Nick Miller & Alpha Ding

Project Proposal

Project Title
Ask Around

Summary
Some questions Google can't answer. Questions like, "Which shirt looks better on me?" or "Which prof should I take disco with?" For these questions, you need human input - input from people who relate to you in the surrounding area.

Enter Ask Around. This app will allow users to ask questions anonymously to other users in the local area. Question askers may choose to specify potential answers or leave options open. They may include pictures with their questions, and potentially audio and video as well. Question answerers will be able to make their opinions known through answering questions, making comments, and voting on comments, too. They will further be able to view the results of questions they have answered.

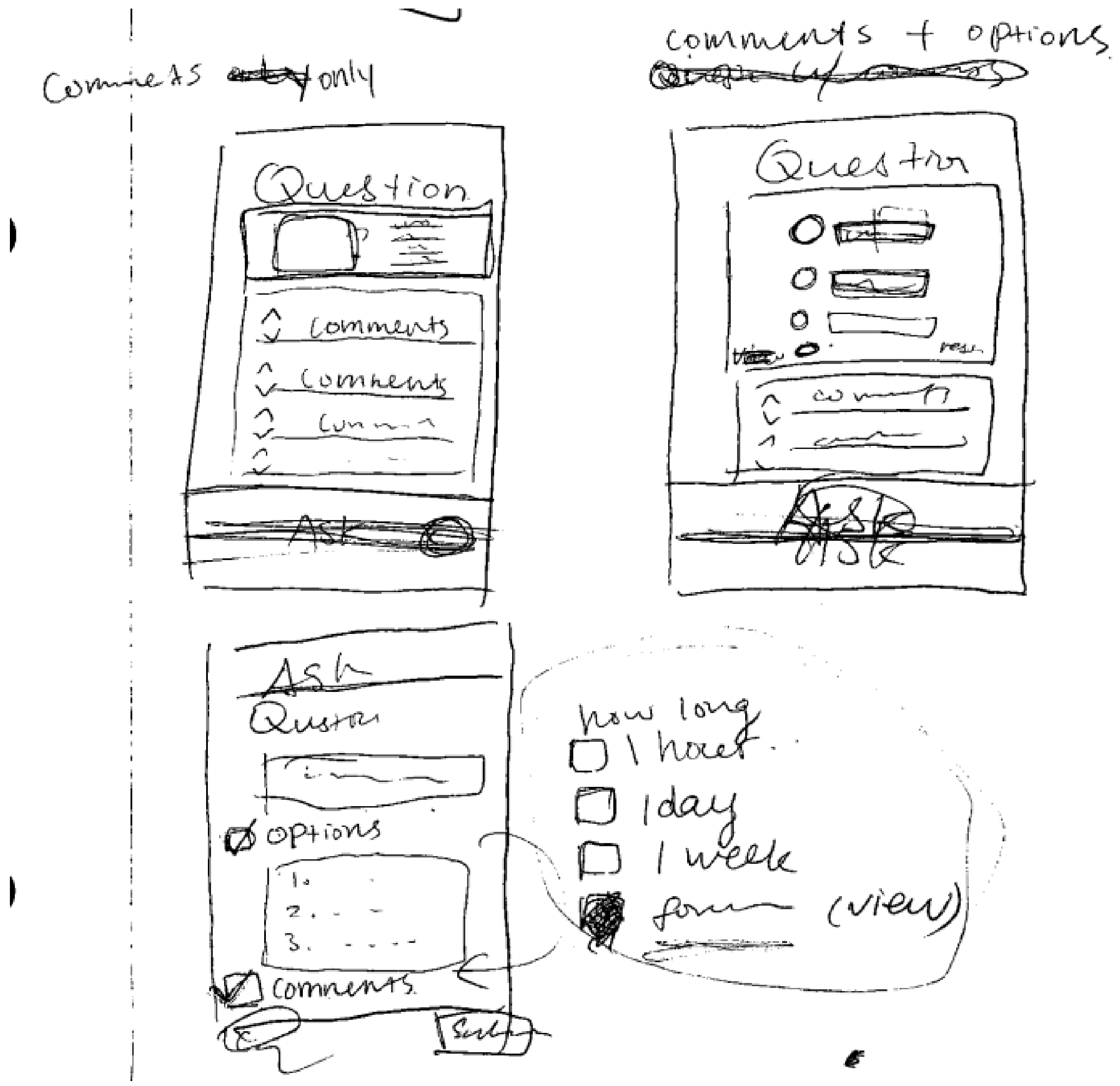
More example questions:
- What ice cream flavor should I get today?
  - Chocolate? Vanilla? Jelly bean?
- Where can I get some good Italian food?
  - Open to comments
- Who has the best reception on campus?
  - Verizon, AT&T, TMobile, Sprint
- Do I need a jacket today?
  - Yes, no, maybe
- Is the ARA worth it today?
  - Yes, no, never
- What bands are you listening to?
  - Open to comments

Functional Requirements
- Users will be able to ask questions, specifying parameters
- Users will be able to view the results of their active and inactive questions
- Users will be able to answer other user's questions based on the question parameters
- Users will be able to comment on questions and vote on these comments
- Questions will only be shown to other users nearby

- Users must answer the current question before they may view the next question
- Users must answer a question before they may see the results of that question
- Users will be able to include pictures with their questions. Support for audio and video may be added as well
- Users will earn points by asking questions, answering questions, commenting, and voting on comments which may be redeemed for rewards within the app

Layout Mocks

- Top Left: Answering a question, just comments
- Top Right: Answering a question with options and comments
- Bottom: Creating a question



Wireframe

https://www.fluidui.com/editor/live/preview/p_Ig6k7TtQu69UMdkJsZBEU1MtGXFkLHhv.1420591422620

To navigate the mock demo, click on the following buttons:

1. Login Page: Click Register
2. Registration Page: Click Register
3. Home Screen (Ice Cream Question): Click Submit Answer
4. Results Screen A: Click Next Question!
5. Home Screen (Reception Question): Click Skip Question
6. Home Screen (Bands Question): Click the green plus in the top right corner
7. Ask a Question Screen: Click Submit Question
8. Home Screen (Bands Question) again: Click Results
9. Results Screen B: You’re done!

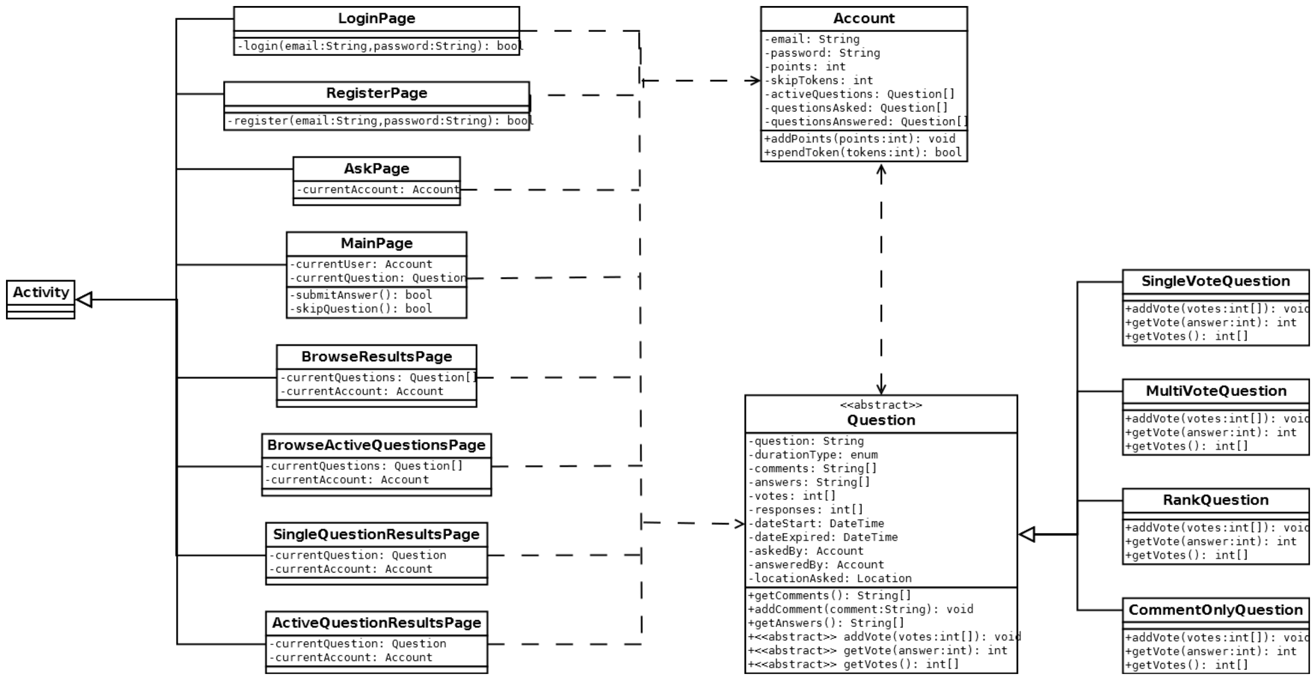
Requirements and Design

User Stories

- Registering an account and logging in
  - When the user opens the app for the first time, a log in screen appears. They can register an account by supplying their email address and a password. They will then gain access to the features of the app.
- Asking a question
  - From the main screen of the app, the user can press on the ask a question button. This will bring up the ask a question page. Here the user can ask a question along with specifying the type of answers, answer choices (if applicable), and duration of the question. When the user presses the submit button, the question goes live.
- Answering a question with comments
  - The main screen of the app displays a question that the user can answer. Depending on the type of question, the user will be allowed to select one answer, select many answers, or rank answers to the question by pressing the answers accordingly. Some questions have no answer choices. All questions will allow the user to leave a comment of a reasonable length. When the user presses the submit button, their answer will be recorded and the user will be redirected to the results page for that question. This page will display the number of each answer recieved as well as the comments submitted.
- Browsing answered questions by themselves and others
  - From the main screen of the app, the user can press the results button. This will bring up the browse results page. Here the user can view questions that have been previously answered along with the age of the question and the number of responses. They can alternate between sorting questions by time submitted or number of responses by pressing the sort button. They can alternate between filtering questions by who asked it (the user or someone else) by pressing the filter button. When the user presses a question, it will take them to the results screen for that question.
- Viewing Active Questions
  - From the main screen of the app, the user can press the your active question button. This will bring up the browse active questions page. Here the questions that user has asked which have not yet expired will be listed, along with their age, number of responses, and time left before expiring. The user can press any of the questions to view the results page for that question, with a note that these results are not final.
- Earning & Spending Points
  - When the user submits questions and answers questions, they recieve a certain amount of points. The user’s current total of points is visible in a bar at the top of most pages of the app. After earning a sufficient number of points,

the user is rewarded with a skip token. The skip token may be spent to skip answering a question by pressing the skip button on the main page. Skipping a question allows the user to directly view the results page of the question and prevents the user from answering that question. The next question is then displayed on the main page.

UML



Activity
LoginPage
-login(email:String,password:String): bool
RegisterPage
-register(email:String,password:String): bool
AskPage
-currentAccount: Account
MainPage
-currentUser: Account
-currentQuestion: Question
-submitAnswer(): bool
-skipQuestion(): bool
BrowseResultsPage
-currentQuestions: Question[]
-currentAccount: Account
BrowseActiveQuestionsPage
-currentQuestions: Question[]
-currentAccount: Account
SingleQuestionResultsPage
-currentQuestion: Question
-currentAccount: Account
ActiveQuestionResultsPage
-currentQuestion: Question
-currentAccount: Account
Account
-email: String
-password: String
-points: int
-skipTokens: int
-activeQuestions: Question[]
-questionsAsked: Question[]
-questionsAnswered: Question[]
+addPoints(points:int): void
+spendToken(tokens:int): bool
<<abstract>>
Question
-question: String
-durationType: enum
-comments: String[]
-answers: String[]
-votes: int[]
-responses: int[]
-dateStart: DateTime
-dateExpired: DateTime
-askedBy: Account
-answeredBy: Account
-locationAsked: Location
+getComments(): String[]
+addComment(comment:String): void
+getAnswers(): String[]
+<<abstract>> addVote(votes:int[]): void
+<<abstract>> getVote(answer:int): int
+<<abstract>> getVotes(): int[]
SingleVoteQuestion
+addVote(votes:int[]): void
+getVote(answer:int): int
+getVotes(): int[]
MultiVoteQuestion
+addVote(votes:int[]): void
+getVote(answer:int): int
+getVotes(): int[]
RankQuestion
+addVote(votes:int[]): void
+getVote(answer:int): int
+getVotes(): int[]
CommentOnlyQuestion
+addVote(votes:int[]): void
+getVote(answer:int): int
+getVotes(): int[]

Link to larger version:
https://drive.google.com/file/d/0B4xz3Ieu9v5PaUowOGI5Mzh1bnc/view?usp=sharing

Sprint 1 Goals
We will implement the majority of the GUI. We will create the activity classes along with their associated XML layouts. We will leave the logic implementation for future sprints.

Sprint 1 Progress
We implemented most of the activity XML. Our layouts, while not exact matches, closesly approximate our mock-ups. They include all elements needed for functionality (i.e. buttons, input text, etc.) We were unsure how to properly implement tabs, so we have left them for next sprint. We created stubs for the activity classes.

Sprint 2 Goals
We will implement simple versions of our features. This will include:

- Asking one type of question
- Answering one type of question
- Viewing the results of one type of question
- Loggin In
- Registration
- GUI tabs

This will involve implementation in the activity classes, backend, and models for questions and votes.