## Summary

### Progress since M7

On the project level, we generated a **new organization on GitHub(CPEN321Closet)** and transferred the original repository into the organization. Then, we enabled **Codacy** for code checking.

On the frontend side, We **finished the UI for complex logic** (a “get an outfit” button and like/dislike buttons). We also **added one more UI test** for the add clothes use case.

On the backend side, we **finished all complex logic** (get one outfit and get multiple outfits). We also **added one extra endpoint** to let the user change his/her opinion of an outfit.

Moreover, we **did the code review** and **fixed all the issues** shown on **Cadacy** on both the frontend and backend sides.

### Plans until M9

On project level, we will add more unit and integration tests.

On the frontend side, we will **allow users to update clothes and get multiple outfits**, and **finish the UI for users to update profiles.** We will also **implement more UI tests** to fully test our app, and improve the performance of getting an outfit.

On the backend side, we will **allow users to update their profile** (including their name, email, city information). We will also **improve the performance** when getting weather information, and this will also improve the performance when getting an outfit.

### Major decisions and changes

No change

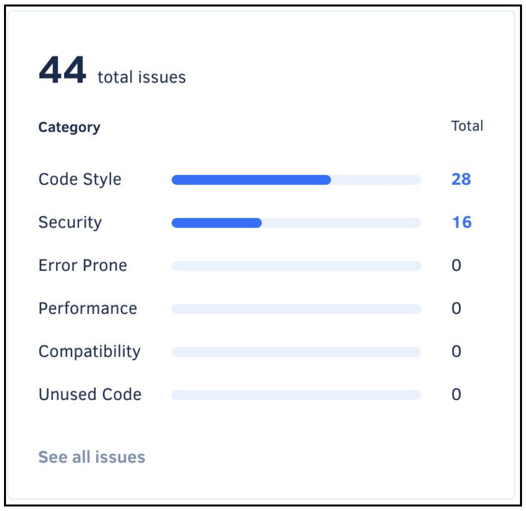
### Member contributions

- **Steven Yang**:
  Fixed **code issues** shown on Codacy and improve code integration and deployment
- **John Li**:
  Completed **complex logic**, added **user like/dislike logic**, and fixed **code issues** shown on Codacy
- **Zhuoyi Li**:
  Completed **like/dislike buttons** and fixed **code issues** shown on Codacy
- **Jingyang Sun**:
  Completed “**get an outfit” button** and fixed **code issues** shown on Codacy

## Automated Code Review

- **Run Codacy**
    - Here is the dashboard page of our Codacy https://app.codacy.com/gh/CPEN321Closet/closet/dashboard

- **Report the number of identified issues in each category**
    - Our initial report showed that there are 44 issues to be improved

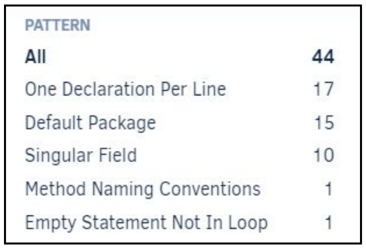


44 total issues

| Category | Total |
|---|---|
| Code Style | 28 |
| Security | 16 |
| Error Prone | 0 |
| Performance | 0 |
| Compatibility | 0 |
| Unused Code | 0 |

See all issues

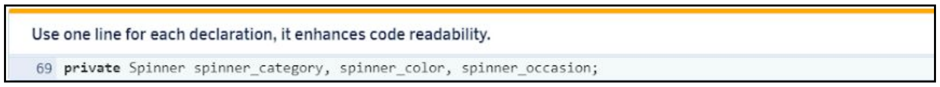
- **Report the number of identified issues in each of the identified problem patterns, Give a concrete example of an issue in each pattern**



| PATTERN | |
|---|---|
| All | 44 |
| One Declaration Per Line | 17 |
| Default Package | 15 |
| Singular Field | 10 |
| Method Naming Conventions | 1 |
| Empty Statement Not In Loop | 1 |

Concrete examples of an issue in each pattern:

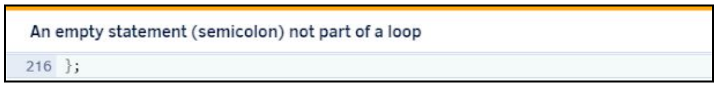
One Declaration Per Line



Use one line for each declaration, it enhances code readability.

```
private Spinner spinner_category, spinner_color, spinner_occasion;
```

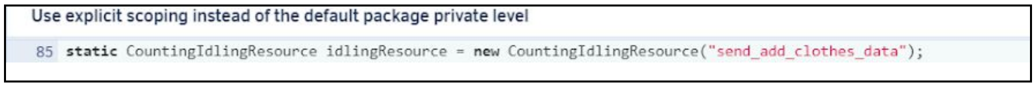
Empty statement Not In Loop



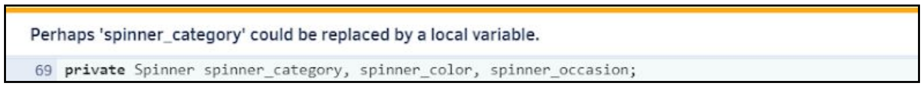
An empty statement (semicolon) not part of a loop

```
};
```

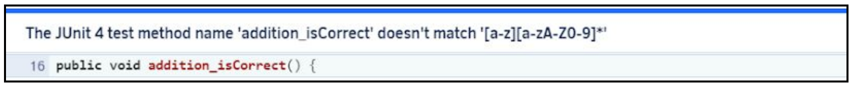
Default Package


Use explicit scoping instead of the default package private level
```
static CountingIdlingResource idlingResource = new CountingIdlingResource("send_add_clothes_data");
```

Singular field

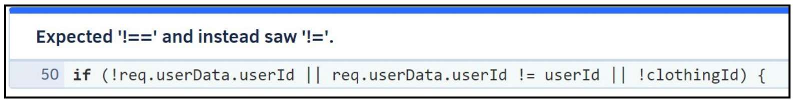

Perhaps 'spinner_category' could be replaced by a local variable.
```
private Spinner spinner_category, spinner_color, spinner_occasion;
```

Method Naming Conventions

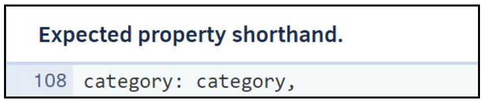

The JUnit 4 test method name 'addition_isCorrect' doesn't match '[a-z][a-zA-Z0-9]*'
```
public void addition_isCorrect() {
```

<u>Other related and common issues</u>

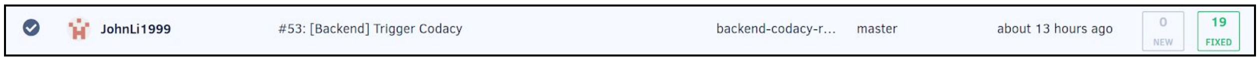
JavaScript equals

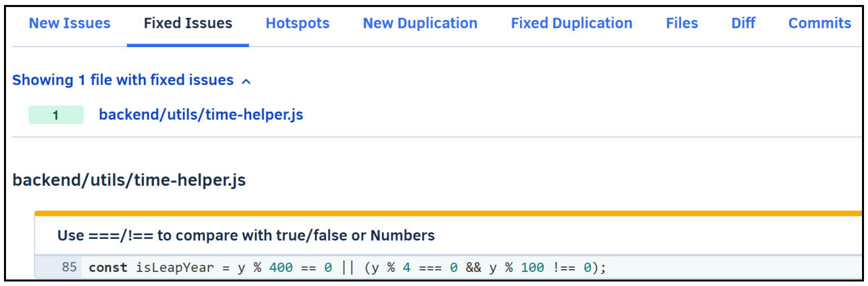

Expected '!==' and instead saw '!='.
```
if (!req.userData.userId || req.userData.userId != userId || !clothingId) {
```

Object literal shorthand (ES6 notation)


Expected property shorthand.
```
category: category,
```

- **Fix the identified issues and Run Codacy again**
    - Backend and frontend each fixed their related issues with pull requests (below examples of merged commits and fixed issue)


JohnLi1999 #53: [Backend] Trigger Codacy backend-codacy-r... master about 13 hours ago 0 NEW 19 FIXED

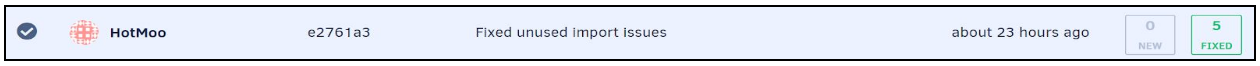

New Issues | Fixed Issues | Hotspots | New Duplication | Fixed Duplication | Files | Diff | Commits

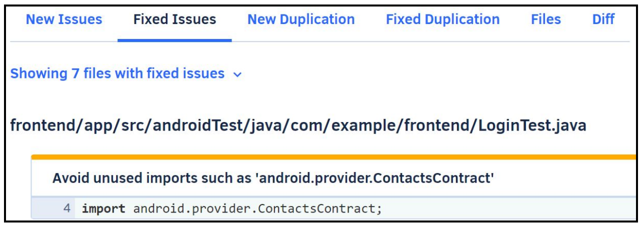
Showing 1 file with fixed issues

1 backend/utils/time-helper.js

backend/utils/time-helper.js

Use ===/!== to compare with true/false or Numbers
```
const isLeapYear = y % 400 == 0 || (y % 4 === 0 && y % 100 !== 0);
```


HotMoo e2761a3 Fixed unused import issues about 23 hours ago 0 NEW 5 FIXED


New Issues | Fixed Issues | New Duplication | Fixed Duplication | Files | Diff

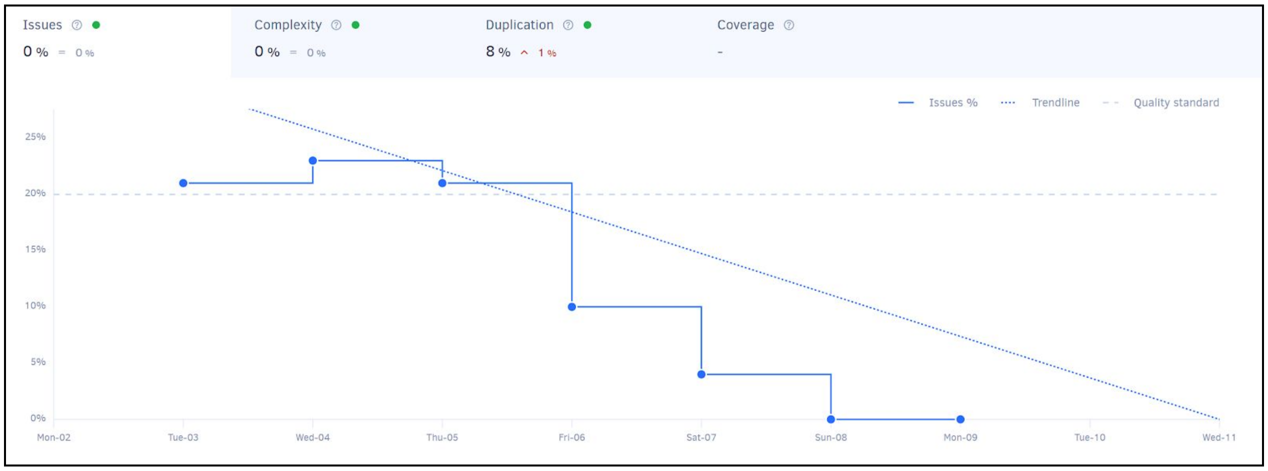
Showing 7 files with fixed issues

frontend/app/src/androidTest/java/com/example/frontend/LoginTest.java

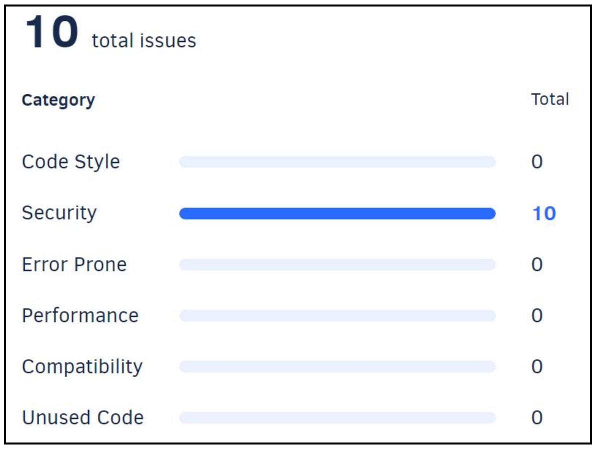
Avoid unused imports such as 'android.provider.ContactsContract'

```
import android.provider.ContactsContract;
```

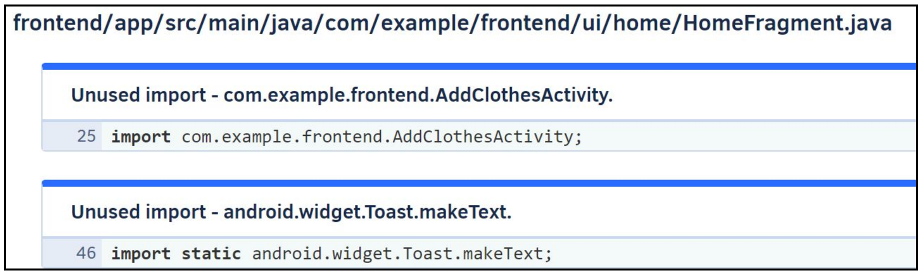
- We are able to continue and fix all issues over M8



- **Report the number of identified issues in each category and the number of identified issues for each pattern after fix**
  - Example issues after some fixing


**10** total issues

| Category | Total |
|---|---|
| Code Style | 0 |
| Security | 10 |
| Error Prone | 0 |
| Performance | 0 |
| Compatibility | 0 |
| Unused Code | 0 |


frontend/app/src/main/java/com/example/frontend/ui/home/HomeFragment.java

Unused import - com.example.frontend.AddClothesActivity.

```
import com.example.frontend.AddClothesActivity;
```

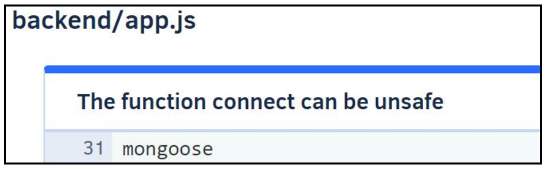
Unused import - android.widget.Toast.makeText.

```
import static android.widget.Toast.makeText;
```

- One image above is an example after we ran Codacy

  - We also mentioned issues that we are unable to fix but did provide mitigation strategies instead to better improve quality and safety

- **Issues that we are unable to fix**

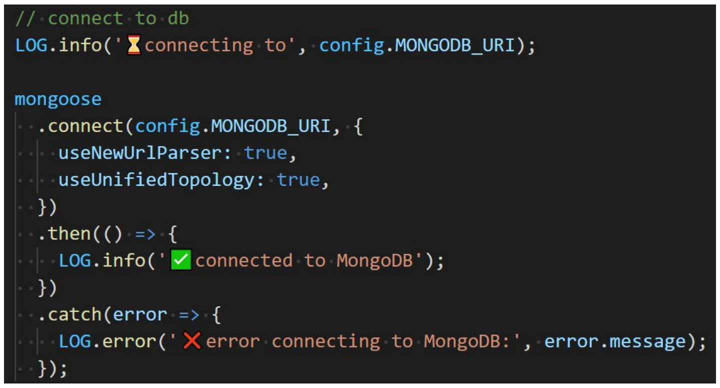


backend/app.js

The function connect can be unsafe

```
31 mongoose
```

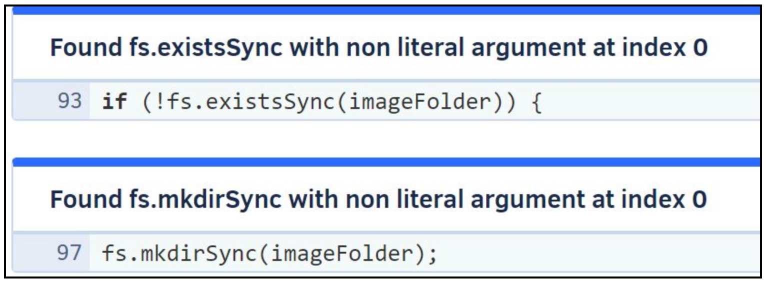
  - This issues is related to calling “connect” on mongoose
  - Issues reported by **Scanjs rules: Call_connect** ESLint_scanjs-rules_call__connect
    - The rule implemented ([https://github.com/mozfreddyb/eslint-plugin-scanjs-rules/blob/master/lib/rules/call_connect.js](https://github.com/mozfreddyb/eslint-plugin-scanjs-rules/blob/master/lib/rules/call_connect.js)) does overall enforces not to call “connect” on any object
  - We did follow official documentation provided by Mongoose to call connect to the database ([https://mongoosejs.com/docs/connections.html](https://mongoosejs.com/docs/connections.html))
  - To mitigate this issue, we have already implemented Promise catching on error is mongoose.connect throws error



```
// connect to db
LOG.info('⌛connecting to', config.MONGODB_URI);

mongoose
  .connect(config.MONGODB_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    LOG.info('✅connected to MongoDB');
  })
  .catch(error => {
    LOG.error('❌error connecting to MongoDB:', error.message);
  });
```

  - If such error is thrown, we would also be able to see it during automatic deployment log or manually log on our Azure VM



Found fs.existsSync with non literal argument at index 0

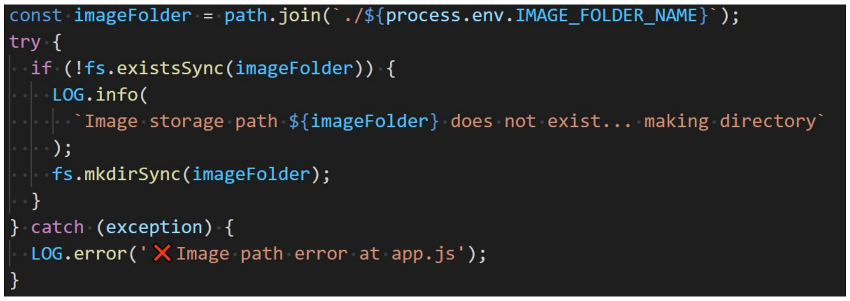
```
93 if (!fs.existsSync(imageFolder)) {
```

Found fs.mkdirSync with non literal argument at index 0

```
97 fs.mkdirSync(imageFolder);
```

  - These issues are related to how we are using a variable to call fs functions
  - This seems to be a common issue that it has an open Github issue
    - [https://stackoverflow.com/questions/63262683/how-to-fix-found-fs-readfile-with-non-literal-argument-at-index-0](https://stackoverflow.com/questions/63262683/how-to-fix-found-fs-readfile-with-non-literal-argument-at-index-0)
    - [https://github.com/nodesecurity/eslint-plugin-security/issues/65](https://github.com/nodesecurity/eslint-plugin-security/issues/65)

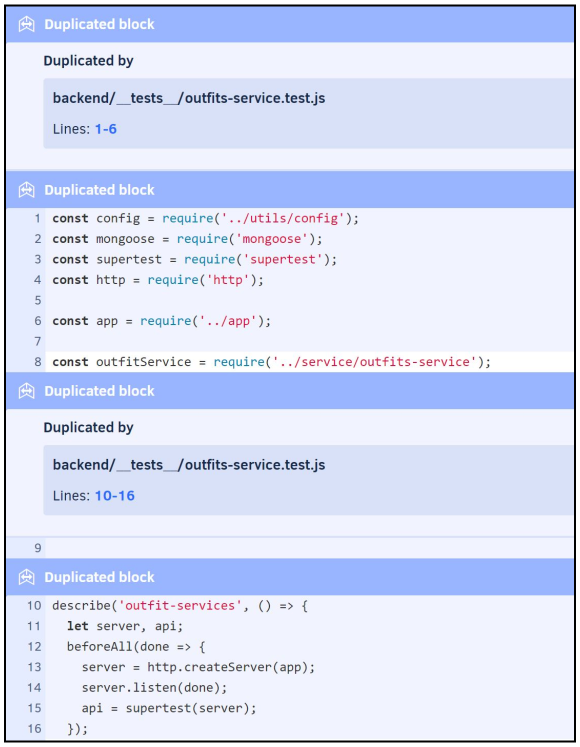
- Currently there are no official response other than disabling ESLint on this pattern or using absolute string path to call fs function
    - We could temporarily disable the pattern --- since this is not a major issue and we have unit test around fs functions (in image-controller) for checking --- until an official response from the Github open issue
    - Using absolute path defeats the purpose of using variable and relatively path
    - Since this is related to our logic of finding user images, we need relatively path to manage files and folders, so we can not simply switch to another method
- We implemented few mitigations and our own error checking to ensure better safety



```
const imageFolder = path.join(`./${process.env.IMAGE_FOLDER_NAME}`);
try {
  if (!fs.existsSync(imageFolder)) {
    LOG.info(
      `Image storage path ${imageFolder} does not exist... making directory`
    );
    fs.mkdirSync(imageFolder);
  }
} catch (exception) {
  LOG.error('❌Image path error at app.js');
}
```

    - We implemented try/catch to ensure our app.js does not completely crash, but instead send an error (although not needed, a good extra layer or safety)
    - Wrapping file checking inside if-else ensure we can also see the result
    - We applied our mitigation to similar issues of the same types

- **Other improvements**
    - Although not related to issues, for duplication, these are the result of unit test configuration and setups, and are not related to the actual core functions
        - Duplications are very minimal such that we are passing Codacy review
        - These setup codes are needed so that each test suite are able to run independently from each other
        - Putting these separately might cause more confusion when running or editing test cases



Duplicated block

Duplicated by

backend/__tests__/outfits-service.test.js

Lines: 1-6

Duplicated block

```
const config = require('../utils/config');
const mongoose = require('mongoose');
const supertest = require('supertest');
const http = require('http');

const app = require('../app');

const outfitService = require('../service/outfits-service');
```

Duplicated block

Duplicated by

backend/__tests__/outfits-service.test.js

Lines: 10-16

Duplicated block

```
describe('outfit-services', () => {
  let server, api;
  beforeAll(done => {
    server = http.createServer(app);
    server.listen(done);
    api = supertest(server);
  });
```

## Manual Code Review

### Report two major issues you found in your customer code

1. First, they delete the old id data when copy residence information from the "search" database to "recommend" database. They may lose reference if they try to search the residence data in the old database. This should be avoided.
2. Second, they randomly return four relevant search histories to the user as the recommendations. However, they may recommend duplicated data if the user searches a residence multiple times. This case should be avoided as well.

### Report two major issues that your peer team found in your code

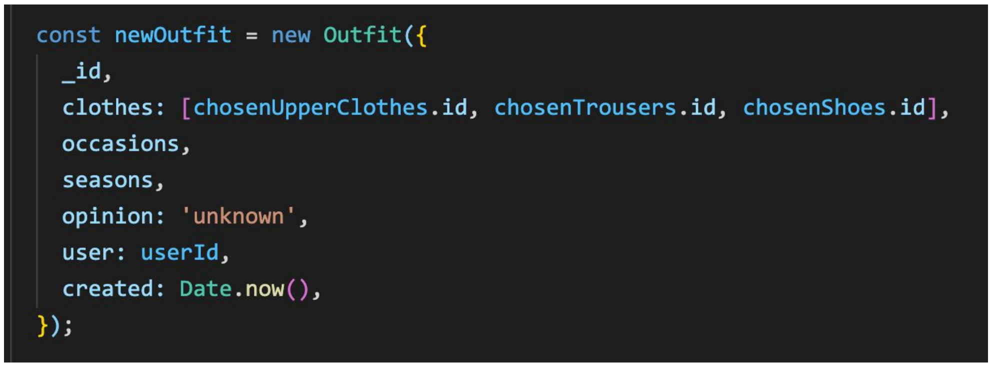
1. Javascript Date prototype and mongoDB use UTC as the default timezone, but the users can use the app in different time zones. So, there will be a mismatch between outfit created time and calendar event time (Google Calendar adjusts time zones for events time). This mismatch may let us generate the wrong outfit for users.
2. When we generate a new normal outfit, we first get all possible clothes combinations and exclude all the user-disliked combinations. Then we generate a new outfit using the rest of the combinations. However, if we have generated all possible combinations but the user dislikes all of them, the response will always be an empty object without a descriptive explanation. This should be avoided.

**Specify how you addressed these issues**

- **Solution of the first problem**

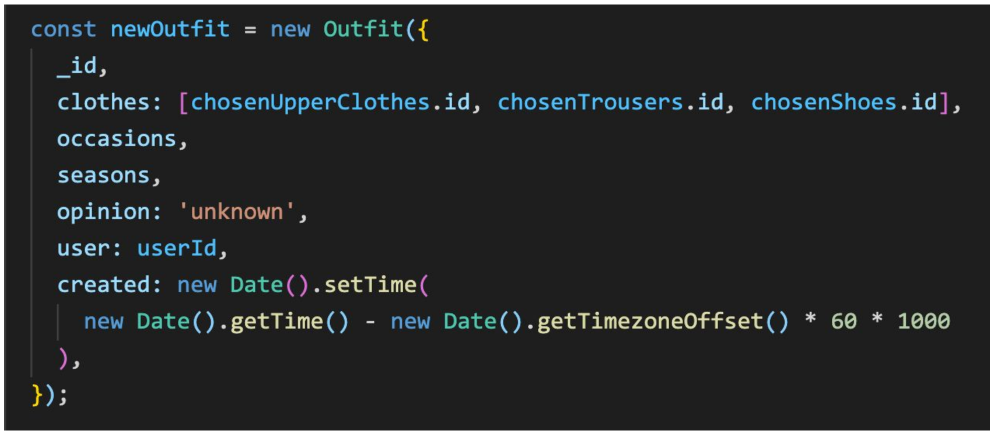
  We are able to identify this problem very quickly as a backend issue considering the precise description provided by the peer team. This issue could lead to confusion between how different API handles timezone and how users might see outfit generation.

  Before



```
const newOutfit = new Outfit({
  _id,
  clothes: [chosenUpperClothes.id, chosenTrousers.id, chosenShoes.id],
  occasions,
  seasons,
  opinion: 'unknown',
  user: userId,
  created: Date.now(),
});
```

  After



```
const newOutfit = new Outfit({
  _id,
  clothes: [chosenUpperClothes.id, chosenTrousers.id, chosenShoes.id],
  occasions,
  seasons,
  opinion: 'unknown',
  user: userId,
  created: new Date().setTime(
    new Date().getTime() - new Date().getTimezoneOffset() * 60 * 1000
  ),
});
```

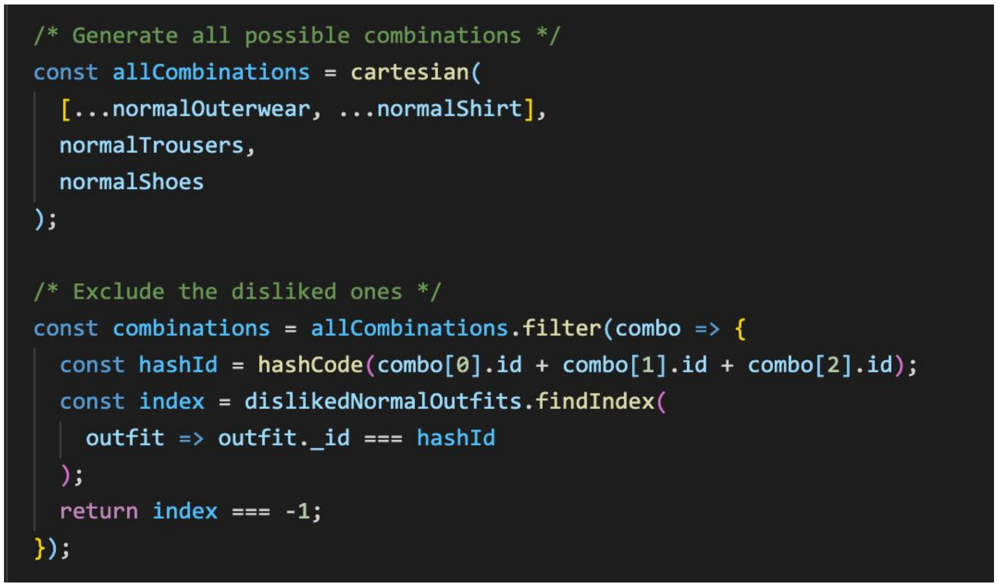
  Explanation

  - Before, we used Date.now() as the outfit created time, but this uses UTC as the default time zone.
  - Now we added the time zone offset before saving an outfit into the database, so that we are able to use current time.
  - Instead of saving UTC, we saved local time and upon returning are more clear

- **Solution of the second problem**

  This problem is related to how we keep track of user liked/disliked outfits and how we should generate outfits.

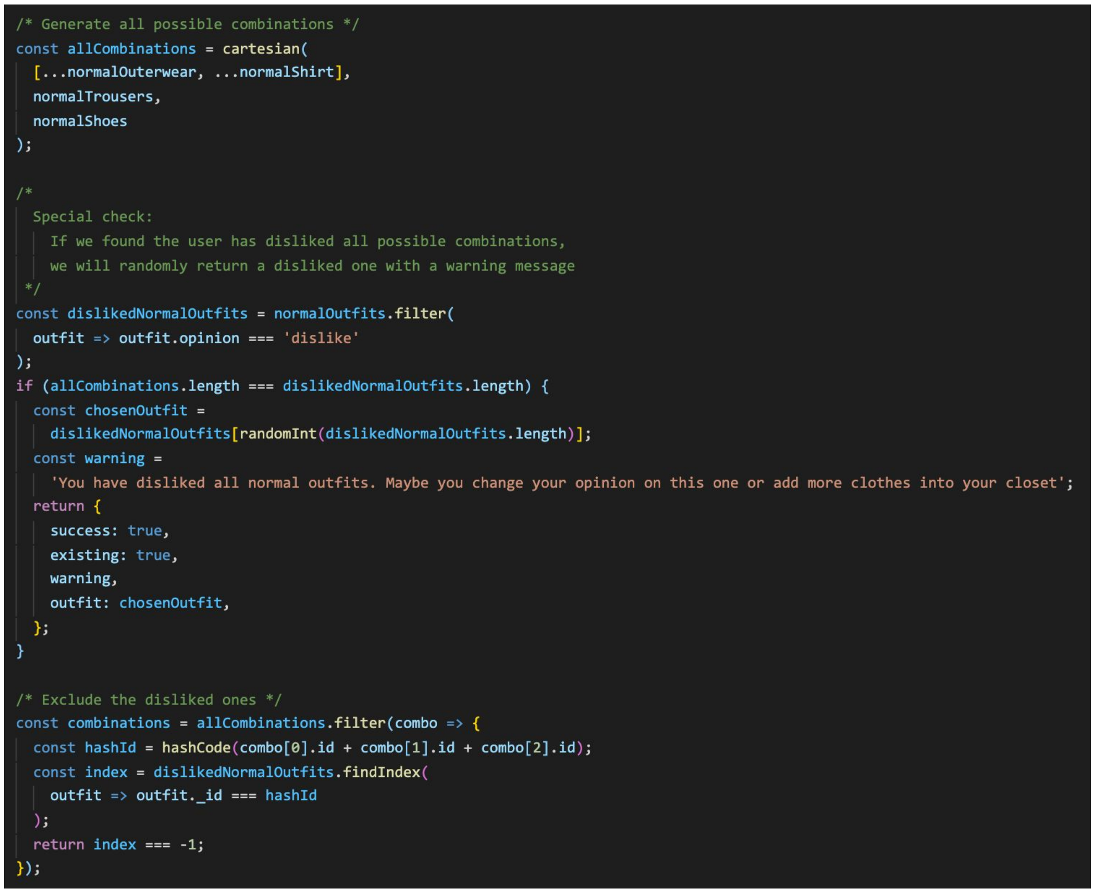
  Before



```
/* Generate all possible combinations */
const allCombinations = cartesian(
  [...normalOuterwear, ...normalShirt],
  normalTrousers,
  normalShoes
);

/* Exclude the disliked ones */
const combinations = allCombinations.filter(combo => {
  const hashId = hashCode(combo[0].id + combo[1].id + combo[2].id);
  const index = dislikedNormalOutfits.findIndex(
    outfit => outfit._id === hashId
  );
  return index === -1;
});
```

After



```
/* Generate all possible combinations */
const allCombinations = cartesian(
  [...normalOuterwear, ...normalShirt],
  normalTrousers,
  normalShoes
);

/*
  Special check:
    If we found the user has disliked all possible combinations,
    we will randomly return a disliked one with a warning message
 */
const dislikedNormalOutfits = normalOutfits.filter(
  outfit => outfit.opinion === 'dislike'
);
if (allCombinations.length === dislikedNormalOutfits.length) {
  const chosenOutfit =
    dislikedNormalOutfits[randomInt(dislikedNormalOutfits.length)];
  const warning =
    'You have disliked all normal outfits. Maybe you change your opinion on this one or add more clothes into your closet';
  return {
    success: true,
    existing: true,
    warning,
    outfit: chosenOutfit,
  };
}

/* Exclude the disliked ones */
const combinations = allCombinations.filter(combo => {
  const hashId = hashCode(combo[0].id + combo[1].id + combo[2].id);
  const index = dislikedNormalOutfits.findIndex(
    outfit => outfit._id === hashId
  );
  return index === -1;
});
```

Explanation

- Before, we directly exclude all user-disliked combinations without considering that the user might dislike all the generated outfits.

- Now, we added a special for this case. If the user dislikes all clothes combinations, our response will contain a randomly selected outfit with a warning message. The message tells the user why we return that outfit and reminds the user to either change their opinion or add more clothes.