Teamwork homework1
Assignment T1: Preliminary Project Proposal

Team name: Initial Dream
Member names: Yifan Li (yl4287), Shuwan Yao(sy2884), Zixian Zhu(zz2689), Yuqiu Gan(yg2684)

Part 0:

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, you do not need to include part 0 in your submission.

Part 1:

Write a few paragraphs that provide an overview of the software engineering project that your team would like to do and answers all five of the numbered questions.

1. What will your project do?

Our project aims to develop a Healthy Code System for the Columbia University Community(Staff & Students) and Tourists.

We collect users' data including identity information and travel history and evaluate the risk level of being affected by Covid-19 of users. Then we will show a different Healthy Code to each user according to its risk level.

"Yellow Healthy Code"

For example, if a user traveled to a high risky state in the last 14 days without symptoms, we'll assign him/her "Yellow Healthy Code" to indicate he/she has the risk of being affected by Covid-19.

User can clear his/her "Yellow Healthy Code" and gain a "Green Healthy Code" in two ways:

The first way, quarantining himself/herself for 14 days after returning from a highly risky area without any symptoms.

The second way, getting a Covid-19 test as soon as possible. Once the result shows negative, he/she can clear "Yellow Healthy Code" and get a "Green Healthy Code".

"Red Healthy Code"

For users who are already affected by Covid-19(Covid-19 test positive or having symptoms), he/she will be assigned the "Red Healthy Code" directly.

"Green Healthy Code"

For users who never traveled to any high risky areas in the recent 14 days and don't have any symptoms or users who got recent Covid-19 negative reports or users who had "Yellow Healthy Code" and satisfied conditions of clearing risk, they will be assigned "Green Healthy Code".

Use of "Healthy Code"

"Healthy Code" is a certification of a user. Only users who hold the "Green Healthy Code" can access the Columbia University campus and buildings, labs. This system will associate with Columbia University ID, those who have "Green Healthy Code" can tap their Columbia University ID Card on the sensor and access campus, buildings, labs. For tourists, they will get a temporary account, "Green Healthy Code" holders can access campus, buildings, labs via showing "Healthy Code" to doormen or administrators.

2. Who or what will be its users? Your project must impose registration, authenticated login and timeout/explicit logout, so your answer should say something about this.

Users: people who want to enter the campus, including authenticated users (students and staffs) and guests(tourists, visiting scholars)

For authenticated users, they already have unis, so they can just login in.

For guests, they will be allocated a temporary CUID for logging in.

Once successfully logged in, people can enter any part of the campus within the six hours. After the time expires, they will automatically log out and must log in again.

3. Your project must be demoable, but does not need a GUI if there's a command line console or some other way to demonstrate. (All demos must be entirely online, there will be no in-person demos.) What do you think you'll be able to show in your demo?

In the demo, we can show the process of how a user interacts with our system and get a healthy code. Also, we can show how the manager views the information of students and staff who use our system today and the current status of users.

4. Your project must store and retrieve some application data persistently (e.g., using a database or key-value store, not just a file), such that when your application terminates and starts up again some arbitrary time later, the data is still there and used for later processing. What kind of data do you plan to store?

We will use a database to store our data persistently.

There will be at least 2 databases:

1. One database will restore users' healthy status records.
2. Another database will restore all users' CUID.

5. Your project must leverage some publicly available API beyond those that "come with" the platform; it is acceptable to use an API for external data retrieval instead of to call a library or service. The API does not need to be a REST API. There are many public APIs linked at https://github.com/public-apis/public-apis

(Links to an external site.)

and https://github.com/n0shake/Public-APIs

(Links to an external site.)

. What API do you plan to use and what will you use it for?

Google search API: for scraping information of COVID-19 data.

Part 2:

Write three to five user stories for your proposed application, constituting a Minimal Viable Product (MVP); registration/login/logout should not be included among these user stories, nor should 'help' or other generic functionality. That is, your application should do at least three application-specific things. Use the format

< label >: As a < type of user >, I want < some goal > so that < some reason >.

My conditions of satisfaction are < list of common cases and special cases that must work >.

The type of user (role) does not need to be human. You may optionally include a wishlist of additional user stories to add if time permits. Keep in mind that the type of user, the goal, the reason (if applicable within the system), all the common cases and all the special cases must be testable and demoable.

As a student/staff, I want to submit the form and update my daily health information so that I would get a green pass code.

As a manager, I want to review the information about people who get into campus so that I could log into the platform and check the information table.

As a tourist or visiting scholar without UNI, I want to get a temporary healthy code so that I could enroll in the system and get into campus temporarily.

Part 3:

Explain how you will conduct acceptance testing on your project. This means that every MVP user story must be associated with a plan for user-level testing. The test plan should address both common cases and special cases. Discuss sample inputs the user or client would enter and the results expected for the corresponding test to pass vs. fail. Note inputs might come from files, network, devices, etc., not necessarily from a GUI or command line, and results might involve changes in application state, files, outgoing network traffics, control of devices, etc., not necessarily outputs via a GUI or command line. You may optionally discuss testing plans for your wishlist additional user stories, if any.

If a user logs in as student/staff, reports no symptom and no explosion to potential risk, and gets a green-pass code, the test case passes.

If a user logs in as student/staff, reports no symptom and explosion to potential risk, and gets a yellow code, the test case passes.

If a user logs in as student/staff, reports symptoms but no explosion to potential risk, and gets a red code, the test case passes.

If a user logs in as student/staff, reports some symptom and explosion to potential risk, and gets a red code, the test case passes.

Manager will be assigned a superuser account. When a manager uses its superuser account to log in, he will have access to all the information about people who visited campus. The interface of superusers' account will be different from others, and will receive a message "Welcome Superuser!". If a message pops out, then the test will pass, else it will fail.

When a tourist wants to use this system, he/she will click a button to indicate he/she doesn't have a UNI, then the system will automatically assign this user a temporary account which will last for 24 hours. If this user gets an account number and associated passcode, this test will pass or it will fail.

Part 4:

Identify the specific facilities corresponding to JDK, Eclipse, Maven, CheckStyle, JUnit, Emma, Spotbugs, and SQLite that your team plans to use. That is, state what you plan to use for compiler/runtime (or equivalent), an IDE or code editor, a build tool (or package manager if 'build' not applicable), a style checker, a unit testing tool, a coverage tracking tool, a bug finder (that's not just a style checker), and a persistent data store appropriate for your chosen language(s) and platform(s). If different

members of the team plan to use different tools, please explain. It is ok to change your choice of tools later after you start developing your application.

We plan to use eclipse, wix, Maven, Checkstyle, Junit, Emma, Spotbugs, Postman, Mysql for our project.