Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Assignment T2: Revised Project Proposal

Part 1:
State the date on which you met with your IA mentor. This should be in the past, not the future.

Date met with IA mentor: October 19, 2021

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.

To explain the features and functionality we will implement, these are our user stories:

1. As a Traveler user, I want to be able to find the top-10 safe-travel recommendations within a specified country so that I can know where I can travel reasonably safely.
   My conditions of satisfaction are:
   a. When my proper credentials are provided (i.e. logged into my account), I get the top-10 safe-travel recommendations.
   b. I can look through comments about the country I searched for. These are from registered users' reviews for the places they have traveled to within the country and could include numeric ratings (e.g. how enjoyable they found their trip to this state on a scale of 1-5).

   In order to do so, I want to be able to access a /recommendations/country/<country> endpoint (e.g. /recommendations/country/US). Then, the service will attempt to create the list through a decision engine that adds up weighted scores through APIs like the below:
   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: the service should be weighted more highly towards the COVID-19 cases in the state (e.g. 50% COVID-19 cases, 25% weather, and 25% air quality) since my goal is to have a safe-travel recommender service while the pandemic (the highest health-risk at the moment) is still going on. The ratings for the states within the specified country might also be part of the weighting in the decision engine that ranks the top-10 safe-travel locations list for me.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

2. As a Location Inquiry user, I would like to know basic travel information about a specific state (for other countries: province, etc) so that I can decide for myself whether I want to travel here or not.
   My conditions of satisfaction are:
   a. I get basic travel information only when my proper credentials are provided (i.e. I am logged in)
   b. I receive COVID-19 information of my specified state.
   c. I receive weather information of my specified state.
   d. I receive air quality information of my specified state.
   e. I can look through comments/ratings about the state I searched for.

   When I hit the /recommendations/country/<country>/state/<state> endpoint (eg. /recommendations/country/US/state/NY), I want to be able to see COVID-19, weather, and air quality information for a specific state I'm interested in. I would also be able to look through comments and/or reviews for the specific state.

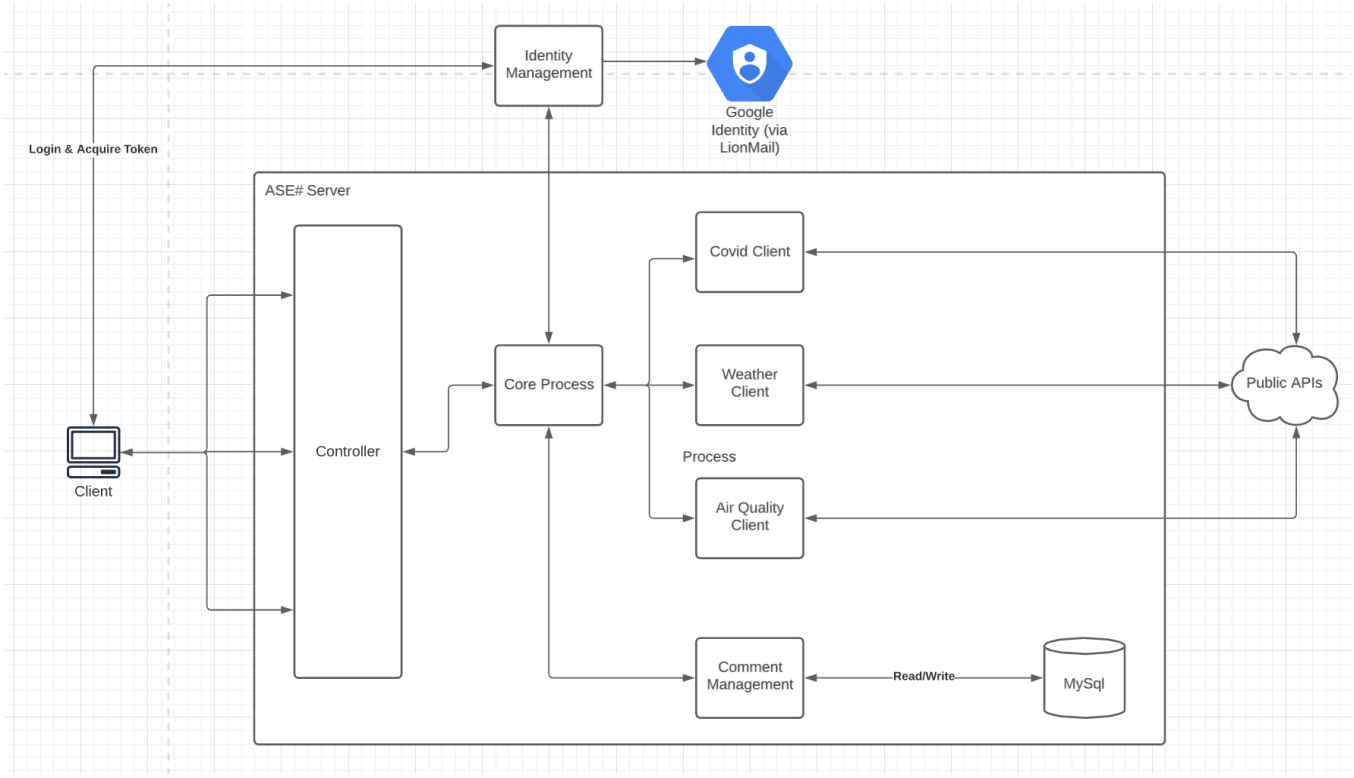
3. As a Commentator user, I want to be able to provide comments and/or ratings on a location that I have already traveled to add to the collective knowledge of the travel community.
   My conditions of satisfaction are:
   a. I can provide a comment only when my proper credentials are provided (i.e. I am logged in)
   b. My comments and/or ratings should immediately be visible on the server.
   c. I can choose the location to comment and/or rate on.

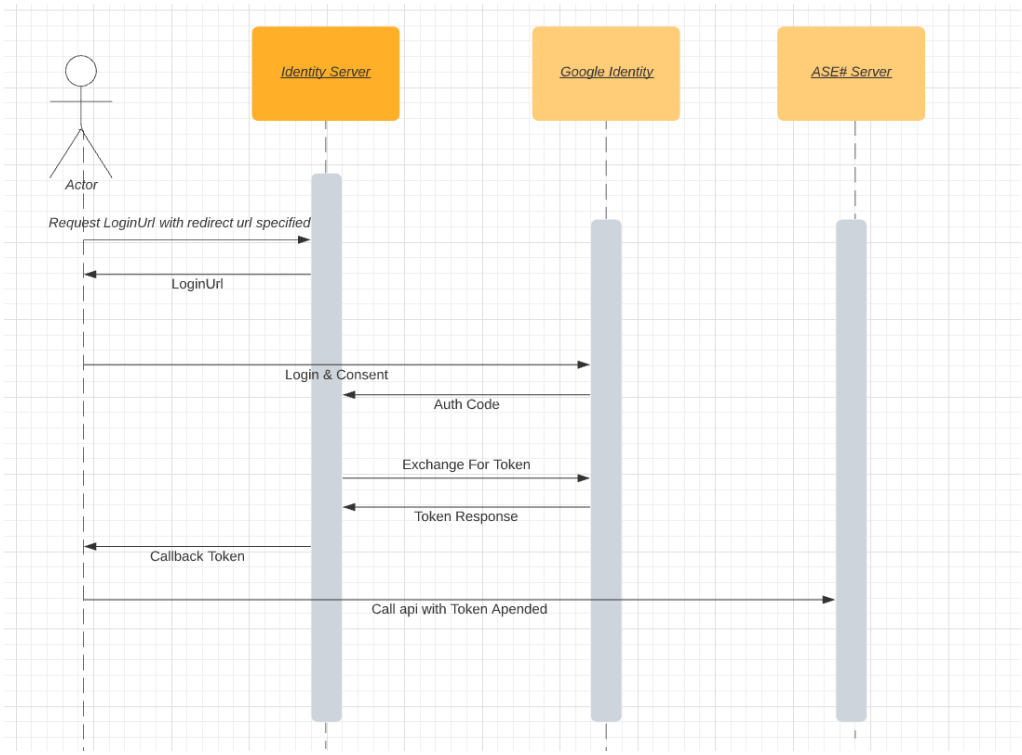
   When I access the /comment/country/<country> or /comment/country/<country>/state/<state> endpoints (eg. /comment/country/US/state/NY), it should allow me to write comments and see comments/ratings that other Commentator users have left for the specified location. My comments and/or ratings would be helpful to other users, who can see how my experience was and make their own decisions accordingly.

Note: We also considered a Passport user story, who would be able to find safe-travel locations depending on the current travel restrictions and their passport/visas. However, we decided not to implement this user story (or only implement it if we had the time to do so), since the list of restrictions seems pretty complicated to implement or hard-code.

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

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





Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

As a summary, we are intending to have business processor logic that uses 1-3 APIs (e.g. COVID-19, Weather, Air Quality) that feed the decision engine, account management, and a commenting system. We are intending to use the standard REST API, and support multiple clients. We will have no GUI. For login, we will be using an OAuth2 token; the user will have a callback listener to receive the token, given the correct username and password (as detailed above).

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

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

The users, as described by the user stories, are as follows.

Traveler users are users that wish to travel safely, and could include:

- People who want to travel for fun and stay safe at the same time.
- Politicians who want to gauge the least-risky states to travel to (to provide relief, support, campaign about certain issues, etc.) or who want a general sense of travelability.

Location Inquiry users are users that are curious about a specific location's travel information, and could include:

- 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 specific locations.
- People who want to know information about their own state's "travelability" (which likely correlates to how safe it is considered to live in the state).

Commentator users are users that want to comment on a specific state or country, and could include:

- Residents who want to talk about their state from their point of view.
- People who have traveled to certain locations and wish to leave specific feedback for the travel community to know.

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?

User data: we require all users to create/log in with an account to use our service (in our case, we can also restrict it to students with Columbia emails so that they are "verifiable" in some sense). The account will hold their basic information, and their reviews. This User data in the database (e.g. NoSQL) will be maintained persistently through operations, and will initially start off as very basic (e.g. name, uni, school), but in later iterations, can be connected with more information (e.g. comments).

Comments: we will set up the database to hold and accumulate all the Comment data, and it will include the specific user who wrote it, the state the comment is written for, and country the comment is written for, and the start date and end date of the trip. These will also be maintained persistently.

Ratings: we will set up the database to hold and accumulate all the Ratings persistently as well, which will be on a numeric scale (e.g. 1-5, 1-10), and it will include the specific user who made the rating, the state the rating is written for, the country the rating is for, and the start date and end date of the trip.

For the data we pull from the APIs, we can store a sanitized version of the data with a TTL (time-to-live) expiration time which automatically deletes the data after it expires (implemented by NoSQL already). The point of the TTL is to always have somewhat up-to-date COVID-19, weather, and air quality information.

- *COVID-19 info:* We will receive COVID-19 data from the API and should store the sanitized version (including relevant information) in our database somewhere in order to make decisions with our decision engine for the recommendation list, as well as be presented as information for specific locations.
- *Weather info:* We will receive Weather data from the API and should store the sanitized version (including relevant information) in our database somewhere in order to make decisions with our decision engine for the recommendation list, as well as be presented as information for specific locations.
- *Air quality info:* We will receive Air Quality data from the API and should store the sanitized version (including relevant information) in our database somewhere in order to make decisions with our decision engine for the recommendation list, as well as be presented as information for specific locations.

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 without 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 will test all our user stories.

For the Traveler user, we will test to make sure they can hit the /recommendations/country/<country> endpoint, and receive a response with a list of recommended safe-travel locations and comments for the specified country. We will test the U.S., as well as other countries (depending on if they are implemented or not, the response will return valid information, or return an informational 404 Not Found)

1. User sends GET request to /recommendations/country/<country> endpoint
2. Service checks if user has OAuth token (meaning they are logged in). If they are not, return 401 for authentication.
3. Service queries a database to find necessary data (e.g. COVID-19, Weather, Air Quality data); if data is expired, call relevant APIs to repopulate the information with newer data.
4. Service runs its decision engine to find a ranking of safe-travel locations.
5. Service reports safe-travel recommendation list and comments for the specified country

For the Location Inquiry user, we will test to make sure they can hit the /recommendations/country/<country> and /recommendations/country/<country>/state/<state> endpoints, and receive a response with the COVID-19, weather, air quality, and comment information for the specified state. We will test all 50 states of the United States. Specifically, when these users hit the endpoint with the “get” method and valid country and state code, we will return the comments that are already in the Comment database for this specific country and state combination. This test can be achieved by initially inserting several fake entries into the Comment database, and checking to make sure that these fake entries are being returned.

1. User sends GET request to /recommendations/country/<country> or /recommendations/country/<country>/state/<state> endpoint (we will test both)
2. Service checks if user has OAuth token (meaning they are logged in). If they are not, return 401 for authentication.

3. Service queries a database to find necessary data (e.g. COVID-19, Weather, Air Quality data); if data is expired, call relevant APIs to repopulate the information with newer data.
4. Service reports COVID-19 information, weather, and air quality information, as well as comments for the specified state

For the Commentator user, we will test to make sure they can hit the /comment/country/<country>/state/<state> endpoint, "post" their comment and/or rating, and receive a response with existing comments. Specifically, when these users hit this endpoint with the "post" method and valid country and state code, as well as a valid request body, this comment will be inserted into the Comment database. We can verify this by comparing the Comment database before and after the request, and make sure that the comment from the user's post request is inserted correctly.

1. User sends GET request to the /comment/country/<country> endpoint or /comment/country/<country>/state/<state> endpoint
2. Service checks if the user has an OAuth token (meaning they are logged in). If they are not, return 401 for authentication.
3. User sends POST request to a specific endpoint with their written comment/rating for the country/state
4. Service checks if the user has an OAuth token (meaning they are logged in). If they are not, return 401 for authentication.
5. Database is populated with new user comment information
6. Service reports a successful post

We will make sure to test that the user account registration/login process works and test with clients with incorrect/correct tokens: if the user does not have an account, they will not be allowed to successfully send requests to any of the service's endpoints; if the user has an account and correctly verifies themselves (username/password), then they can hit the endpoints and send requests.

As usual, we will have the necessary unit tests, smoke tests, and mapping tests for all the code logic we have as well.

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?

Please see Swagger documentation, which also provides more tail on example requests and responses for endpoints under different situations. (Click 'Hide Editor and Nav' on the left side, and you will be able to see all our documented GET, POST, and PUTs)

We should be able to handle different user input (no matter if it is "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. When the system errors out, it should not shut down but rather return the error code and message. Service misbehaviors and invalid inputs include:

If any of the users queries an invalid endpoint, we should return 404, "Resource not found." Examples include the below:

- "/recommendation" (invalid endpoint due to typo: 's' is missing at the end)
- "/login" (part of endpoint missing: need to be either /login/registration or /login/token)
- "/comment/state" (incorrect path: should be /comment/country/state)

If an unauthorized user attempts to send a request, we will return 401, "Unauthorized user." Examples include the below:

- user tries to access any valid endpoints without login token
- user tries to comment without login token
- user tries to rate without login login token
- user sends a request, logs out, and tries to send another request

If a Traveler or Location Inquiry user queries with malformed user inputs, we return as below:

- non-existing country →return 404, "Country not found/supported." (Initially we will only implement it for the United States).
- non-existing state → return 404, "State not found/supported." (If country and state don't match, it should return this error code because the state doesn't exist under this country)
- non-existing and/or malicious field in post request → return 400, will fail in the JSON deserialization step and return a bad request.

If the Traveler or Location Inquiry user queries for any endpoints, and a third party service is down, we will return 404, "Third party service is down" (API calls return an error or unexpected data). Examples of third-party services include: Covid Tracking Project, AccuWeather, and Purple Air.

If a Commentator user tries to post any malformed user inputs, we return as below:

- comment is too long (we can set this specified length later, mainly want to reduce spam) → return 400
- comment has SQL code in it (this could mess with posting to our database) → return 400

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?

We test getting default recommendations for Traveler users:

- There should be no input.
- Traveler users should be able to receive a list of the top 10 objects for recommendation.
- Status code - 200

We test getting recommendations for Location Inquiry users by country & state:

- The Location Inquiry user's input must be a valid country and/or state code and it is in the path.
- Location Inquiry users should be able to receive information about the specific country or state, which includes COVID-19 cases, weather, and air quality data.
- Status code - 200

We test getting comments by country & state for Traveler and Location Inquiry users.

- The Traveler or Location Inquiry user's input must be a valid country and/or state code and it is in the path.
- The Traveler or Location Inquiry users should be able to receive a list of comments which are ordered by the latest update time.
- Status code - 200

We test posting comments by country & state for Commentator users.

- The Commentators user’s inputs must be valid for the country and/or state in the path.
- The Commentator user’s inputs are valid JSON request bodies that only contain valid fields defined in the API documentation.
- Commentator users should receive a location id.
- Status code - 201

Alina Ying
Muzhi Yang
Tian Liu
Team ASE# Mengwen Li

Part 4:

List the libraries, frameworks, tools, etc. you think you might use. Make sure to include (at least) a style checker, static analysis bug finder, test runner and coverage tracker suitable for your language and platform. You will not be held to this list.

Style checker: StyleCop
Static analysis bug finder: Coverity
Test runner: XUnit + Moq
Coverage tracker: NCrunch -- pay, 30 day trial [TBD]
Framework: ASP.NET
Runtime: .Net Core 3.1
Libraries (nuget packages): NewtonSoft.Json, Dapper, Optional, etc.