DBQuery
dbc: Connection
statement: Statement
rs: ResultSet
+Login(String email, String password): User
+checkUser(String username) boolean
+getPoll(int pollNumber): Poll
+getPolls():ArrayList<Poll>
+getPublicPolls(): ArrayList<Poll>
+pollTakerCount(int pollNum): int
+endTotalCount(int pollNum): int
+getAllEmails(): ArrayList<String>
+getTotalPolls(): int
+getMyPolls(String username): ArrayList<Poll>
+getAdmin(String username): Administrator
+isCurrent(int pollNum): Boolean
+getPollDescription(String pollId): String
+addAnswer(int index, int pollNum): void
+addPartaker(int pollNum): void
+deleteAccount(String username): void
+updateAccount(String queryName, String username, String email, String password): void
+cancelPoll(int pollNum): void
+sendFeedback(String textarea): void
+getFeedback(): ArrayList<String>
+sendSupportTicket(String textarea, String username): void
+getSupportTickets():ArrayList<Administrator>
+updatePoll(int pollNum, String pollName, String pollQuestion, String pollDescription, String pollType, int endTotal): void
+getPollOfTheDay(): Poll
+getActivePublicPolls(): ArrayList<Integer>
+checkCurrent(int pollNum): void
+forgotPassword(String email): void
+addPollTaker(PollTaker pollTaker): void
+addReportedQuestion(String username, int pollNum): void
+createAdmin(String username, String password): void
+adminLogin(String email, String password): Administrator
+addPoll(Poll poll, String username, ArrayList<String> answerOptions): void
+deletePoll(int pollNum): void
+createGroup(String groupName, int pollNum, String username): void
+getPollGroup(int groupNum): Group
+addUserToGroup(String username, String groupNum): void
+deleteUserFromGroup(String username, String groupNum): void
+getGroupMembers(String groupNum): ArrayList<User>
+getYourPollGroups(String username): ArrayList<Group>
+getYourPrivatePolls(String username): ArrayList<Poll>
+getYourInvitedGroups(String username): ArrayList<Group>
```

Class Diagram 5: This diagram Displays the DBQuery class. Since it is so big, it is a standalone diagram. All proceeding diagrams will refer to it without methods for viewability


User
Email: String
+getEmail(): String
+setEmail(String email): void
<<uses>>
<<uses>>
DBQuery
conn: Connection
rs: ResultSet
stmt: Statement
<<uses>>
DBConnection
JDBC_DRIVER: String
DB_URL: String
USER: String
PASS: String
conn: Connection
+ DBConnection()
+getConnection(): Connection
ReportedQuestion
pollDescription: String
reason: String
Question: String
username: String
pollNum: int
+ReportedQuestion(int pollNum, String username, String Question, String pollDescription, String pollName)
+addReportedQuestion(String username, int PollNum): voidr
+getQuestion(): String
+getReason(): String
+setReason(String reason): void
+getPollNum(): int
+setPollNum(int pollNum): void
+getPollName(): String
setPollName(String pollName): void
+getUsername(): String
+setUsername(String username): String
+setPollDescription(String pollDescription): void
+setQuestion(String question): void
+getPollDescription(): String
*
1
checks
*
1
Administrator
Username: String
Email: String
Password: String
-memberName
ReportedQuestions: ReportedQuestion[]
+Administrator()
+verifyAdmin(String email, String password): Administrator
+createAdmin(String username, String password): void
+sendFeedback(String textarea): void
+getFeedback(): ArrayList<String>
+sendSupportTicket(String textarea, String username) void
+getSupportTickets(): ArrayList<Administrator>
+setAdmin(Administrator au): void
+Administrator(String username)
+Administrator(String username, String email, ArrayList<ReportedQuestion>)
+getUsername(): String
+setUsername(String username): void
+getPassword(): String
+setPassword(String pword): String
+getEmail(): String
+setEmail(String email): void
+getReportedQuestions(): ArrayList<ReportedQuestions>
+setReportedQuestions(ArrayList<ReportedQuestion> rq): void
Class Diagram 6: This Diagram centers around the functionality of Administrator and Reported Question.


«uses»
Group
GroupID: integer
groupName: String
Admin: RegisteredUser
GroupPoll: Private Poll object
Members: ArrayList<String>
+Group(int, String, Poll, ArrayList<String>)
+Group(int, String, Poll, String)
+Group()
+addUserToGroup(String username, String groupNum): void
+deleteUserToGroup(String username, String groupNum): void
+getGroup(int): Group
+getAdmin(): String
+createGroup(String, int, String): void
+deleteGroup(String groupNum): void
+getYourPollGroups(String Username): ArrayList<Group>
+getRegisteredUsers(): ArrayList<String>
+setAdmin(String admin): void
+getGroupPoll(): Poll
+setGroupPoll(Poll poll): void
+setGroupID(int ID): void
+getGroupID(): int
DBQuery
conn: Connection
rs: ResultSet
stmt: Statement
«uses»
User
Email: String
+getEmail(): String
+setEmail(String email): void
«uses»
0
*
joins
1
*
RegisteredUser
Username: String
Password: String
myPolls: ArrayList<Poll>
allPolls: ArrayList<Poll>
PollGroups: ArrayList<Group>
+RegisteredUser()
+createPrivatePoll(): void
+createPublicPoll(): void
+getUsername(): String
+setUsername(String username): void
+getPassword(): String
+setPassword(String pword): String
+setRUser(RegisteredUser ru): void
+getRUser(String username): RegisteredUser
+getPublicPolls(): ArrayList<Poll>
+RegisteredUser(String Username, String password)
+getMyPolls(String username): ArrayList<Poll>
has
1
*
*
0
Poll
PollNum: int
PollName: String
isCurrent: boolean
IsPublic: boolean
pollQuestion: String
pollData: PollData
isCurrent: String
closeDate: Date
pollDescription: String
endTotal: int
partakers: int
pollType: String
pollPoster: String
answerParams: String
+getAnswerParams(): String
+setAnswerParams(String answerParams): void
+getPollNum(): int
-setPollNum(int pollNum): void
+Poll()
+updatePoll(int pollNum, String pollName, String pollQuestion, String pollDescription, String pollType, int endTotal): void
+deletePoll(int pollNum): void
+cancelPoll(int pollNum): void
+getPollOfTheDay(): Poll
+pollTakerCount(int pollNum): int
+endTotalCount(int pollNum): int
+Poll(int Pollnum)
+checkCurrent(int pollNum): void
+getYourPrivatePolls(String username): ArrayList<Poll>
+getPollData(): PollData
+getPartakers(): int
+addPartaker(): void
+getIsCurrent(): String
+isCurrent(int pollNum): Boolean
+getCloseDate(): Date
+getEndTotal(): int
+setEndTotal(int endTotal): void
+getTotalPoll(): int
+getActivePublicPolls(): ArrayList<Integer>
+getPollPoster(): String
+setPollPoster(String pollPoster): void
+getPollType(): String
+setPollName(String pollName): void
+getPollName(): String
+getPollQuestion(): String
+setPollQuestion(String pollQuestion): void
+setPollDescription(String pollDescription): void
+getPollDescription(): String
+setPollType(String pollType): void

Class Diagram 7: This diagram shows the core functionality of a registered user. A registered User has polls and groups and uses the DBQuery class.



Class Diagram 8: This diagram shows all dependent classes of Poll and how they all connect to DBQuery