# Part 2 + Bonus Test

Tamara Seban

# Part 2 / First Question

1. What is the purpose of the group isolation feature? Is it to prevent further spread of the virus within a household, or to track and monitor the spread of the virus within a specific geographic area?
2. Is the group isolation feature optional or mandatory for users who have been exposed to the virus? Can users opt out of group isolation if they do not want to participate?
3. How will the system determine which people should be included in group isolation? Will users manually enter the information or will it be automatically detected based on location data or other factors?
4. What is the criteria for determining the geographic radius for group isolation? Will it be a fixed radius, or will it be based on the specific location and characteristics of each group?
5. How will the system ensure that users' privacy and data security are protected during the group isolation process? Will any personally identifiable information be shared with third parties?
6. Will the system provide any additional support or resources for users who are in group isolation, such as access to medical advice or supplies?
7. What is the expected timeline for the implementation of the group isolation feature, and how will it be rolled out to users?
8. How will the system handle any technical issues or bugs that arise during the implementation of the group isolation feature? Will users be notified of any changes or updates to the feature?
9. Do users need to be logged in to create isolation?
10. Is it necessary for users to log in with a username and password to access the system?
11. Who is allowed to create group isolation? After all, dishonest people can create isolations and lie that they have been exposed to the virus.
12. Will users receive a confirmation message when isolation is created and a notification is sent to people exposed to the virus?
13. Can users cancel the isolation they created?
14. Will error messages be displayed when a field is formatted incorrectly or when there is an error on the server side?
15. What is the purpose of the feature?

16. Is the feature designed to identify groups of people who were exposed to the virus in a common radius?
17. Will the feature allow for easy isolation and updates to their status in the system?
18. What are the functional requirements of the system?
19. What are the user interface requirements?
20. What are the performance requirements?
21. Can the system handle a large number of requests from users?
22. Can the system update the database in real-time to reflect changes in isolation status?
23. What are the security requirements?
24. Is the system designed with security in mind to protect user data?
25. Does the system use secure protocols to transmit data between the server and the client?
26. What are the testing requirements?
27. Will the system undergo extensive testing to ensure that it is reliable and accurate?
28. Will the system be tested in various scenarios to ensure that it is robust and able to handle unexpected inputs and conditions?
29. What does "NC - the recipient of the geographic data where the virus was" mean?
30. What is the recipient supposed to do with the geographic data?
31. What database schema is being used to store the isolation data, and what endpoints are being used to retrieve or update the data?
32. How are the phone icons of the entire population of Israel being used to determine who has been exposed to the virus? Is this data being collected by the system, or is it being provided by a third-party API?
33. How is the UI communicating with the server to send the request for group isolation? Is this being done via an API endpoint, or some other mechanism?

The feature design seems promising, but additional details would be helpful to fully understand the implementation and ensure that all requirements are being met.

# Part 2 / Second Question

Here are some suggestions for tests for the feature:

1. Unit tests:
   - Test the patch route with valid and invalid requests to ensure the correct status codes are returned.
   - Test the function that updates the database for a group of people to ensure that it correctly handles edge cases such as empty arrays or invalid dates.

- Test the integration with the Google Maps API to ensure that it correctly retrieves and displays the location data.
- Test the integration with the phone icons API to ensure that it correctly identifies the people who need to be updated.

2. Integration tests:
- Test the feature end-to-end to ensure that it works as expected from the UI to the server and the database.
- Test the feature with different scenarios such as different group sizes, different exposure locations, and different recovery dates.
- Test the feature with different user roles and permissions to ensure that data is not accessible to unauthorized users.
- Test the feature with a large amount of data to ensure that it can handle the load and does not crash or produce errors.

3. User acceptance tests:
- Conduct user acceptance tests with a small group of users to ensure that the feature meets their expectations and needs.
- Collect feedback from users to identify any usability or functionality issues that need to be addressed.
- Use the feedback to improve the feature and update the documentation and tests accordingly.

Here are some additional edge cases to consider:

- What happens if the user selects an invalid location on the map? For example, if they select a location that is not within the bounds of Israel.
- What happens if the user selects a date of exposure that is in the future?
- What happens if the user selects a recovery date that is before the date of exposure?
- What happens if the user selects a recovery date that is in the future?
- What happens if the user selects a date of exposure that is too far in the past? For example, if the exposure occurred more than 14 days ago, and the recovery date is not within that range.
- What happens if the user tries to create a new isolation group for a set of people who are already in an existing isolation group? Do we allow them to be added to multiple groups, or do we prevent this?
- What happens if the user tries to add a person to an isolation group who is already marked as recovered from the virus? Do we allow this, or do we prevent it?
- What happens if the user tries to add a person to an isolation group who has already passed away? Do we allow this, or do we prevent it?

Here are some more tests we should conduct:

- Testing for incorrect or invalid data types being passed in the request body (e.g. passing a string instead of a date).
- Testing for potential security vulnerabilities, such as SQL injection or cross-site scripting (XSS) attacks.
- Testing for performance and scalability, especially when dealing with a large number of users and requests.
- Testing for compatibility with different devices, browsers, and operating systems.
- Testing for handling unexpected errors or exceptions and verifying that appropriate error messages are displayed to users.
- Testing for handling edge cases such as when a user is part of multiple groups or when a user has recovered from the virus but was previously in isolation.
- Testing for handling situations where a user is already in isolation and needs to be added to a new group.
- Testing for handling situations where a user is added to a group but later their location or exposure date changes, and verifying that their information is updated accordingly.

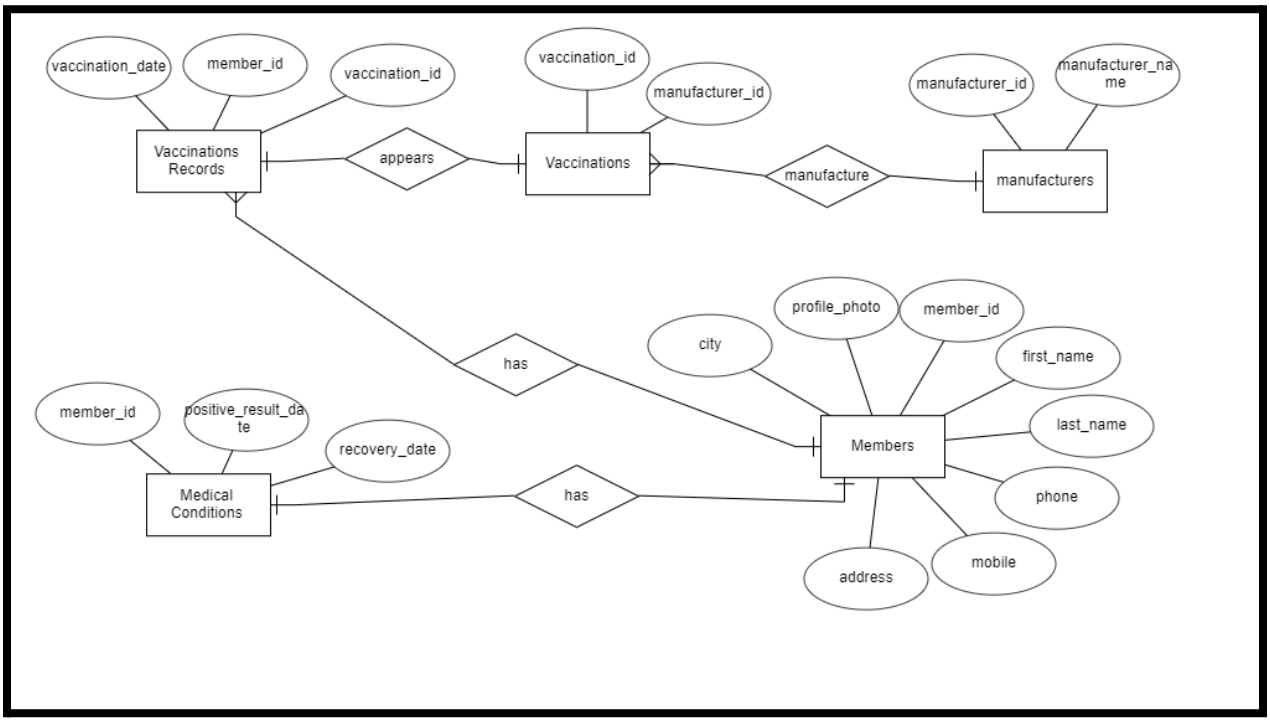
Some validation tests:
1. Empty or invalid response from Google Maps API when fetching location data.
2. Invalid or missing data in the array of people who need to be updated.
3. Duplicate entries in the array of people who need to be updated.
4. Testing with a large number of users in the array of people who need to be updated to ensure that performance is not impacted.
5. Testing with various timezones to ensure that date/time data is handled correctly.
6. Invalid or missing data in the request body such as invalid dates, missing location data, or missing required fields.
7. Testing with different browsers to ensure that the UI and map functionality works as expected.
8. Testing with slow network connections to ensure that the feature works optimally even in poor network conditions.
9. Testing with different device types such as desktop, mobile, and tablet to ensure that the UI is responsive and works correctly on all devices.
10. Testing the recovery date against the exposure date to ensure that it is valid and does not occur before the exposure date.

# BONUS / First Question

```
ALTER TABLE Members
ADD COLUMN profile_photo VARCHAR(255);
```

-- Insert profile photos into Members table
UPDATE Members SET profile_photo = 'https://example.com/john_doe.jpg' WHERE first_name = 'John' AND last_name = 'Doe';
UPDATE Members SET profile_photo = 'https://example.com/jane_doe.jpg' WHERE first_name = 'Jane' AND last_name = 'Doe';

# BONUS / Second Question




# BONUS / Third Question / A

```
/**
 * get: /members/sickCount:
 *     Summary: Retrieve the number of members who were sick in the last month
 *     responses:
 */

app.get("/sickCount", async (req, res) => {
  try {
```

```
    // Get today's date
    const today = new Date();
    // Calculate the date 30 days ago
    const thirtyDaysAgo = new Date(today.setDate(today.getDate() - 30));

    // Query the database for sick members within the last month
    const { rows } = await pool.query(
      `
      SELECT COUNT(*) AS count, DATE_TRUNC('day', positive_result_date) AS date
      FROM MedicalConditions
      WHERE positive_result_date >= $1 AND recovery_date <= $2
      GROUP BY date
      ORDER BY date ASC
    `,
      [thirtyDaysAgo, today]
    );

    // Send the result as a JSON response
    res.json(rows);
  } catch (err) {
    console.error(err);
    res.status(500).send("Internal server error");
  }
});
```

# BONUS / Third Question / B

```
//Endpoint: GET /unvaccinatedMembersCount
//Description: Retrieves the number of members who have not been vaccinated at all
app.get("/unvaccinatedMembersCount", async (req, res) => {
  try {
    const { rows } = await pool.query(`
      SELECT COUNT(*)
      FROM Members
      WHERE member_id NOT IN (
        SELECT member_id
        FROM vaccinationsRecords
      )
    `);
    res.send(rows[0].count);
  } catch (err) {
```

```
    console.error(err);
    res.status(500).send("Internal server error");
  }
});
```