Example M5: Testing and Code Review

1. Change History

Change Date	Modified Sections	Rationale
Nothing to show		

2. Back-end Test Specification: APIs

2.1. Locations of Back-end Tests and Instructions to Run Them

2.1.1. Tests

Interface	Describe Group Location, No Mocks	Describe Group Location, With Mo
POST /recommendations/:email	backend/tests/no-mocks/recommendationControllers.test.ts	backend/tests/mocks/recommendationControllers
POST /api/users/location/:email	backend/tests/no-mocks/recommendationControllers.test.ts	backend/tests/mocks/recommendationControllers
GET /chat/:email	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
GET /chat/:email	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
GET /chat/:chatId	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
GET /chat/messages/:chatId/:messageId	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
GET /chat/members/:chatId	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
POST /chat/:email	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
POST /chat/message/:chatId	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
POST /chat/dm/:email	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
PUT /chat/:email	backend/tests/no-mocks/messagingControllers.test.ts	backend/tests/mocks/messagingControllers.mock
GET /User/:email	backend/tests/no-mocks/userControllers.test.ts	
PUT /User/:email	backend/tests/no-mocks/userControllers.test.ts	backend/tests/mocks/userControllers.mock.test
POST /api/v1/auth/google	backend/tests/no-mocks/userControllers.test.ts	backend/tests/mocks/userControllers.mock.test

2.1.2. Commit Hash Where Tests Run

`90f791b3c71aff908a52c9293fd24820235095a6`

2.1.3. Explanation on How to Run the Tests

All Backend Tests are located under `backend/tests`

We assume that you have MongoDB installed and running in your machine.

First clone the reporsitory as follows:

```
git clone https://github.com/TrailBlazersInc/TrailBlazers.git Trailblazers
```

Then cd into `Trailblazers/backend` directory, and create an .env file with the following properties:

```
DB_URI: mongodb://localhost:27017/tests
PORT: 3000
GOOGLE_CLIENT_ID: << "Your GOOGLE CLIENT ID" >>
JWT_SECRET: helloWorld
IS_TESTING: true
```

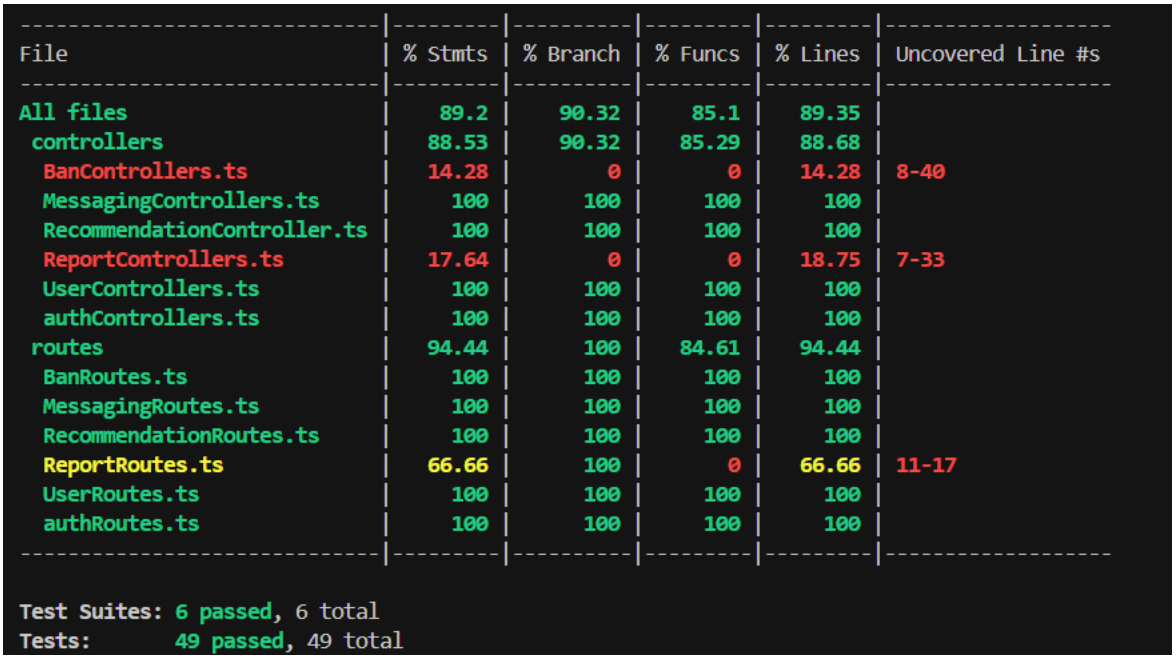
Make sure to replace << Your GOOGLE CLIENT ID >> with your own google OAuth web client ID. Then to start the test run the following commands:

```
npm install
npx ts-jest config:init
npm test # Make sure to add the .env file before running this command
```

2.2. GitHub Actions Configuration Location

`.github/workflows/backendTests.yml`

2.3. Jest Coverage Report Screenshots With Mocks

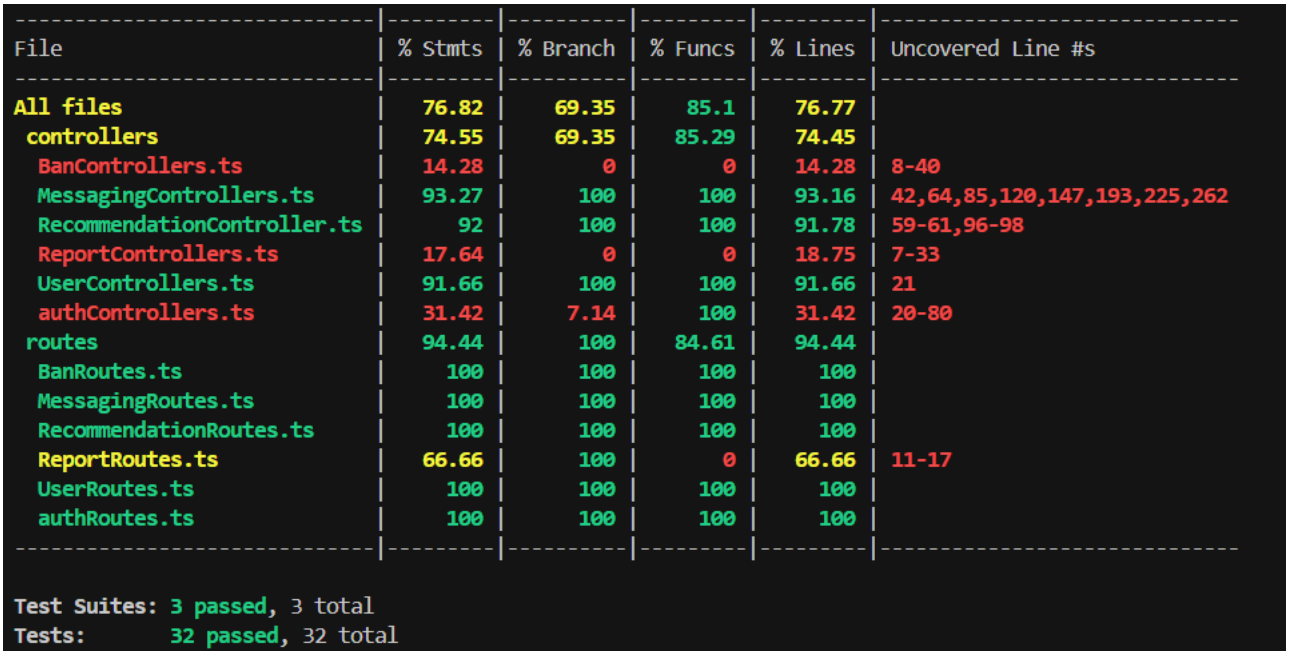


File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	89.2	90.32	85.1	89.35	
controllers	88.53	90.32	85.29	88.68	
BanControllers.ts	14.28	0	0	14.28	8-40
MessagingControllers.ts	100	100	100	100	
RecommendationController.ts	100	100	100	100	
ReportControllers.ts	17.64	0	0	18.75	7-33
UserControllers.ts	100	100	100	100	
authControllers.ts	100	100	100	100	
routes	94.44	100	84.61	94.44	
BanRoutes.ts	100	100	100	100	
MessagingRoutes.ts	100	100	100	100	
RecommendationRoutes.ts	100	100	100	100	
ReportRoutes.ts	66.66	100	0	66.66	11-17
UserRoutes.ts	100	100	100	100	
authRoutes.ts	100	100	100	100	

Test Suites: 6 passed, 6 total
Tests: 49 passed, 49 total

Please note that our team was not required to test BanControllers, BanRoutes, ReportRoutes, nor ReportControllers due to team reduction.

2.4. Jest Coverage Report Screenshots Without Mocks



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	76.82	69.35	85.1	76.77	
controllers	74.55	69.35	85.29	74.45	
BanControllers.ts	14.28	0	0	14.28	8-40
MessagingControllers.ts	93.27	100	100	93.16	42,64,85,120,147,193,225,262
RecommendationController.ts	92	100	100	91.78	59-61,96-98
ReportControllers.ts	17.64	0	0	18.75	7-33
UserControllers.ts	91.66	100	100	91.66	21
authControllers.ts	31.42	7.14	100	31.42	20-80
routes	94.44	100	84.61	94.44	
BanRoutes.ts	100	100	100	100	
MessagingRoutes.ts	100	100	100	100	
RecommendationRoutes.ts	100	100	100	100	
ReportRoutes.ts	66.66	100	0	66.66	11-17
UserRoutes.ts	100	100	100	100	
authRoutes.ts	100	100	100	100	

Test Suites: 3 passed, 3 total
Tests: 32 passed, 32 total

Please note that our team was not required to test BanControllers, BanRoutes, ReportRoutes, nor ReportControllers due to team reduction.

3. Back-end Test Specification: Tests of Non-Functional Requirements

3.1. Test Locations in Git

Non-Functional Requirement	Location in Git
Recommendation Usability	`android_app/app/src/androidTest/java/com/example/cpen321project/RecommendationTest.kt#L109`
Performance (Profile Preferences Updates Response Time)	`android_app/app/src/androidTest/java/com/example/cpen321project/ManageProfileTest.kt`

3.2. Test Verification and Logs

- **Recommendation Usability**
  - **Verification:** This test suite simulates a user to get the top 5 recommendation list. The focus is on parsing user's weight for location, speed and distance to the backend and the matching algorithm will be executed to display the recommendation list within the target response time of 4 seconds. We use Espresso's onView().check(matches(isDisplayed())) to ensure the timer does not stop until the component is well displayed for the user. The test logs let us know if the system is executed within out expected response time.
  - **Log Output**

    ```
    Test 3: Successfully get recommendation in 379ms!
    ```

- **Manage Profile Usability**
  - **Verification:** This test suite simulates updated the pace number entry and saving the changes to your profile. The focus of this test is on parsing updated user information to the backend and updating the database. The test lets us know if the system is executed within our expected response time.
  - **Log Output**

    ```
    Response Time: $responseTime ms
    Test 2: Successfully checked response time for updated profile
    ```

4. Front-end Test Specification

4.1. Location in Git of Front-end Test Suite:

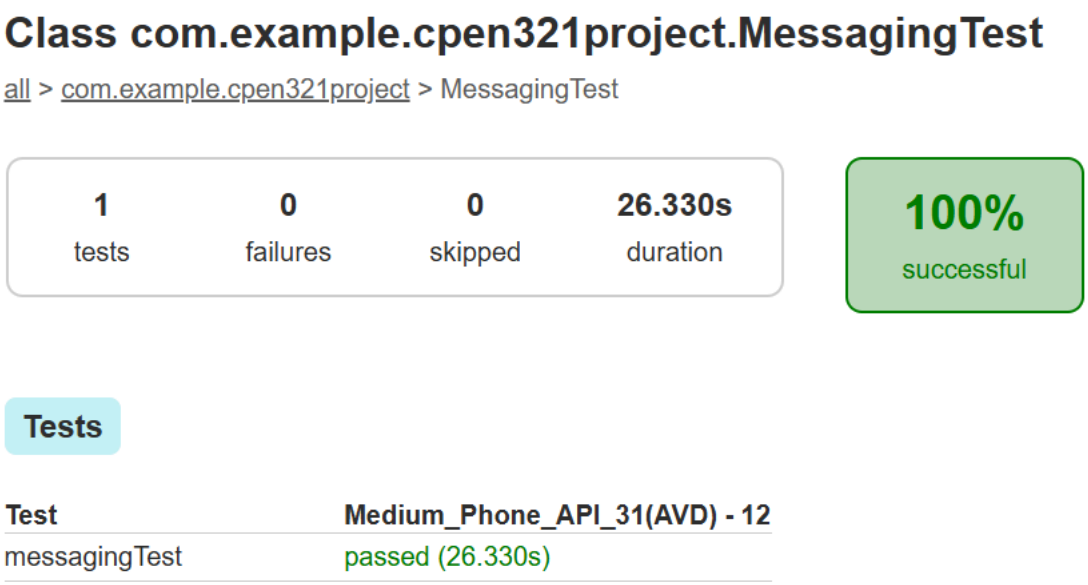
`android_app/src/androidTest/java/com/cpen321project/`

4.2. Tests

- **Use Case: Login**
- **Use Case: Message**
  - **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. User Enters the Chat Overview	click button "My groups in the main page"
2. User Selects Chat	click on the first DM chat available from the overview
3. User enters message into the textbox and clicks on send	Input a "hello, howe are you?" and click send
4. Message is displayed on the chat	Assert that a new message with content hello, howe are you?" is displayed

  - **Test Logs:**

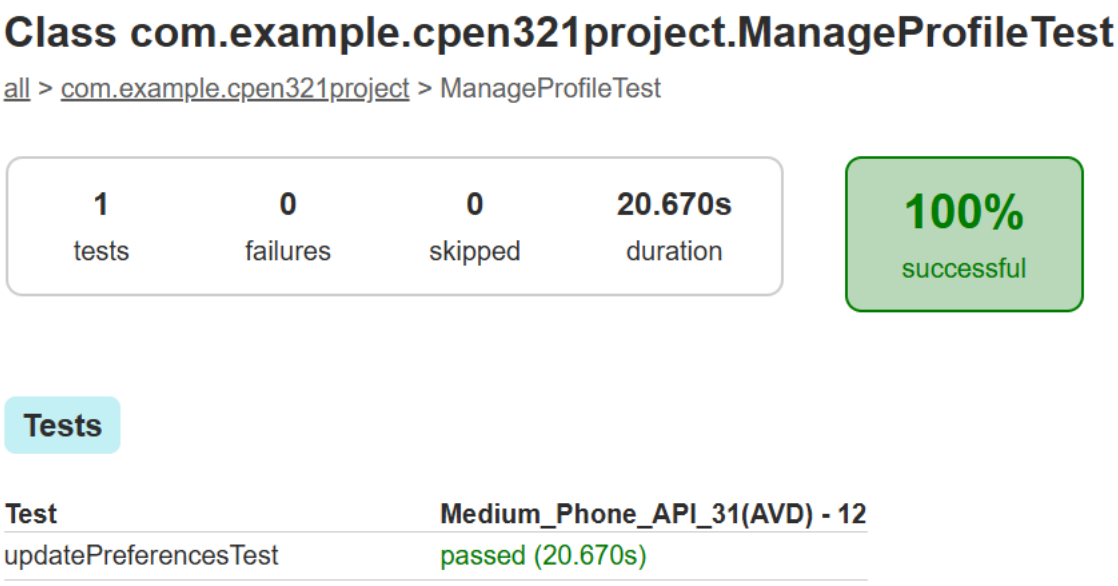


- **Use Case: Manage Profile**
  - **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. User enters the Recommendation Overview.	Click button "Manage Profile" at HomeActivity and it will navigate to ManageProfile.
2a. User inputs invalid number for pace and tries to save changes.	Enter 25.0 into the text field for pace and click "Save Changes" button.
2a1. The app shows an error message prompting the user for a correct value.	Check dialog is opened with text: “Please enter a valid pace”.
2. User inputs valid pace.	Enter a randomly generated number from 1.0 to 20.0 into the text field for pace.
3. User presses the save button.	Click "Save Changes" button.

Scenario Steps	Test Case Steps
4. The app shows a message telling the user the changes were changed successfully.	Check dialog is opened with text: “Please enter a valid pace”.

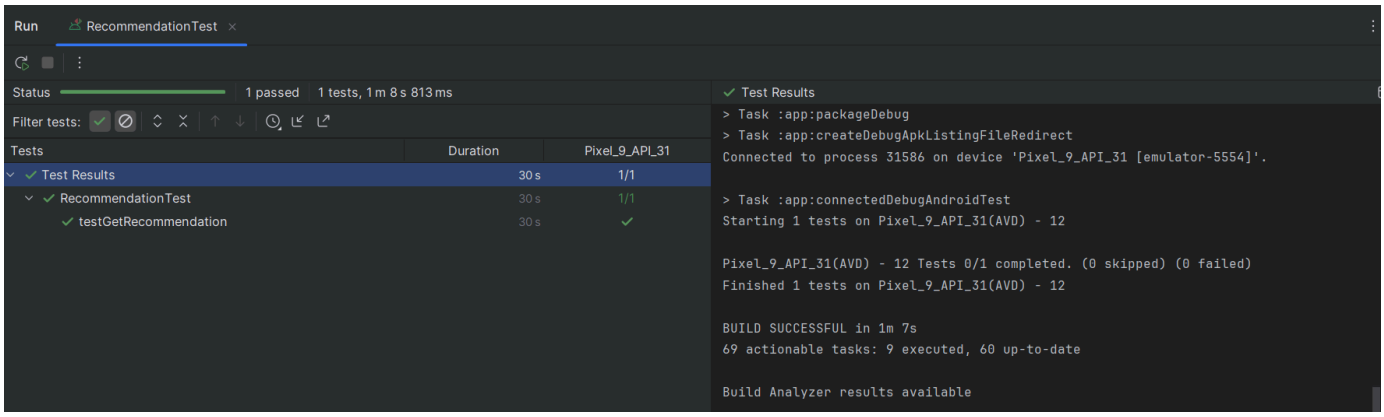
- **Test Logs:**



- **Use Case: Recommendation**
  - **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. User enters the Recommendation Overview.	Click button "Recommendation" at HomeActivity and it will navigates to RecommendationActivity.
2a. User inputs invalid weight for location, speed and distance respectively.	Enter abc, def and ghi respectively.
2. User inputs valid weight for location, speed and distance respectively.	Enter 5, 6 and 7 respectively.
3. User grants location permission.	Click "Grant Location Permission" button.
4. User get recommendation list.	Click "Get Recommendations" button and the top 5 recommendation will be displayed.
5. User can see it's location (and location of joggers if they are nearby)	Click "View on Map" button and it will navigate to MapActivity.
6. User can direct message jogger	Click "Message" button and it will navigate to ChatActivity.

  - **Test Logs:**

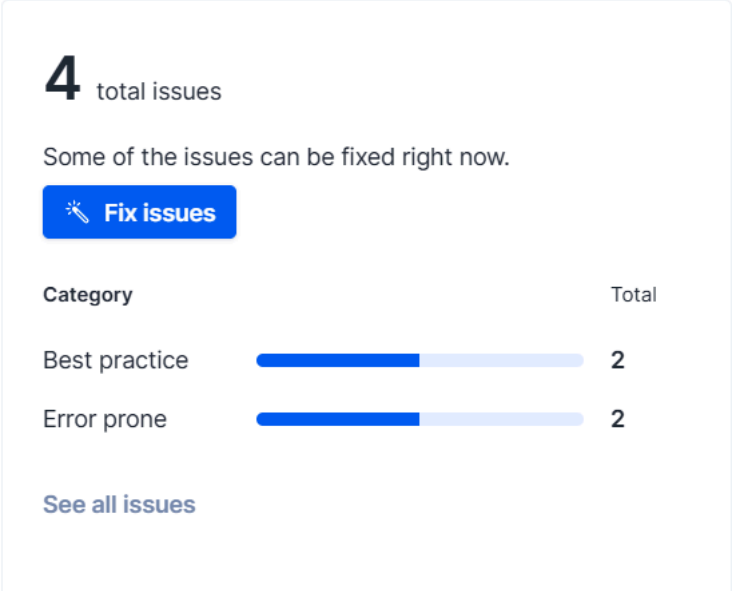


5. Automated Code Review Results

5.1. Commit Hash Where Codacy Ran

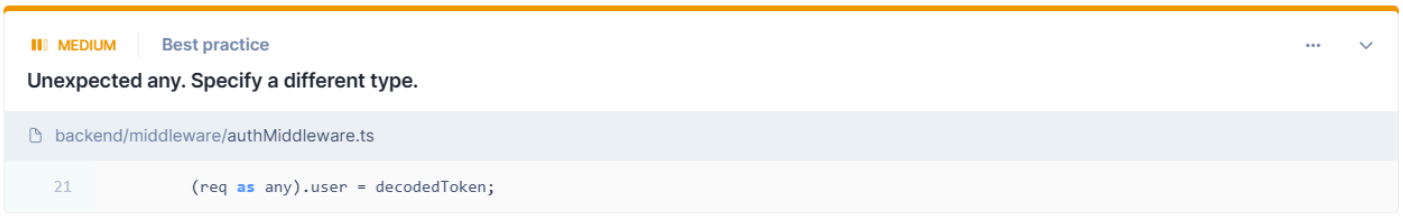
`90f791b3c71aff908a52c9293fd24820235095a6`

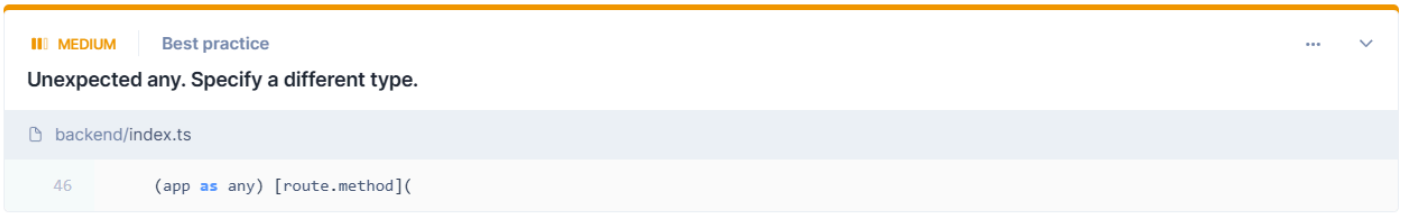
5.2. Unfixed Issues per Codacy Category



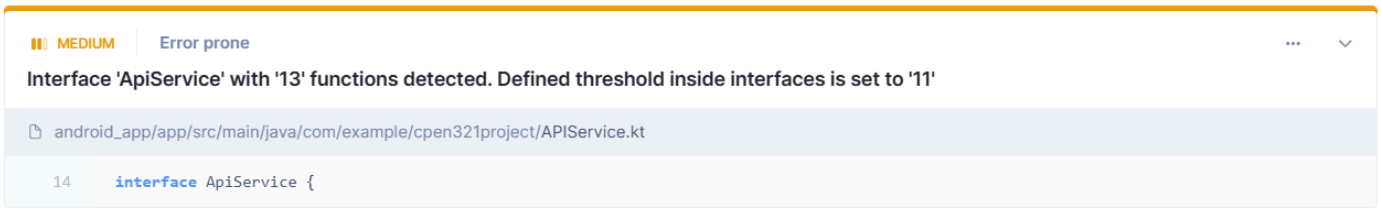
5.3. Unfixed Issues per Codacy Code Pattern

1. **@typescript eslint:** No explicit any

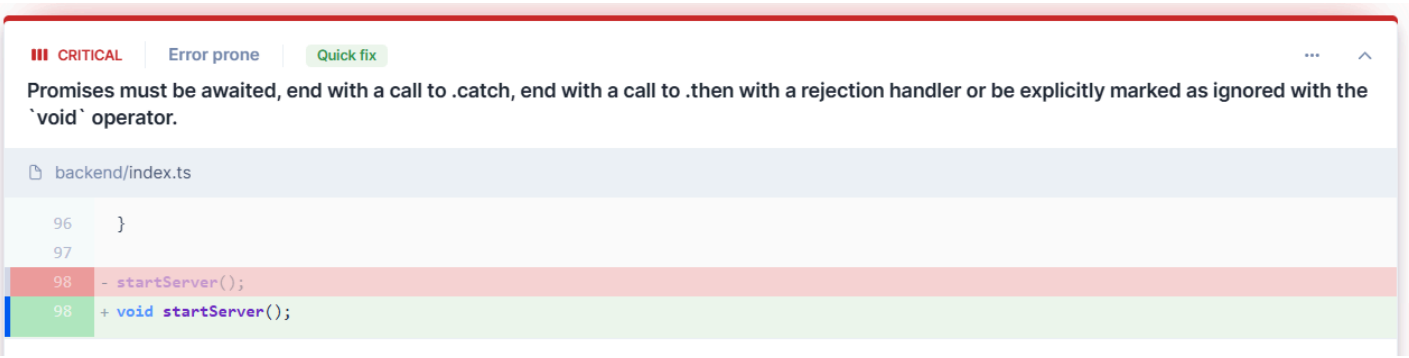




2. **Too many functions inside a/an file/class/object/interface always indicate a violation of the single responsibility principle. Maybe the file/class/object/interface wants to manage too many things at once.**



3. **Promises must be awaited, end with a call to .catch, end with a call to .then with a rejection handler or be explicitly marked as ignored with the `void` operator.**



5.4. Justifications for Unfixed Issues

1. **@typescript eslint:** No explicit any

- **Location in Git:** `backend/index.ts`
- **Justification:** Since we are using exclusively our own routes and do not need to fetch it from some outter source and we are doing this only to set up the server, it is not necessary to heavily type the express application.
- **Location in Git:** `backend/middleware/authMiddleware.ts`
- **Justification:** Since this is middleware we want the response to be fast so it doesn't slow down the response time of the requests. By using any we are prioritizing speed and since we are not using strict mode in our typescript config it is unnecessary to add type safety to any.

2. **Too many functions inside a/an file/class/object/interface always indicate a violation of the single responsibility principle. Maybe the file/class/object/interface wants to manage too many things at once.**
   - **Location in Git:** `android_app/app/src/main/java/com/example/cpen321project/APIService.kt`
   - **Justification:** ApiService is used for all of our API request functions, we can not reduce the amount of API request functions we have for it.
3. **Promises must be awaited, end with a call to .catch, end with a call to .then with a rejection handler or be explicitly marked as ignored with the `void` operator.**
   - **Location in Git:** `backend/index.ts`
   - **Justification:** Since the try catch is already handled inside the function, and it is only used to start the server, there is no need to specify void on startServer().