# ECE1779 Assignment III
# AskCrowd - Final Report

Hao Jin (1000303405)
Louis Chen (1000303502)

# Table of Contents

**1.0 What is AskCrowd? 2**

1.1 Background and Motivation 2

1.2 General Architecture 2

**2.0 How to Use AskCrowd? 3**

**3.0 Advanced Features 4**

3.1 Automatic Poll Suggestion 4

3.2 Search and Filter 4

3.3 Email Sharing 4

3.4 Google Search 4

3.5 Comments 4

**4.0 Software Architecture 5**

**5.0 Database Architecture 8**

**6.0 Storage Architecture 9**

**7.0 APIs 9**

**8.0 Future Work 10**

**9.0 Individual Contribution 11**

**10.0 Revision Control 12**

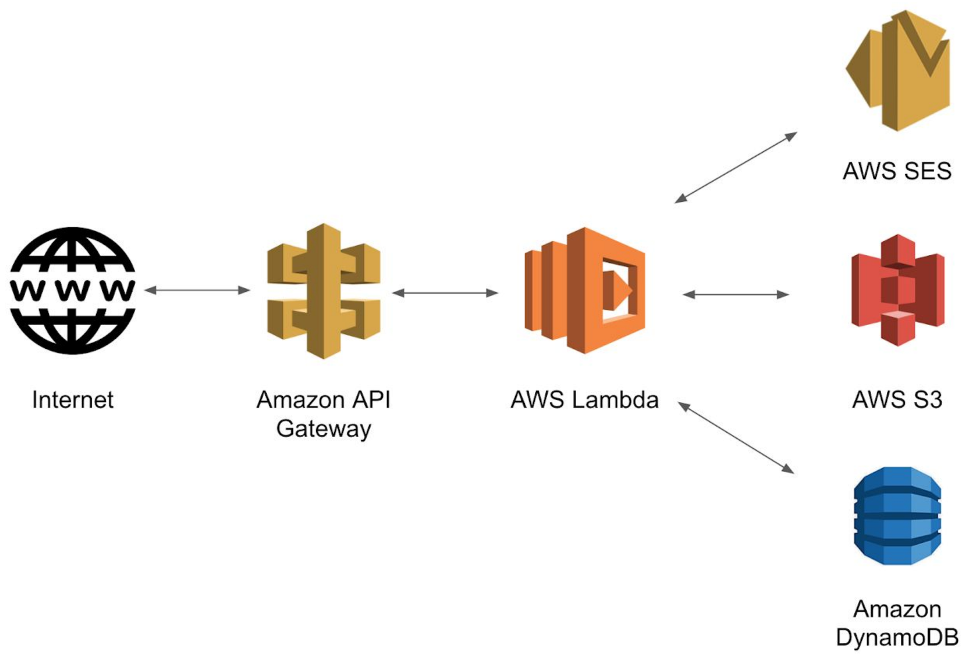
# 1.0 What is AskCrowd?

## 1.1 Background and Motivation

AskCrowd is an online polling platform which allows everyone to create a poll for the crowd to vote. In a democratic society, polling is an important factor for political parties to consider, especially when it is near election period. Traditional polling services are almost always conducted by private sector. This may lead to biased results due to political or financial interest of the polling company. Besides, traditional polling services are usually done over phones, and therefore are time-consuming and require human resources. There are online polling services such as Doodle which conduct polling services over the Internet. Nevertheless, these services are private to a designated group of people; they are not public to the crowd. Moreover, news websites and social networking platforms such as Facebook and YouTube have incorporated polling as part of their services, yet these websites are not mainly dedicated for poll sharing. For example, the concept of Facebook's feed is to share one's daily moment with his/her friends, instead of creating a poll. Therefore, the team sees a potential to create such online polling-oriented platform, public for the crowd to create and vote a poll.

## 1.2 General Architecture

AskCrowd is a serverless web application developed using Flask in Python, deployed on the Internet using Amazon Web Services (AWS) Lambda and API Gateway. All data is stored on AWS DynamoDB, and all files are stored on AWS Simple Storage Service (S3). AskCrowd also has an email functionality hosted and served by Amazon Simple Email Service (Amazon SES). The following graph shows the relationship between these AWS services.



All functions needed for the web application are uploaded on Lambda, redirected by API Gateway. Lambda then communicates with S3, DynamoDB and SES when necessary.

Basic functionalities of AskCrowd include user registration, creating, voting, and commenting a poll. However, more user-friendly features are added to enhance user experience, such as automatic poll categorization, automatic poll suggestion, and filter/search bar. Details for each feature will be explained in further sections.

## 2.0 How to Use AskCrowd?

Since AskCrowd is deployed using AWS Lambda and API Gateway, Amazon provides a URL link which allows one to access the web application, as follows:

https://tasitavbrf.execute-api.us-east-1.amazonaws.com/dev/

Go ahead and open the link. Once open, press “Register” on the navigation menu to register for a new account. If registered, press “Login” and enter your credentials to login.

Now the website should display a list of polls, in two categories: polls that your have not voted, and polls that you have voted. Now, there are several operations you could take: create a poll, view a poll, vote a poll, and analyze a poll. These operations will be explained chronologically.

**Create a poll**: Press “Ask” on the navigation menu. You should be redirected to the “AskCrowd - Ask” page. You can now type in the question, description (optional), and answer options. By default, number of options is set to two. However, you may click on the “plus” button to add a new option, and “minus” button to delete the last option. Finally, check “Allow Comments” to enable comments, or uncheck to disable. Click “Ask” to create the poll. The page should now redirect you to the details page of the poll you just created (AskCrowd - Poll).

**View a poll**: Press “Home” on the navigation menu. You should now see a list of polls in a table, with five columns: poll, author, date, category, and participation. Column “poll” shows the question of the poll. Column “author” shows the user who created the poll. Column “date” shows date of poll creation. Column “category” shows the auto predicted category of the poll. Column “participation” shows the participation rate in percentage. Click any question you wish you view. Once clicked, you should be redirected to the details page of the poll.

**Vote a poll**: On the poll details page (AskCrowd - Poll), there is a table with three columns: voted, answers, and votes. Column “voted” shows the option you’ve previously voted, if applicable. Column “answers” shows the options for you to vote. Click on one of the options to vote. Once voted, the option you chose will show an “X” under column “voted”. If applicable, you may also be suggested with other polls that share similar content to the poll you just voted. You may also leave comments in the Comments section.

**Analyze a poll**: To analyze statistics of votes in a poll, click on any element under column "Votes". This will redirect you to page "AskCrowd - Analysis", which shows histograms of the votes in several attributes. Current attributes are date of birth, sex, and occupation.

AskCrowd contains several advanced features, which will be described in the next section.

## 3.0 Advanced Features

### 3.1 Automatic Poll Suggestion

AskCrowd has a user-friendly feature: automatic poll suggestion. When a user votes on a certain poll, the website will suggest polls which the user has not voted, but share similar content with the poll the user just voted. Category of a poll is automatically predicted by a machine learning classifier during creation of the poll. During initialization, the web application trains a classifier using multinomial Naive Bayes. The team references online existing source code to do this. However, the prediction does not have high enough accuracy due to insufficient training on classifier. This is done on purpose to speed things up for the sake of assignment demo. Therefore, a more accurate model will be used for further development in the future.

### 3.2 Search and Filter

A user is able to filter polls by category by selecting a category on the main page. He/She can also search a poll by typing keywords in the search bar.

### 3.3 Email Sharing

AskCrowd enables users to share polls through sending emails. On the poll detail page, a user can click on the button with email icon beside the poll question to send out the emails. Email service is hosted and served by AWS SES. Previously the team used Flask-email module to handle the sending email request. However, using the flask-email makes the web app less secure. Thus, the team sought another way of sending emails and found AWS SES would be the solution. One thing to mention is that as the team's aws account is in free trial limit, thus AWS SES only allows registered emails to send and receive emails.

### 3.4 Google Search

AskCrowd provides google search link for each poll for users to research more about the topic. One the poll detail page, a user can simply click on the google icon beside the question and a new tab showing Google search results will be opened in the browser as triggered by javascript functions.

### 3.5 Comments

AskCrowd provides comment section for each poll. A user may choose to able or disable commenting during creating a poll. Commenting enables users to share their ideas and communicate inside the website.

## 4.0 Software Architecture

The following is the architecture of AskCrowd in directory tree view.

- ❏ app/
    - ❏ templates/
        - ❏ account.html
        - ❏ analysis.html
        - ❏ ask.html
        - ❏ login.html
        - ❏ poll.html
        - ❏ profile.html
        - ❏ register.html
    - ❏ __init__.py
    - ❏ analysis.py
    - ❏ ask.py
    - ❏ classify.py
    - ❏ db.py
    - ❏ login.py
    - ❏ macro.py
    - ❏ main.py
    - ❏ poll.py
    - ❏ profile.py
    - ❏ register.py
- ❏ run.py

**Templates** folder contains seven HTML templates, which are account.html, analysis.html, ask.html, login.html, poll.html, profile.html, and register.html. Following describes what users are able to do in these pages.

- account.html
    - This template renders a page for users to view their account information. The page is only accessible if user is logged in. There are two sections in this page: *Account Information* and *Polls*. *Account Information* shows user's information such as username, sex, date of birth, occupation, profile image. It also allows user to change his/her password. *Polls* shows the polls created by the user as well as ones the user has voted. It also allows user to delete his/her poll.
- analysis.html
    - This template renders a page for users to view vote analysis. The page is only accessible if user is logged in. It shows the number of votes associated with a certain answer option in three attributes: sex, date of birth, and occupation. Main purpose of this page is for the crowd to analyze the distribution of votes across different attributes.
- ask.html

    - This template renders a page for users to create a new poll. The page is only accessible if user is logged in. It has the following sections: question, description, a list of options, and a checkbox to enable/disable commenting. User may create up to six answer options for the poll, with a minimum of two options.
- login.html
    - This template renders a page for users to log in with their credentials if they have registered before. It has two inputs. One is the username. Another one is the password. If the user is not registered, there is a button linked to the registration page. If the user inputs wrong username or password, an error message will be displayed colored in red.
- poll.html
    - This template renders a page which shows the details of a poll. The page is only accessible if user is logged in. It contains two sections: *question* and *comments*. The *question* section contains the question, description, author, date, category of the poll. It also shows a table of answer options with three columns. Column "Voted?" shows the answer option that the user has previously voted, if voted. Column "Answers (click to vote)" shows a list of answer options for user to click to vote. Column "Votes (click for analysis)" shows the number of current votes per answer option, and the user may click on it to view vote analysis. If the user has voted for this poll, the page may suggest a list of polls that share similar content with the current poll, if any. On the top-right corner shows two icons: Google and Email. User may press the Google icon to be redirected to the Google search engine with the question searched. User may also press the Email icon to share this poll via email. The *comments* section shows a list of comments, if the author of this poll has enabled commenting. Each comment contains commenter's username, comment, profile image, and timestamp. At the very bottom, user may leave his/her comment.
- profile.html
    - This template renders a page for users to view all of the polls. The page is only accessible if user is logged in. The polls are categorized into two groups: polls which user has not voted, and polls which user has voted. For each poll, it shows the question, author, date, category, and participation rate. At the top, user may click on the categories to filter the polls. User may also use the search bar to filter using keywords.
- register.html
    - This template renders a page for users to register for a new account. It has three inputs which are username, password, confirm password. If the user inputs a username that has been registered or mistypes the password, an error message will be displayed. There is also a button below these inputs to redirect the URL to the login page.

**Python files** that contribute to the main functionality of AskCrowd are listed below.

- __init__.py
    - Initializes AskCrowd.
    - Imports python files including main.py, login.py, register.py, profile.py, ask.py, poll.py, and analysis.py.
- analysis.py

        - Contains an HTTP GET handler function to perform voting analysis on a specific poll option.
            - Uses boto3 to retrieve poll information from AWS DynamoDB.
            - Calculates number of users who have voted on given option in the poll.
            - Categorizes into attributes (date of birth, sex, occupation).
            - Plots the results on histograms.
- ask.py
    - Contains an HTTP GET handler function to render the ask.html page.
    - Contains an HTTP POST handler function for poll creation.
        - Uses function in classify.py to categorizes question.
        - Writes poll into AWS DynamoDB.
        - Updates user info in AWS DynamoDB.
        - Redirects to poll detail page.
- classify.py
    - Downloads stopwords from NLTK.
    - Uses boto3 to download category training datasets from AWS S3.
    - Pre-processes datasets by tokenizing and vectorizing them.
    - Trains the categorization model using multinomial Naive Bayes classifier.
    - Contains a function that takes a question as input and returns predicted category for that question.
- db.py
    - Contains a function to retrieve AWS DynamoDB handler.
    - Contains a function to write an element onto AWS DynamoDB table.
    - Contains a function to read an element from AWS DynamoDB table.
        - Returns the element in JSON if found.
        - Returns Python NoneType if not found.
    - Contains a function to retrieve all elements from AWS DynamoDB table.
        - Returns an array of elements if found.
        - Returns Python NoneType if not found.
    - Contains a function to delete an element from AWS DynamoDB table.
    - Contains a function to retrieve AWS S3 handler.
    - Contains a function to upload a file onto AWS S3.
        - Able to take file path or actual file as input.
    - Contains a function to download a file from AWS S3.
        - Returns a URL to the file.
    - Contains a function to generate random salt for password during user registration.
- login.py
    - Contains an HTTP POST handler function to handle user login.
        - Validates username and password.
        - Check if username exists. If not, return error message.
        - Check if password matches. If not, return error message. Otherwise logs user in and redirects to the profile.html page.
- macro.py

    - Stores the secret key for Flask session.
    - Stores the name of S3 bucket.
    - Stores names of the tables on AWS DynamoDB.
    - Stores username/password/salt permitted characters as strings.
- main.py
    - Handles HTTP root/index/main URL GET request.
        - If user session exists (i.e. user was logged in), redirects user to the profile.html page.
        - Otherwise, redirects user to the login.html page.
    - Contains a helper function for the search engine.
- poll.py
    - Contains an HTTP GET handler for rendering the poll.html page.
        - Uses boto3 to retrieve poll content.
        - Calculates number of votes per option.
        - Retrieves suggested polls if current user has voted on this poll.
    - Contains an HTTP GET handler for sharing the poll via email.
        - Uses AWS Simple Email Service (SES) to send the email.
    - Contains an HTTP GET handler for voting the poll.
    - Contains an HTTP GET handler for deleting the poll.
    - Contains an HTTP GET handler for commenting the poll.
    - Contains a function to retrieve all comments in a poll.
- profile.py
    - Contains an HTTP GET handler for rendering the account.html page.
    - Contains an HTTP GET handler for logging out the user.
- register.py
    - Contains a function which retrieves the extension of an image file name.
    - Contains an HTTP GET handler for rendering the register.html form.
    - Contains an HTTP POST handler for registering a new account.

## 5.0 Database Architecture

AskCrowd stores all of its users, polls, comments, etc on AWS DynamoDB, which is a NoSQL type database. There are three tables on the AWS DynamoDB database: *askcrowd-users*, *askcrowd-polls*, *askcrowd-comments*, and *askcrowd-info*. Detailed architecture for each table is illustrated as follows.

Table **askcrowd-users**: Every element stores one user. It contains username, hashed password, salt, attributes (date of birth, sex, occupation), profile image URL, polls created, and polls voted.

Table **askcrowd-polls**: Every element stores one poll. It contains the poll id (named by timestamp), question, author, description, options, votes (usernames of users who vote), category, and a flag indicating whether commenting is allowed.

Table **askcrowd-comments**: Every element stores all comments for one poll. It contains the poll id, and a list of comments. For every element in the comment list, it contains commenter's username, commenter's profile image URL, comment, and timestamp. Reason to split this apart from **askcrowd-polls** is that elements in **askcrowd-polls** are accessed frequently, and therefore this reduces the overhead when read polls.

Table **askcrowd-info**: This table stores credentials for the web application, such as Flask session secret key, Lambda URL, and email used to share poll.

Elements in tables **askcrowd-users**, **askcrowd-polls**, and **askcrowd-comments** are added/modified on the fly, and whereas elements in table **askcrowd-info** are preconstructed. Helper functions for accessing DynamoDB are in db.py.

## 6.0 Storage Architecture

AskCrowd stores all files on AWS S3. The following shows the directory tree view of the S3 architecture.

- ❏ classify/
    - ❏ categories.txt
    - ❏ titles.txt
- ❏ css/
    - ❏ login.css
    - ❏ profile.css
- ❏ profile_image/
    - ❏ <username1>.png
    - ❏ <username2>.png
    - ❏ ...

Folder **classify** stores the training files for automatic poll suggestion. Folder **css** stores the CSS files for HTML files. Folder **profile_image** stores profile images for all registered users. In addition, documents in folders **classify** and **css** are preloaded, and whereas documents in folder **profile_image** are uploaded during user registration. Helper functions for accessing S3 are in db.py.

## 7.0 APIs

Currently, there is no public APIs from AskCrowd available, since the team has been focusing on implementing core functionalities of the web application. However, the team has come up with a list of APIs that may be useful to construct for further development.

**Creating a poll** </api/ask>: This is an HTTP POST API, which allows user to create a poll. For sure, user credentials need to be conveyed along with poll details in the HTTP form.

**View a poll** </api/view>: This is an HTTP GET API, which allows user to retrieve details of a poll. For sure, user credentials need to be conveyed along with poll ID in the URL. If poll is found, the details should be returned in JSON format.

**Vote a poll** </api/vote>: This is an HTTP POST API, which allows user to vote on a poll. For sure, user credentials need to be conveyed along with poll ID in the HTTP form. An error message with proper HTTP status code should be returned if the user has previously voted on the poll.

**Analyze a poll** </api/analyze>: This is an HTTP GET API, which allows user to retrieve vote distribution across user attributes on a certain poll. For sure, user credentials need to be conveyed along with poll ID in the URL. If valid, the response should be returned in proper JSON format.

All of the above are only suggested URLs. Detailed URL format will be revisited when the team is ready to construct APIs.

## 8.0 Future Work

All current features in AskCrowd are developed to form a basic framework. However, the team outlines a list of tasks to be done if this project were to be continued, and potentially be commercialized. They are listed as follows.

1. **Domain Name**: If AskCrowd were to be commercialized, a domain name is needed to bind with the AWS API Gateway URL.
2. **More User Attributes**: Currently, there are only three user attributes available for vote analysis. In the future, more attributes such as fields of interest, political interest, etc should be incorporated.
3. **Real-Name System**: Currently, users only need to provide a unique username in order to register an account. However, to assure the reliability of vote results, AskCrowd will enforce real-name verification to all users.
4. **Advertisement**: If AskCrowd were to be commercialized, the team needs to properly advertise it, since the most important thing is to have a huge pool of registered users before any useful analysis can take place.
5. **AskCrowd App**: Since lots of websites now offer an App version, AskCrowd should also do so to enhance user experience.
6. **UI Improvement**: The team should improve the UI to make it more user interactive and visually appealing.
7. **Database Revisit**: So far, database is constructed with minimal algorithm on timing complexity of data retrieval. However, if AskCrowd were to be continued, database structure needs to be revisited to ensure minimal retrieval time due to increasing number of users and polls.

## 9.0 Individual Contribution

This section shows the workload breakdown in the process of AskCrowd deployment. The following table breaks workload into several areas of implementation. Primary and secondary contributors for each area are shown.

| Area of Implementation | Primary Contributor | Secondary Contributor |
|---|---|---|
| **Infrastructure** | | |
| Frontend Design | Hao Jin | Louis Chen |
| Backend Design | Louis Chen | Hao Jin |
| AWS Lambda/API Gateway | Hao Jin | Louis Chen |
| AWS DynamoDB | Louis Chen | Hao Jin |
| AWS S3 | Louis Chen | Hao Jin |
| AWS SES | Hao Jin | Louis Chen |
| **Advanced Features** | | |
| Automatic Poll Suggestion | Louis Chen | Hao Jin |
| Search and Filter | Hao Jin | Louis Chen |
| Email Sharing | Hao Jin | Louis Chen |
| Google Search | Hao Jin | Louis Chen |
| Comments | Louis Chen | Hao Jin |
| **Other** | | |
| Documentation | Hao Jin & Louis Chen | |

## 10.0 Revision Control

Source code of AskCrowd has been revision-controlled on GitHub. The following table outlines the information of contributors who have co-worked during the development process.

| **Contributor** | **Email** | **GitHub ID** |
|---|---|---|
| Hao Jin | hao.jin@mail.utoronto.ca | erjin |
| Louis Chen | louis.chen@mail.utoronto.ca | louistjchen |

The repository is currently set in private mode. However, it will be accessible to the public at https://github.com/louistjchen/Ask-Crowd sooner or later.