Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Assignment T1: Preliminary Project Proposal

Part 1:
If your team name, team membership, planned programming language, platform and/or team github repository has changed since your team formation assignment, please explain. If not, simply state that nothing has changed. You need permission from the instructor *in advance* (before you submit this assignment) to change to any language other than C++, Java, Python and to change to any platform other than Linux, Mac, Windows.

Nothing has changed.

General info:
- Team name: Team ASE#
- Team members:
    - Alina Ying (ay2355)
    - Muzhi Yang (my2739)
    - Tian Liu (tl3061)
    - Mengwen Li (ml4643)
- Programming language: C#
- Platform: Windows
- Team github: https://github.com/Alying/team_ase_sharp

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Part 2:
Write a few paragraphs that provide an overview of the service that your team would like to develop and answers these three sets of questions:

1. What will your service do? What kind of functionality or features will it provide?

As COVID-19 is getting more and more contained, travel demand has correspondingly risen. The service we want to create is a safe-travel recommender service (initially only within the United States, and then possibly including other countries later): it suggests travel-spots based on the state's current COVID-19 cases, weather conditions, and air pollution levels.

For our users, we want to allow the following features and functionality:

1. Providing the top-10 safe-travel recommendations within the specified country.
   To create this list: the service will have a decision engine that adds up weighted scores for each API we are using:
   a. *Covid Tracking Project:* https://covidtracking.com/data/api/version-2
   b. *AccuWeather:* https://developer.accuweather.com/apis
   c. *Purple Air:* https://www2.purpleair.com/

   To calculate the weighted scores: we initially plan to make it weighted more highly towards the COVID-19 cases in the state (e.g. 50% COVID-19 cases, 25% weather, and 25% air quality) since our goal is to provide a safe-travel recommender service while the pandemic is still going on. In the next phase, we want to also introduce the commenting system, which allows registered users to post their reviews for the places they have traveled to. Later iterations of the commenting system may include numeric ratings from commentators (e.g. how enjoyable they found their trip to this state on a scale of 1-5) to help rank the top-10 safe-travel locations.

   This will be the /recommendations/country/<country> endpoint (e.g. /recommendations/country/US).

2. Providing basic travel information (COVID-19 cases, weather, and air quality) about a specific state. If a user either (1) wants to look more closely at a recommended state or (2) wants to check the conditions of a specific state they're interested in, the service should provide the aforementioned information.

   This will be the /recommendations/country/<country>/state/<state> endpoint (eg. /recommendations/country/US/state/NY).

3. (Future, if time permits): providing a commentator system. This would allow users to comment on and/or rate a state they have traveled to if they wish to share their experiences, and all other users to read comments and ratings for a specific state. This would provide users the capability to provide feedback on any places they have been to, which adds an additional layer to our decision engine so that it can reflect user input.

   This would be the /comment/country/<country>/state/<state> endpoint (eg. /comment/country/US/state/NY).

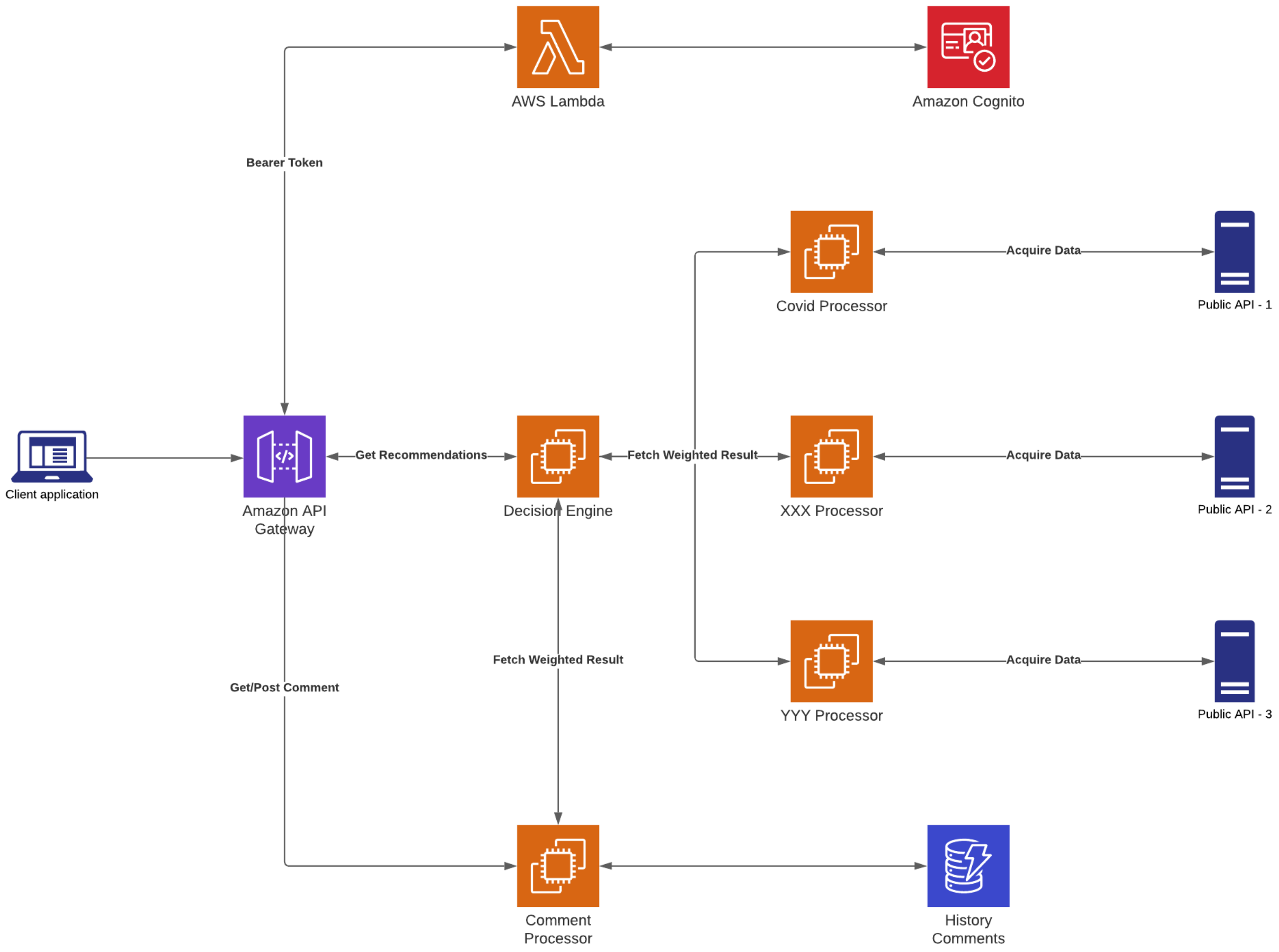
4. Account registration. We require all users to create an account in order to use our service. The account will hold their basic information, and their reviews (future).

   We are considering two options for login:
   (1) Using a Bearer Token, which will have a token endpoint
      (a) The input will be username and password.
      (b) Bearer Token will be returned from cloud service Amazon Cognito or Google Email (Columbia).
      (c) Pre-request: the user has to register in the cloud service first.
      (d) Then, we append the token for each request.
   (2) Our own identity management; this will probably also include our own database.
      (a) User needs to input their username and password for each request.
      (b) We check in the database by ourselves to verify their identity.

   This will include the /login/registration and /login/token endpoint.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Refer to the diagram below for a general structure of our service.



As a summary, we are intending to have business processor logic that uses 1-3 APIs (e.g. COVID-19, Weather, Air Quality) that feeds the decision engine, account management, and a commenting system (if time permits). We are intending to use the standard REST API, and support multiple clients. We will have no GUI.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

2. Who or what will be its users? What might they use the functionality for?

In terms of general population, we believe our service can be used by:

- People who want to travel for fun and stay safe at the same time. (Majority.)
- Students in summer break who must travel to relocate to another state for an internship or summer program.
- Corporate leaders who need to travel for work (e.g. if operations are run in multiple different states) and wish to understand how safe it is to do so at other locations.
- Politicians who must gauge how risky it is to travel to other states (to provide relief, support, campaign about certain issues, etc.)
- People who want to know how information about their own state's "travelability" (which likely correlates to how safe it is considered to live in the state).

General functionalities that users can use this service for include:

- Having a centralized go-to place for getting travel-related information during the COVID-19 pandemic.
- Getting recommendations for alternative travel destinations should they not know where to go for safe travel.
- (Future) Getting information from other users' reviews and experiences for a specific destination.

(Future) In terms of the commentator system, we separate our users as observers and commentators. Both types of users must have accounts in the system in order to use our service.

a. Our first set of users are observers, who simply want to find information about the best-recommended safe-travel destinations or about a specific state.
b. Our second set of users are commentators that wish to comment and rate a specific travel location they have traveled to.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

3. What kind of data will your service create or accumulate? What will the data be used for? For this assignment, you do not need to present your features as user stories or use cases.

Our service will create User data that will be maintained persistently through operations. In order to use our service, the individual must create an account (in our case, we can restrict it to students with Columbia emails so that they are "verifiable" in some sense). The User data will initially start off as very basic (e.g. name, uni, school), but in later iterations, can be populated with more information. For instance, if the commentator system is implemented, the User data can include all the comments they have made on any locations they visited.

As for account management:

1. If we go with our own database (login option 2), then user info will be our responsibility.
2. If we have time for adding a commenting system, the comments will be accumulated, so we have to set up a database (e.g. NoSQL) to hold it.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Part 3:
Write a few paragraphs that describe, at a high level, how you plan to test the functionality of your service <u>without</u> any clients, GUI or otherwise. (It's ok to use testing tools that have GUIs.) You will expand testing in the second iteration to include sample clients, but need to test stand-alone during the first iteration. Think in terms of testing your service as a whole, in addition to unit testing of individual subroutines, and answer these three questions:

1. How will you test that your service does what it is supposed to do and provides the intended functionality?

We intend to have to following tests in order to test that our service provides the intended functionality:

1. *Smoke test:* a small, quick validation test to check nothing breaks with a "happy path" (i.e. we assume a "good" user who doesn't give any breaking inputs)
2. *Unit tests:*
    a. For each API (COVID-19 API, Weather API, Air Quality API), we will write unit tests that cover all code paths: our usage of this API should give correct error codes and responses. It should also be able to handle non-200 error codes from other APIs (which are likely mocked out). Tests should include both good user input and breaking user input.
    b. For each class (except the controller, which is covered by the integration test), we will write unit tests to verify all the logic as well (we intend to use Moq to resolve dependencies).
3. *Integration tests:* we should be able to send requests to different endpoints using Postman, and make sure each response that we receive is an expected response. This should also include good user and breaking user input, as well as different error codes.
    a. We will check Service Layer ⇔ Data storage (RDB OR NSQL) calls.
    b. We will also check User ⇔ Service calls and test error code handling (400, 404, 500, etc).
4. *Mapping Test:* test if we can map data between different layers (controller layer - API model - business logic layer - domain model - DB layer - Storage Model).

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

2. How will you check that your service does not behave badly if its clients use it in unintended ways or provide invalid inputs?

This should be covered by the different unit tests and integration tests we have detailed in question 1. We should be able to handle different user input (no matter "good" input or "invalid" input); in addition to the expected happy paths, we will write the tests to try to cover as many different and breaking cases as possible. Invalid inputs include:

- Invalid endpoints → 404, "Resource not found."
    - "/recommendation"
    - "/login"
- Invalid inputs.
    - non-existing country → 404, "Country not found / supported." (Initially we will only implement it for the United States).
    - non-existing state → 404, "State not found / supported."
    - non-existing and/or malicious field in post request → 400, will fail in the json deserialization step and return a bad request.

We will also hold the service in the cloud. By using an AWS API Gateway / Cloudflare, we can set inbound rules and thresholds for incoming requests.

We additionally have models for each different layer -> API / DOMAIN / STORAGE. We can add checks during the mapping process to make sure of data purity for the domain model. As long as the business logic part is protected, we should be in a good place.

3. How will you test that your service handles its data the way it's supposed to?

This should be covered by the different smoke tests, unit tests, and integration tests we have detailed in question 1. We should be able to walk through our service like external users and receive the correct responses / results. These include the following endpoint paths:

1. /recommendations/country/<country> should lead the user to safe-travel locations within the specified country.
2. /recommendations/country/<country>/state/<state> should lead the user to safe-travel information about the specified state.
3. /comment/state/<state> should lead the user to comments and reviews about the specified state.
4. /login/registration should be an httppost (with registration info)
5. /login/token should be an httppost (with username and password)