Summary

Progress since M8

On the backend side, we **improved complex logic**. We also enhanced our availability to **work with third party APIs**. And, we included more unit tests to an overall **80.04%** statements, **72.79%** branches, **73%** functions, **81%** lines coverage.

On the frontend side, we **continued to add features** related to complex logics and improve existing features**.** We also **added more UI tests** for the add clothes and get outfit use cases. And, we added tests for non-functional requirements.

We also **integrated with Codecov** to generate test coverage reports both on our pull requests and overall master branch so that we are more aware of our tests.

Plans until M10

On project level, we will complete all features and enhancements, and try to reach for overall 100% test coverages.

On the backend side, we will continue to **improve our complex logic** and add additional tests for our complex logics. We hope to also **improve the performance** by implementing a caching system if time allows.

On the frontend side, we will **allow users to get multiple outfits**, and **enhance the UI for users to choose their own outfits.** We will also **fix the bugs** we found in the user acceptance test**.**

Major decisions and changes

No change

Member contributions

- **Steven Yang**:
  Integration with Codecov for better test coverage viewing on pull requests
  Adding unit tests
- **John Li**:
  Improved complex logic with third party APIs
  Adding unit tests
- **Zhuoyi Li**:
  Improved get/post weather and **outfit opinions data** from/to the server
  Adding UI tests and non-functional requirement test
- **Jingyang Sun**:
  Implemented dynamically **add/delete clothes** and **edit clothes image**
  Added UI test and non-functional requirement test

Automated back-end testing

Some clarifications:

We replaced the User module (written in M7) with the Outfit Module because we found the Outfit module is much larger than the User module.

A table summarizing the number of back-end tests per the tested modules and the number of integration tests, as specified in M7

Unit tests:

Each module had multiple code files related to it, mostly a controller file and a service file along with other smaller services that the module uses. We test individually these smaller modules, and then group them together with the main controller for overall testing. We have **53** unit tests and **5** integration tests for the two largest modules in our project.

Clothes Module	21 unit tests total
User clothes tests (controller/clothes.test.js)	15 unit tests
User clothing image test (controller/image.test.js)	6 unit tests

Outfit Module	32 unit tests total
User outfit controller tests (controller/outfits.test.js)	8 unit tests
User outfit service tests (service/outfits-service.test.js)	3 unit tests
Weather service tests (service/weather.test.js)	4 unit tests
Calendar controller tests (controller/calendar.test.js)	5 unit tests
Calendar service tests (service/calendar.test.js)	4 unit tests
Hash utility function tests (utils/hash.test.js)	3 unit tests
Time helper function tests (utils/time-helper.test.js)	5 unit tests

Integration tests:

Clothes Module

User clothing (add clothes, delete clothes)	2 tests

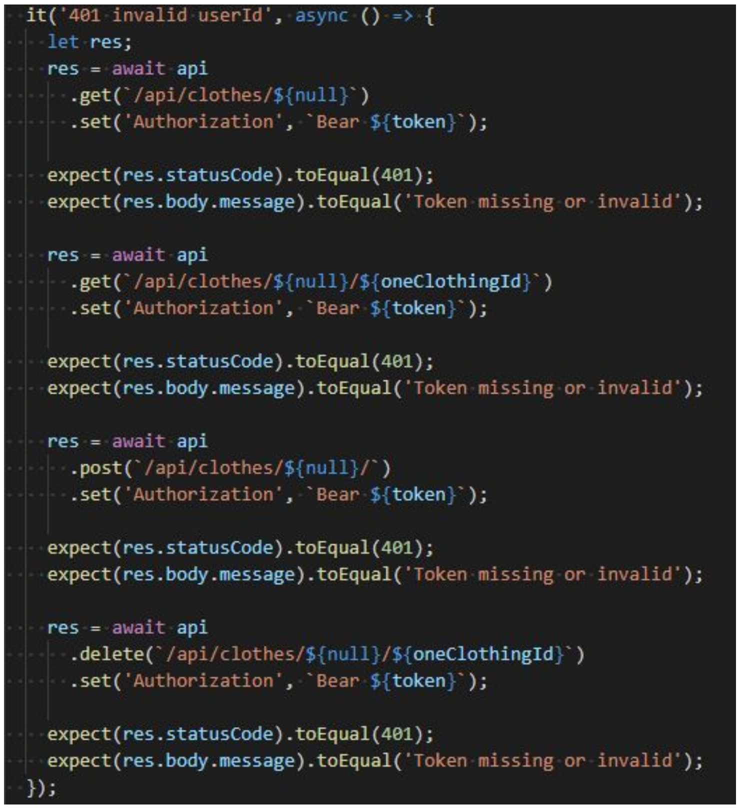
User clothing images (add clothes image, get clothes image)	2 tests

Outfit Module

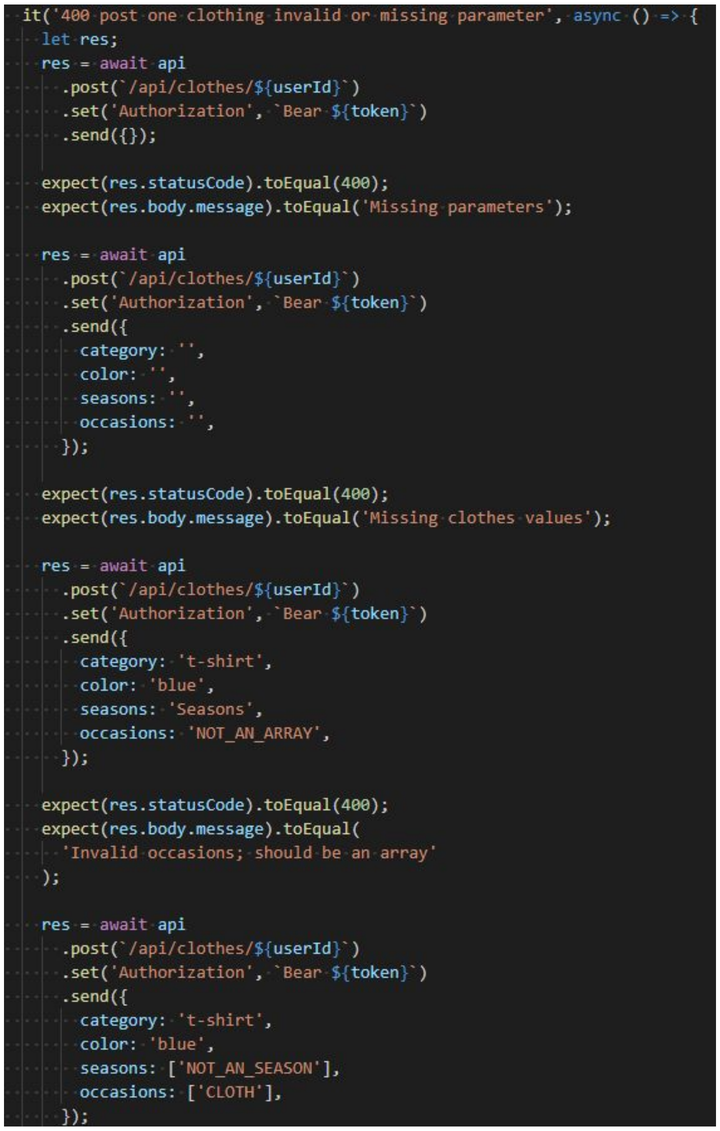
Weather service (get weather)	1 test

A screenshot with success/fail condition of each test specified in M7
Some tests are nested within another test so they might not be very obvious in separate screenshots. Some tests are also repeated testing to ensure validity after some operations.

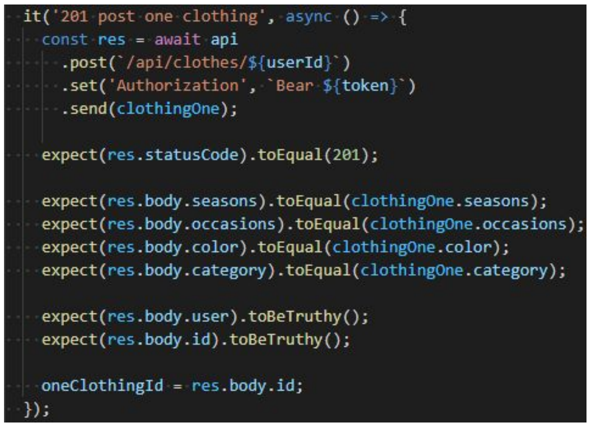
Clothes Module (21 unit tests + 4 integration tests)

User clothes tests (controller/clothes.test.js) 	15 unit tests

```
it('401 invalid userId', async () => {
  let res;
  res = await api
    .get(`/api/clothes/${null}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');

  res = await api
    .get(`/api/clothes/${null}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');

  res = await api
    .post(`/api/clothes/${null}/`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');

  res = await api
    .delete(`/api/clothes/${null}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');
});
```

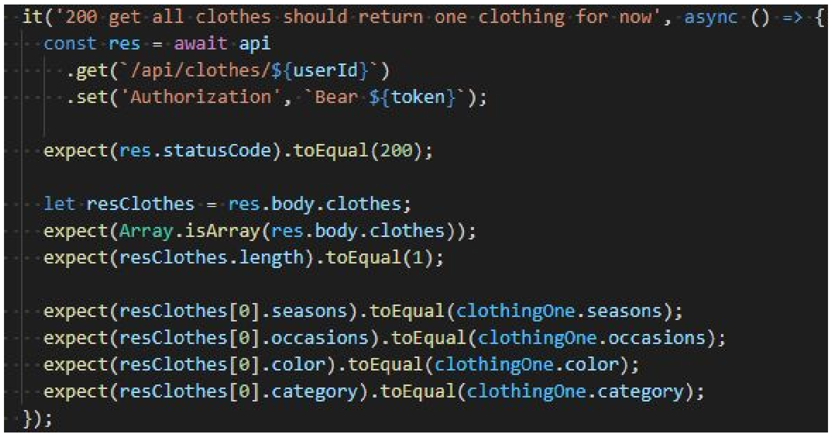


```javascript
it('400 post one clothing invalid or missing parameter', async () => {
  let res;
  res = await api
    .post(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`)
    .send({});

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing parameters');

  res = await api
    .post(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: '',
      color: '',
      seasons: '',
      occasions: '',
    });

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing clothes values');

  res = await api
    .post(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: 't-shirt',
      color: 'blue',
      seasons: 'Seasons',
      occasions: 'NOT_AN_ARRAY',
    });

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual(
    'Invalid occasions; should be an array'
  );

  res = await api
    .post(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: 't-shirt',
      color: 'blue',
      seasons: ['NOT_AN_SEASON'],
      occasions: ['CLOTH'],
    });
```

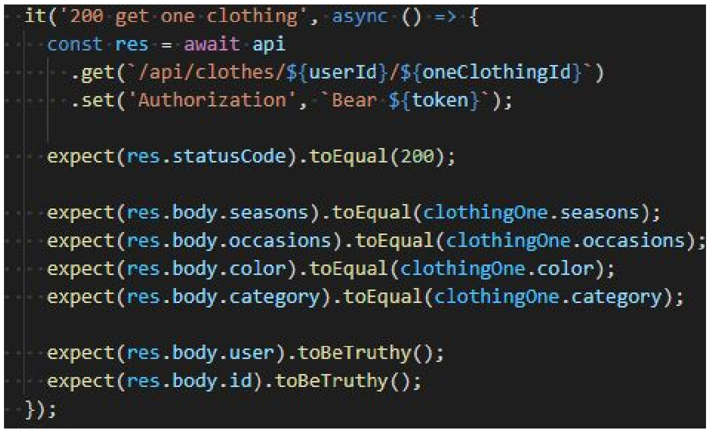


```javascript
it('201 post one clothing', async () => {
  const res = await api
    .post(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`)
    .send(clothingOne);

  expect(res.statusCode).toEqual(201);

  expect(res.body.seasons).toEqual(clothingOne.seasons);
  expect(res.body.occasions).toEqual(clothingOne.occasions);
  expect(res.body.color).toEqual(clothingOne.color);
  expect(res.body.category).toEqual(clothingOne.category);

  expect(res.body.user).toBeTruthy();
  expect(res.body.id).toBeTruthy();

  oneClothingId = res.body.id;
});
```

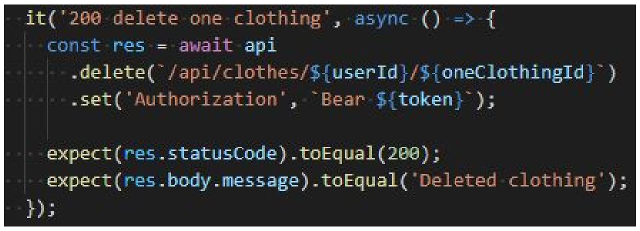
Post a clothes is also an integration test since we do not have any mock in clothes controller tests



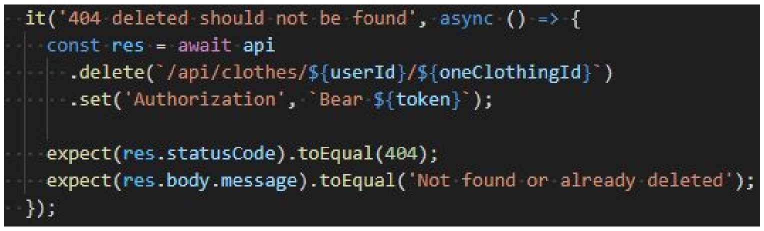
```
it('200 get all clothes should return one clothing for now', async () => {
  const res = await api
    .get(`/api/clothes/${userId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);

  let resClothes = res.body.clothes;
  expect(Array.isArray(res.body.clothes));
  expect(resClothes.length).toEqual(1);

  expect(resClothes[0].seasons).toEqual(clothingOne.seasons);
  expect(resClothes[0].occasions).toEqual(clothingOne.occasions);
  expect(resClothes[0].color).toEqual(clothingOne.color);
  expect(resClothes[0].category).toEqual(clothingOne.category);
});
```



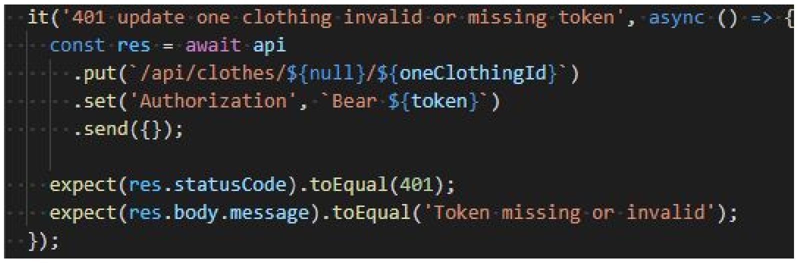
```
it('200 get one clothing', async () => {
  const res = await api
    .get(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);

  expect(res.body.seasons).toEqual(clothingOne.seasons);
  expect(res.body.occasions).toEqual(clothingOne.occasions);
  expect(res.body.color).toEqual(clothingOne.color);
  expect(res.body.category).toEqual(clothingOne.category);

  expect(res.body.user).toBeTruthy();
  expect(res.body.id).toBeTruthy();
});
```



```
it('200 delete one clothing', async () => {
  const res = await api
    .delete(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.message).toEqual('Deleted clothing');
});
```

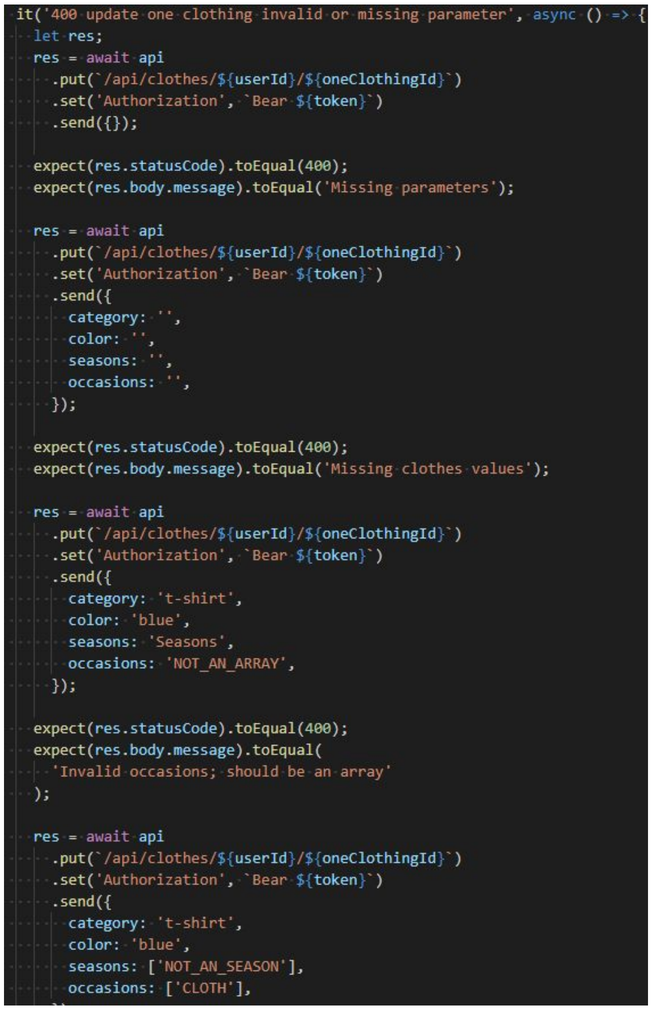


```
it('404 deleted should not be found', async () => {
  const res = await api
    .delete(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(404);
  expect(res.body.message).toEqual('Not found or already deleted');
});
```

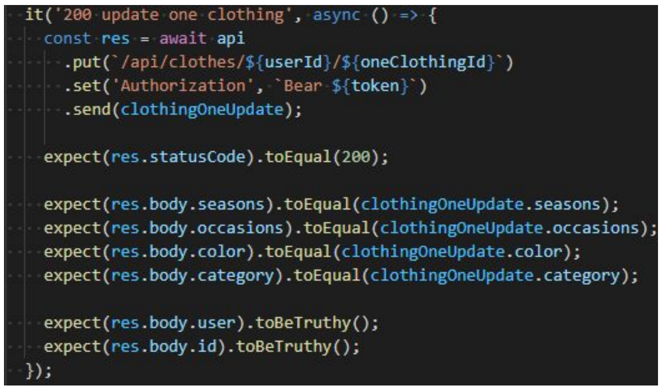


```
it('401 update one clothing invalid or missing token', async () => {
  const res = await api
    .put(`/api/clothes/${null}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send({});

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');
});
```

Delete a clothes is also an integration test since we do not have any mock in clothes controller tests

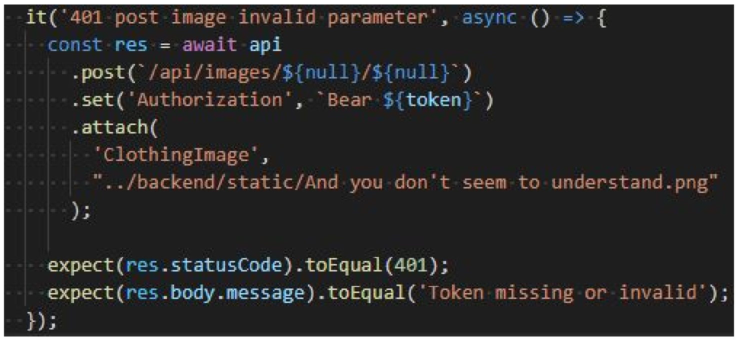


```javascript
it('400 update one clothing invalid or missing parameter', async () => {
  let res;
  res = await api
    .put(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send({});

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing parameters');

  res = await api
    .put(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: '',
      color: '',
      seasons: '',
      occasions: '',
    });

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing clothes values');

  res = await api
    .put(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: 't-shirt',
      color: 'blue',
      seasons: 'Seasons',
      occasions: 'NOT_AN_ARRAY',
    });

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual(
    'Invalid occasions; should be an array'
  );

  res = await api
    .put(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      category: 't-shirt',
      color: 'blue',
      seasons: ['NOT_AN_SEASON'],
      occasions: ['CLOTH'],
```

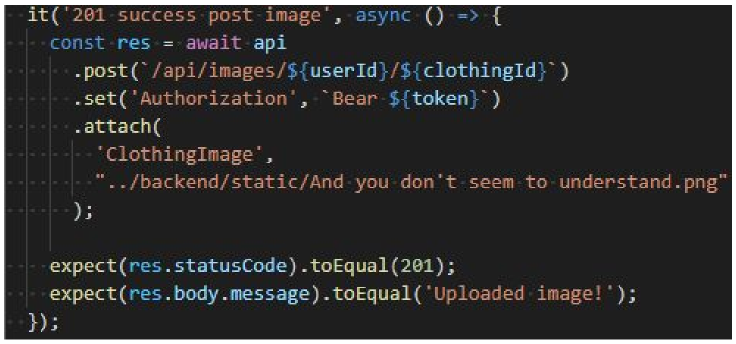


```javascript
it('200 update one clothing', async () => {
  const res = await api
    .put(`/api/clothes/${userId}/${oneClothingId}`)
    .set('Authorization', `Bear ${token}`)
    .send(clothingOneUpdate);

  expect(res.statusCode).toEqual(200);

  expect(res.body.seasons).toEqual(clothingOneUpdate.seasons);
  expect(res.body.occasions).toEqual(clothingOneUpdate.occasions);
  expect(res.body.color).toEqual(clothingOneUpdate.color);
  expect(res.body.category).toEqual(clothingOneUpdate.category);

  expect(res.body.user).toBeTruthy();
  expect(res.body.id).toBeTruthy();
});
```

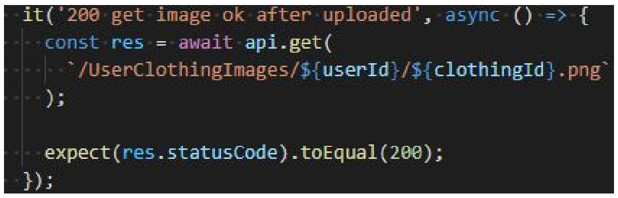
User clothing image test (controller/image.test.js)	6 unit tests



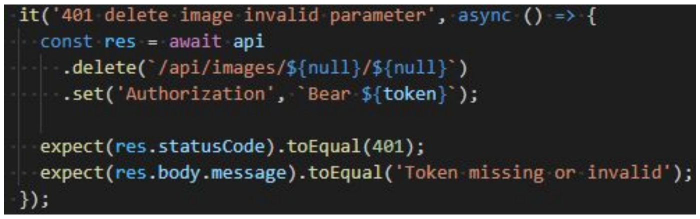
```javascript
it('401 post image invalid parameter', async () => {
  const res = await api
    .post(`/api/images/${null}/${null}`)
    .set('Authorization', `Bear ${token}`)
    .attach(
      'ClothingImage',
      "../backend/static/And you don't seem to understand.png"
    );

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');
});
```



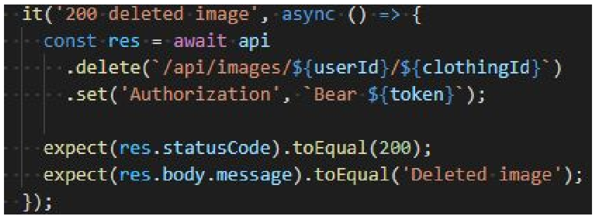
```javascript
it('201 success post image', async () => {
  const res = await api
    .post(`/api/images/${userId}/${clothingId}`)
    .set('Authorization', `Bear ${token}`)
    .attach(
      'ClothingImage',
      "../backend/static/And you don't seem to understand.png"
    );

  expect(res.statusCode).toEqual(201);
  expect(res.body.message).toEqual('Uploaded image!');
});
```



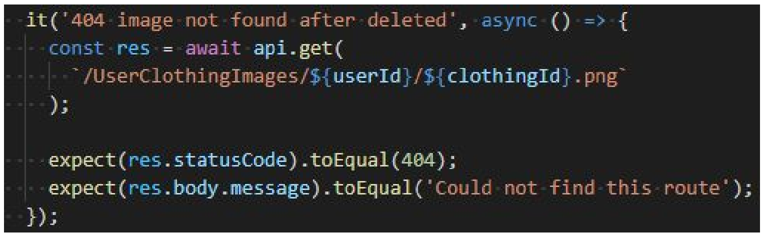
```javascript
it('200 get image ok after uploaded', async () => {
  const res = await api.get(
    `/UserClothingImages/${userId}/${clothingId}.png`
  );

  expect(res.statusCode).toEqual(200);
});
```



```javascript
it('401 delete image invalid parameter', async () => {
  const res = await api
    .delete(`/api/images/${null}/${null}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(401);
  expect(res.body.message).toEqual('Token missing or invalid');
});
```



```javascript
it('200 deleted image', async () => {
  const res = await api
    .delete(`/api/images/${userId}/${clothingId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.message).toEqual('Deleted image');
});
```

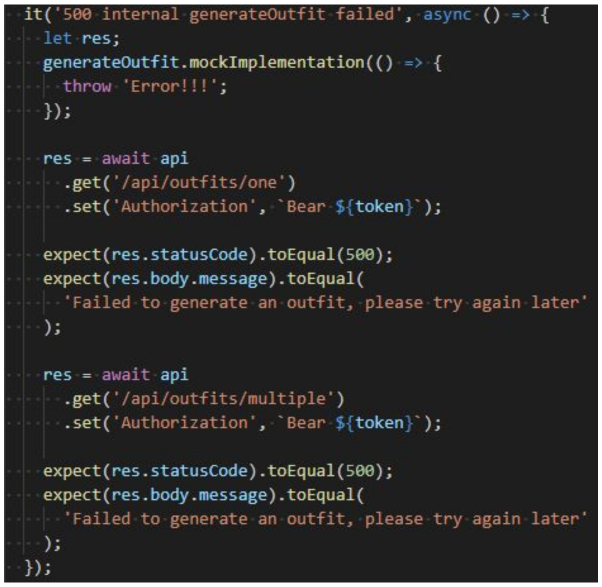
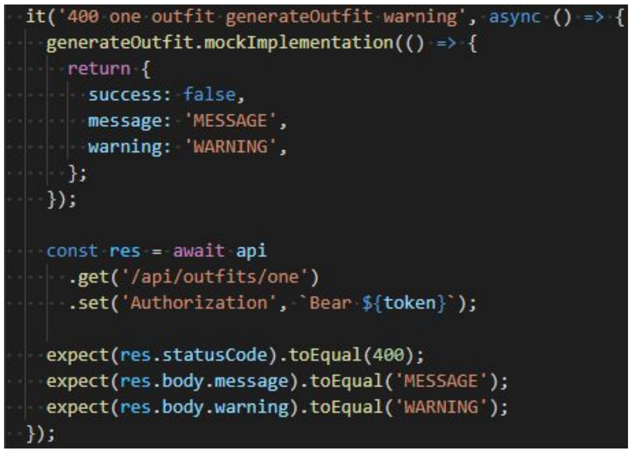
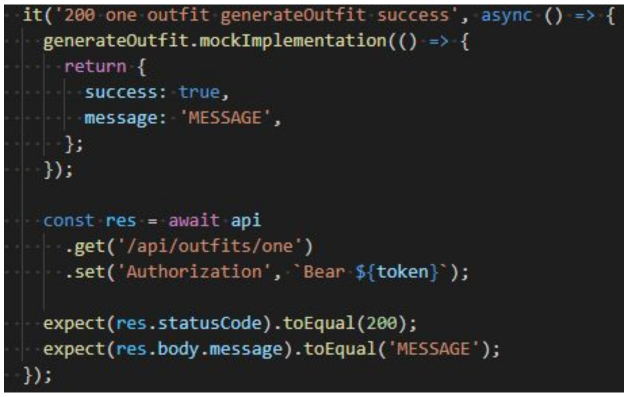


```javascript
it('404 image not found after deleted', async () => {
  const res = await api.get(
    `/UserClothingImages/${userId}/${clothingId}.png`
  );

  expect(res.statusCode).toEqual(404);
  expect(res.body.message).toEqual('Could not find this route');
});
```

Post an image is also an integration test since we do not have any mock in image controller tests

Get an image is also an integration test since we do not have any mock in image controller tests

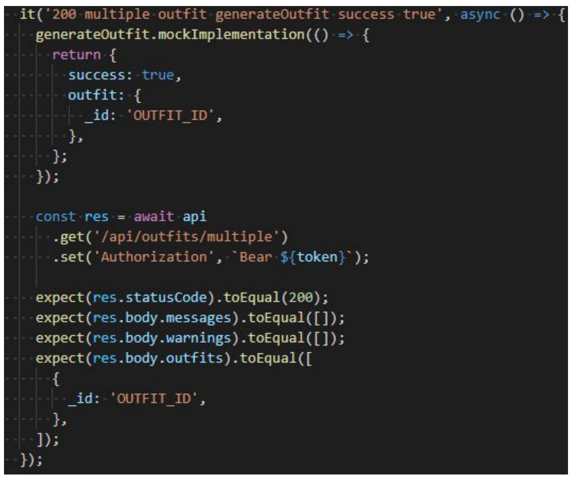
Outfit Module (32 unit tests + 2 integration tests)

User outfit controller tests (controller/outfits.test.js)   	8 unit tests

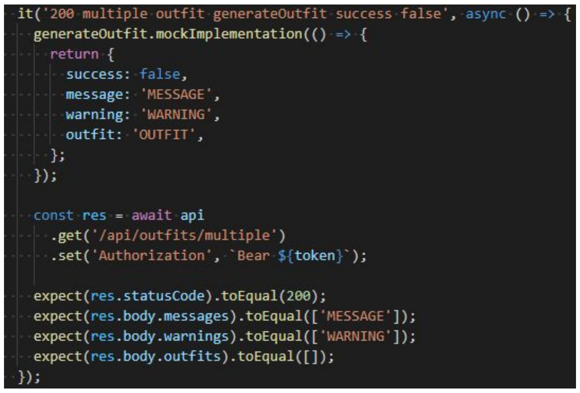
```
it('500 internal generateOutfit failed', async () => {
  let res;
  generateOutfit.mockImplementation(() => {
    throw 'Error!!!';
  });

  res = await api
    .get('/api/outfits/one')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to generate an outfit, please try again later'
  );

  res = await api
    .get('/api/outfits/multiple')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to generate an outfit, please try again later'
  );
});
```

```
it('400 one outfit generateOutfit warning', async () => {
  generateOutfit.mockImplementation(() => {
    return {
      success: false,
      message: 'MESSAGE',
      warning: 'WARNING',
    };
  });

  const res = await api
    .get('/api/outfits/one')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('MESSAGE');
  expect(res.body.warning).toEqual('WARNING');
});
```

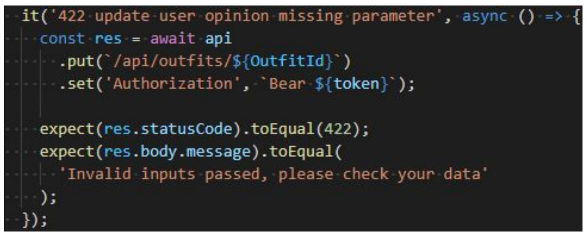
```
it('200 one outfit generateOutfit success', async () => {
  generateOutfit.mockImplementation(() => {
    return {
      success: true,
      message: 'MESSAGE',
    };
  });

  const res = await api
    .get('/api/outfits/one')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.message).toEqual('MESSAGE');
});
```



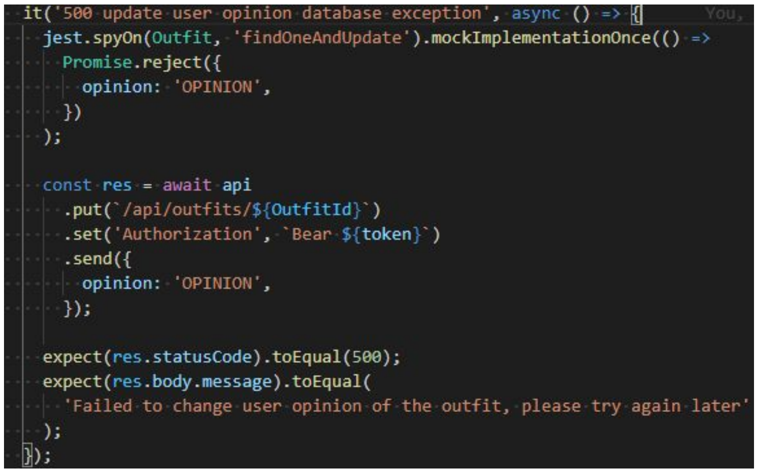
```javascript
it('200 multiple outfit generateOutfit success true', async () => {
  generateOutfit.mockImplementation(() => {
    return {
      success: true,
      outfit: {
        _id: 'OUTFIT_ID',
      },
    };
  });

  const res = await api
    .get('/api/outfits/multiple')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.messages).toEqual([]);
  expect(res.body.warnings).toEqual([]);
  expect(res.body.outfits).toEqual([
    {
      _id: 'OUTFIT_ID',
    },
  ]);
});
```



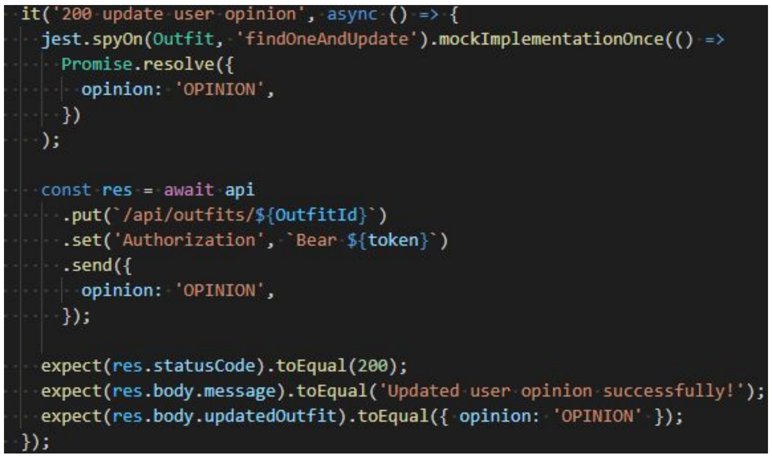
```javascript
it('200 multiple outfit generateOutfit success false', async () => {
  generateOutfit.mockImplementation(() => {
    return {
      success: false,
      message: 'MESSAGE',
      warning: 'WARNING',
      outfit: 'OUTFIT',
    };
  });

  const res = await api
    .get('/api/outfits/multiple')
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.messages).toEqual(['MESSAGE']);
  expect(res.body.warnings).toEqual(['WARNING']);
  expect(res.body.outfits).toEqual([]);
});
```



```javascript
it('422 update user opinion missing parameter', async () => {
  const res = await api
    .put(`/api/outfits/${OutfitId}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(422);
  expect(res.body.message).toEqual(
    'Invalid inputs passed, please check your data'
  );
});
```

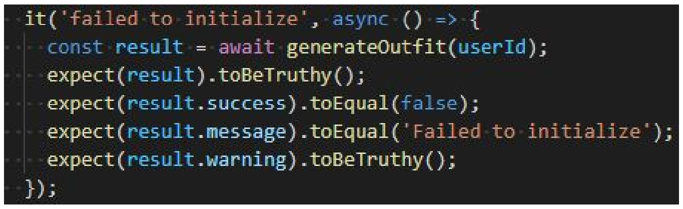


```javascript
it('500 update user opinion database exception', async () => {
  jest.spyOn(Outfit, 'findOneAndUpdate').mockImplementationOnce(() =>
    Promise.reject({
      opinion: 'OPINION',
    })
  );

  const res = await api
    .put(`/api/outfits/${OutfitId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      opinion: 'OPINION',
    });

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to change user opinion of the outfit, please try again later'
  );
});
```

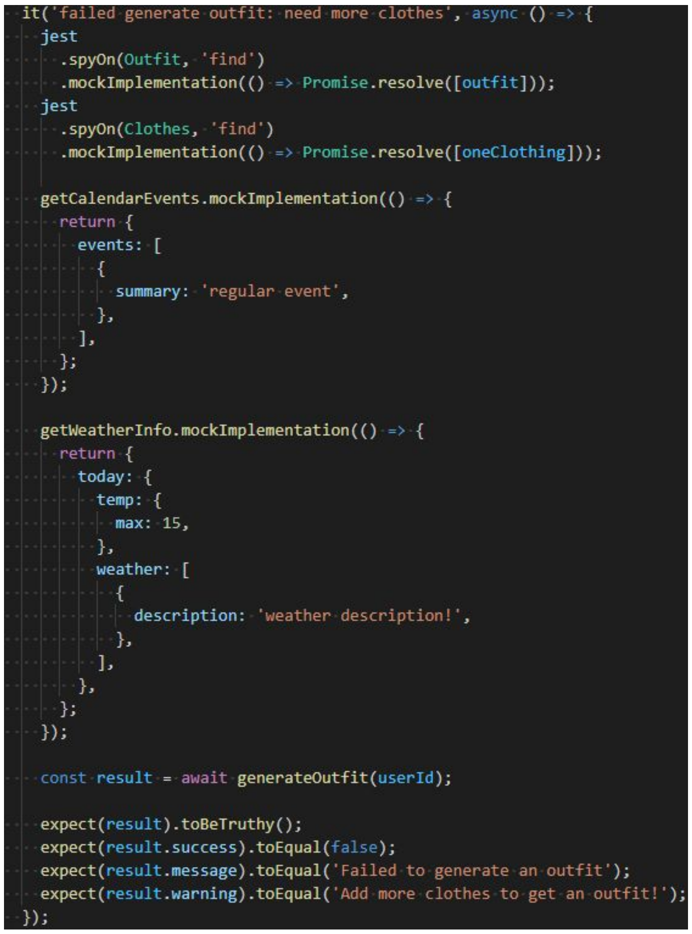


```
it('200 update user opinion', async () => {
  jest.spyOn(Outfit, 'findOneAndUpdate').mockImplementationOnce(() =>
    Promise.resolve({
      opinion: 'OPINION',
    })
  );

  const res = await api
    .put(`/api/outfits/${OutfitId}`)
    .set('Authorization', `Bear ${token}`)
    .send({
      opinion: 'OPINION',
    });

  expect(res.statusCode).toEqual(200);
  expect(res.body.message).toEqual('Updated user opinion successfully!');
  expect(res.body.updatedOutfit).toEqual({ opinion: 'OPINION' });
});
```

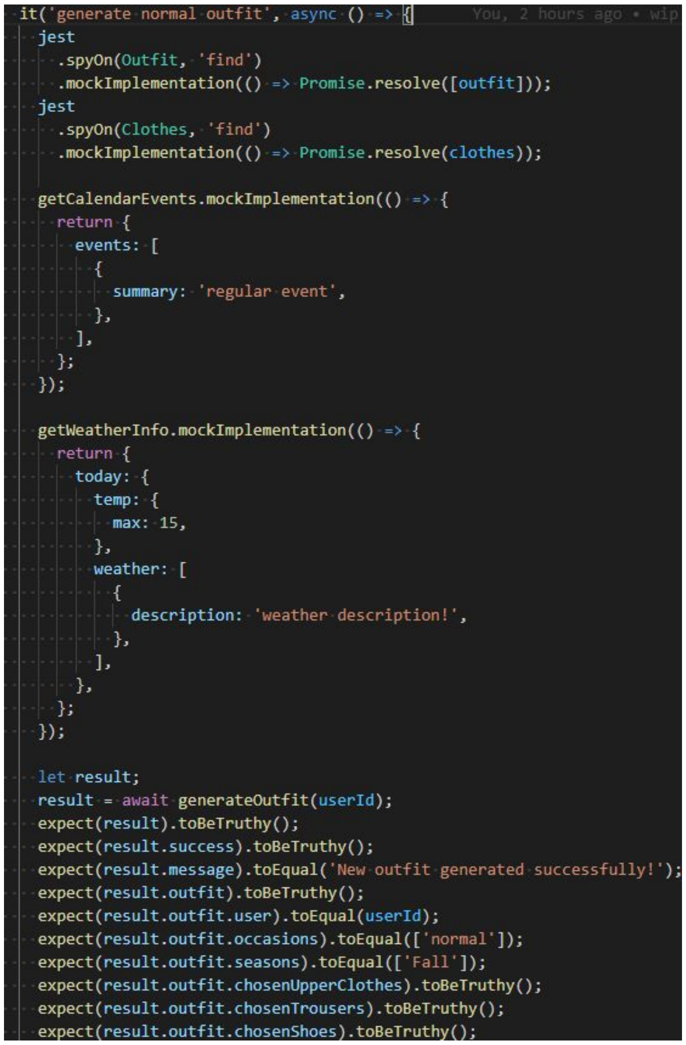
User outfit service tests (service/outfits-service.test.js)	3 unit tests



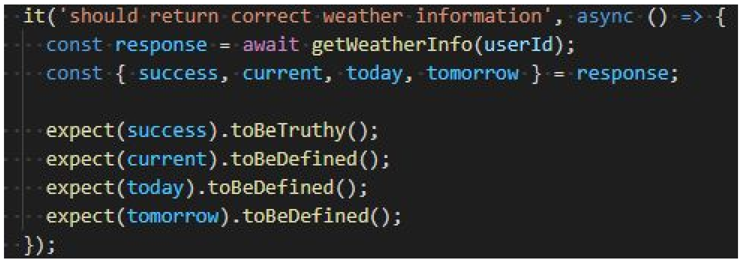
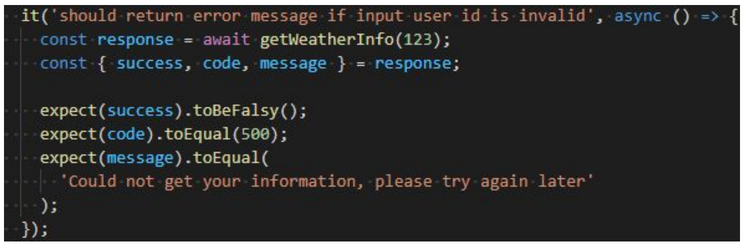
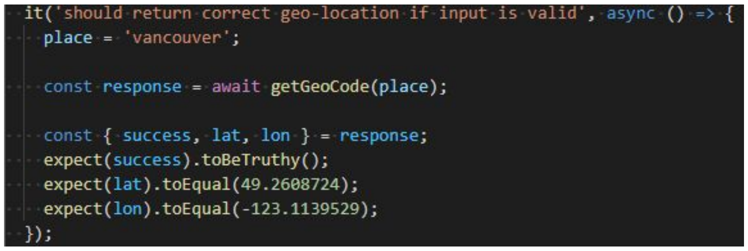
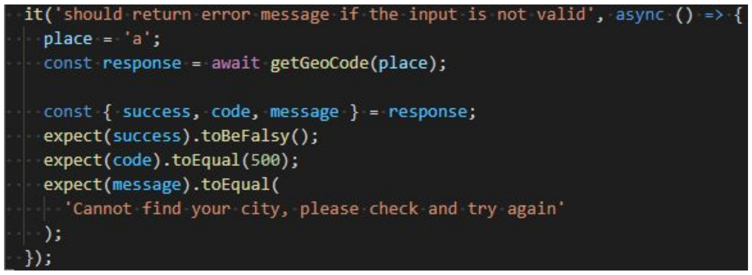
```
it('failed to initialize', async () => {
  const result = await generateOutfit(userId);
  expect(result).toBeTruthy();
  expect(result.success).toEqual(false);
  expect(result.message).toEqual('Failed to initialize');
  expect(result.warning).toBeTruthy();
});
```



```
it('failed generate outfit: need more clothes', async () => {
  jest
    .spyOn(Outfit, 'find')
    .mockImplementation(() => Promise.resolve([outfit]));
  jest
    .spyOn(Clothes, 'find')
    .mockImplementation(() => Promise.resolve([oneClothing]));

  getCalendarEvents.mockImplementation(() => {
    return {
      events: [
        {
          summary: 'regular event',
        },
      ],
    };
  });

  getWeatherInfo.mockImplementation(() => {
    return {
      today: {
        temp: {
          max: 15,
        },
        weather: [
          {
            description: 'weather description!',
          },
        ],
      },
    };
  });

  const result = await generateOutfit(userId);

  expect(result).toBeTruthy();
  expect(result.success).toEqual(false);
  expect(result.message).toEqual('Failed to generate an outfit');
  expect(result.warning).toEqual('Add more clothes to get an outfit!');
});
```



```
it('generate normal outfit', async () => {      You, 2 hours ago • wip
  jest
    .spyOn(Outfit, 'find')
    .mockImplementation(() => Promise.resolve([outfit]));
  jest
    .spyOn(Clothes, 'find')
    .mockImplementation(() => Promise.resolve(clothes));

  getCalendarEvents.mockImplementation(() => {
    return {
      events: [
        {
          summary: 'regular event',
        },
      ],
    };
  });

  getWeatherInfo.mockImplementation(() => {
    return {
      today: {
        temp: {
          max: 15,
        },
        weather: [
          {
            description: 'weather description!',
          },
        ],
      },
    };
  });

  let result;
  result = await generateOutfit(userId);
  expect(result).toBeTruthy();
  expect(result.success).toBeTruthy();
  expect(result.message).toEqual('New outfit generated successfully!');
  expect(result.outfit).toBeTruthy();
  expect(result.outfit.user).toEqual(userId);
  expect(result.outfit.occasions).toEqual(['normal']);
  expect(result.outfit.seasons).toEqual(['Fall']);
  expect(result.outfit.chosenUpperClothes).toBeTruthy();
  expect(result.outfit.chosenTrousers).toBeTruthy();
  expect(result.outfit.chosenShoes).toBeTruthy();
```

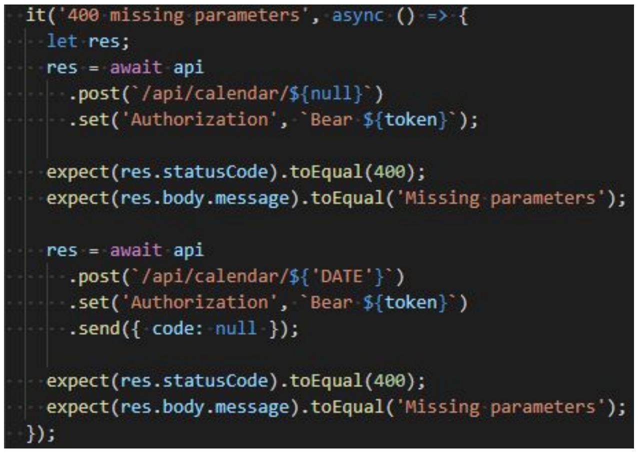
Weather service tests (service/weather.test.js)    	4 unit tests

```
it('should return correct weather information', async () => {
  const response = await getWeatherInfo(userId);
  const { success, current, today, tomorrow } = response;

  expect(success).toBeTruthy();
  expect(current).toBeDefined();
  expect(today).toBeDefined();
  expect(tomorrow).toBeDefined();
});
```

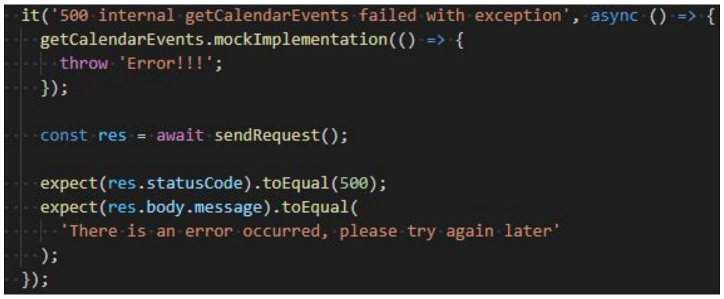
```
it('should return error message if input user id is invalid', async () => {
  const response = await getWeatherInfo(123);
  const { success, code, message } = response;

  expect(success).toBeFalsy();
  expect(code).toEqual(500);
  expect(message).toEqual(
    'Could not get your information, please try again later'
  );
});
```

```
it('should return correct geo-location if input is valid', async () => {
  place = 'vancouver';

  const response = await getGeoCode(place);

  const { success, lat, lon } = response;
  expect(success).toBeTruthy();
  expect(lat).toEqual(49.2608724);
  expect(lon).toEqual(-123.1139529);
});
```

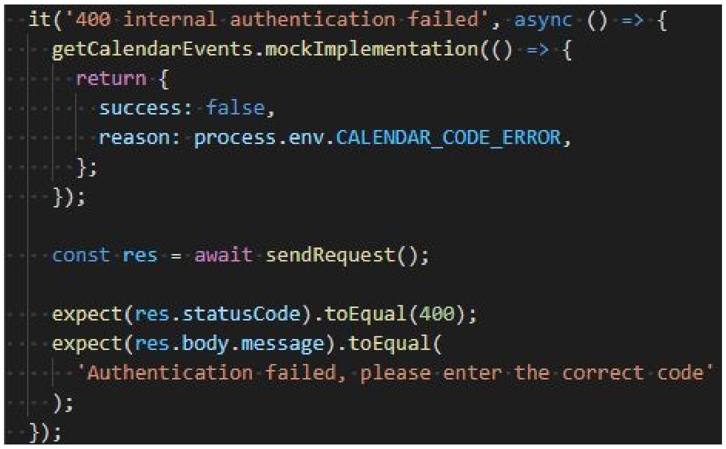
```
it('should return error message if the input is not valid', async () => {
  place = 'a';
  const response = await getGeoCode(place);

  const { success, code, message } = response;
  expect(success).toBeFalsy();
  expect(code).toEqual(500);
  expect(message).toEqual(
    'Cannot find your city, please check and try again'
  );
});
```

Calendar controller tests (controller/calendar.test.js) 	5 unit tests

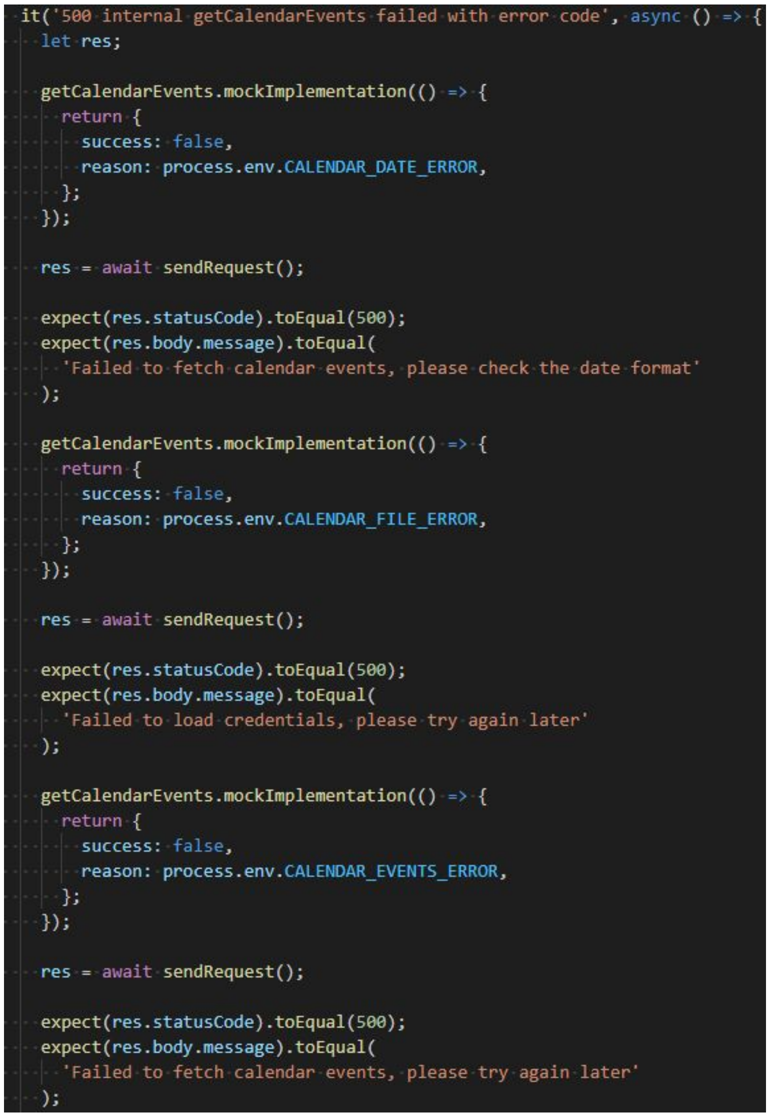
```
it('400 missing parameters', async () => {
  let res;
  res = await api
    .post(`/api/calendar/${null}`)
    .set('Authorization', `Bear ${token}`);

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing parameters');

  res = await api
    .post(`/api/calendar/${'DATE'}`)
    .set('Authorization', `Bear ${token}`)
    .send({ code: null });

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual('Missing parameters');
});
```



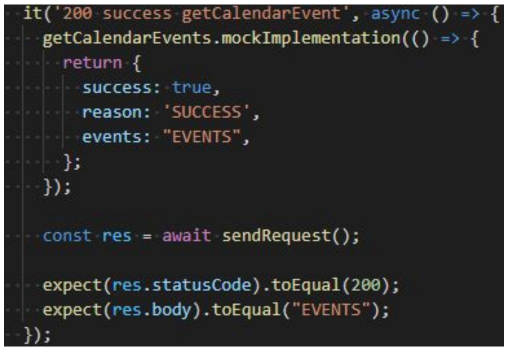
```javascript
it('500 internal getCalendarEvents failed with exception', async () => {
  getCalendarEvents.mockImplementation(() => {
    throw 'Error!!!';
  });

  const res = await sendRequest();

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'There is an error occurred, please try again later'
  );
});
```



```javascript
it('400 internal authentication failed', async () => {
  getCalendarEvents.mockImplementation(() => {
    return {
      success: false,
      reason: process.env.CALENDAR_CODE_ERROR,
    };
  });

  const res = await sendRequest();

  expect(res.statusCode).toEqual(400);
  expect(res.body.message).toEqual(
    'Authentication failed, please enter the correct code'
  );
});
```

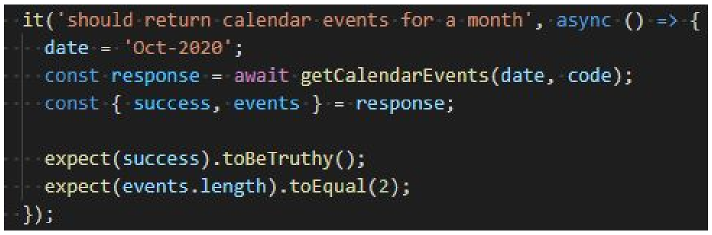


```javascript
it('500 internal getCalendarEvents failed with error code', async () => {
  let res;

  getCalendarEvents.mockImplementation(() => {
    return {
      success: false,
      reason: process.env.CALENDAR_DATE_ERROR,
    };
  });

  res = await sendRequest();

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to fetch calendar events, please check the date format'
  );

  getCalendarEvents.mockImplementation(() => {
    return {
      success: false,
      reason: process.env.CALENDAR_FILE_ERROR,
    };
  });

  res = await sendRequest();

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to load credentials, please try again later'
  );

  getCalendarEvents.mockImplementation(() => {
    return {
      success: false,
      reason: process.env.CALENDAR_EVENTS_ERROR,
    };
  });

  res = await sendRequest();

  expect(res.statusCode).toEqual(500);
  expect(res.body.message).toEqual(
    'Failed to fetch calendar events, please try again later'
  );
```

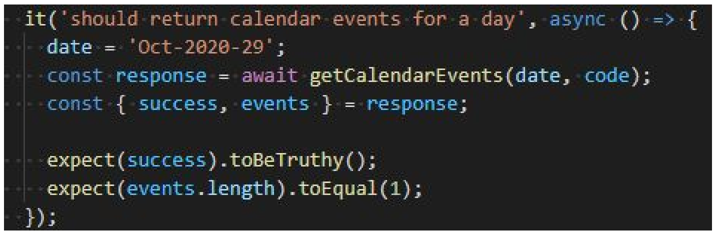


```javascript
it('200 success getCalendarEvent', async () => {
  getCalendarEvents.mockImplementation(() => {
    return {
      success: true,
      reason: 'SUCCESS',
      events: "EVENTS",
    };
  });

  const res = await sendRequest();

  expect(res.statusCode).toEqual(200);
  expect(res.body).toEqual("EVENTS");
});
```

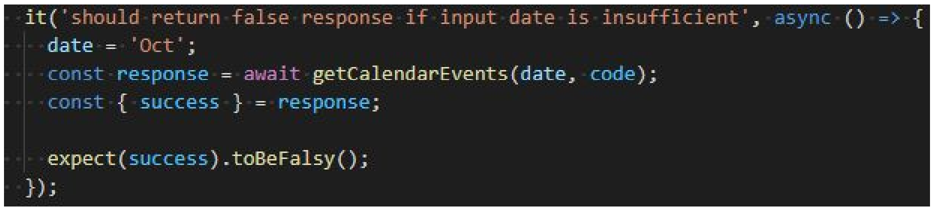
Calendar service tests (service/calendar.test.js)	4 unit tests



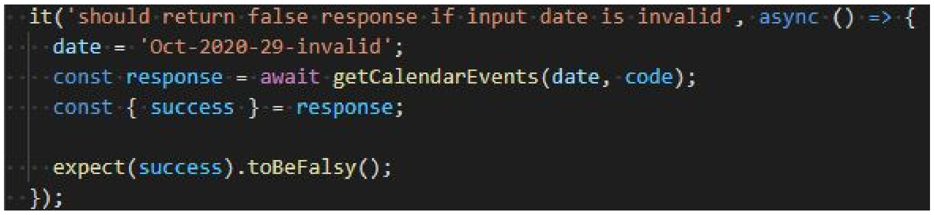
```
it('should return calendar events for a month', async () => {
  date = 'Oct-2020';
  const response = await getCalendarEvents(date, code);
  const { success, events } = response;

  expect(success).toBeTruthy();
  expect(events.length).toEqual(2);
});
```



```
it('should return calendar events for a day', async () => {
  date = 'Oct-2020-29';
  const response = await getCalendarEvents(date, code);
  const { success, events } = response;

  expect(success).toBeTruthy();
  expect(events.length).toEqual(1);
});
```

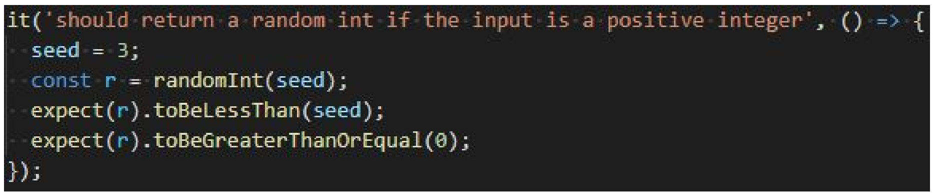


```
it('should return false response if input date is insufficient', async () => {
  date = 'Oct';
  const response = await getCalendarEvents(date, code);
  const { success } = response;

  expect(success).toBeFalsy();
});
```

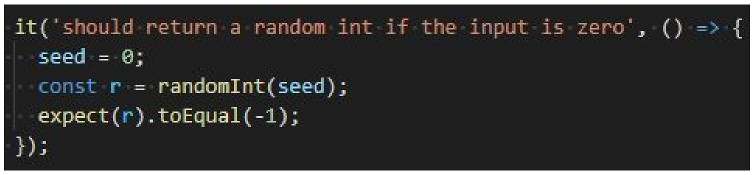


```
it('should return false response if input date is invalid', async () => {
  date = 'Oct-2020-29-invalid';
  const response = await getCalendarEvents(date, code);
  const { success } = response;

  expect(success).toBeFalsy();
});
```

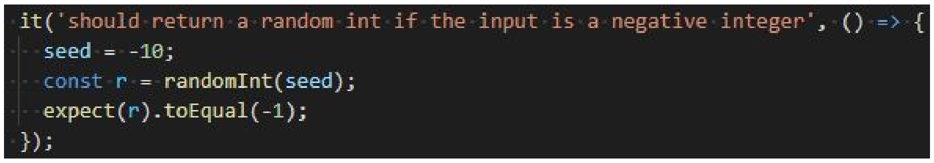
Hash utility function tests (utils/hash.test.js)	3 unit tests



```
it('should return a random int if the input is a positive integer', () => {
  seed = 3;
  const r = randomInt(seed);
  expect(r).toBeLessThan(seed);
  expect(r).toBeGreaterThanOrEqual(0);
});
```

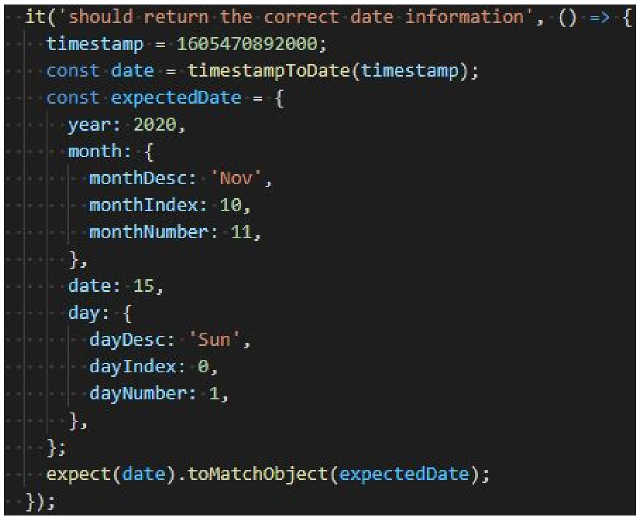


```
it('should return a random int if the input is zero', () => {
  seed = 0;
  const r = randomInt(seed);
  expect(r).toEqual(-1);
});
```

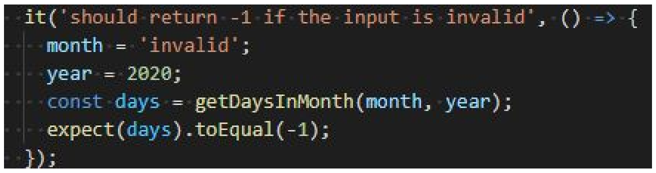


```
it('should return a random int if the input is a negative integer', () => {
  seed = -10;
  const r = randomInt(seed);
  expect(r).toEqual(-1);
});
```

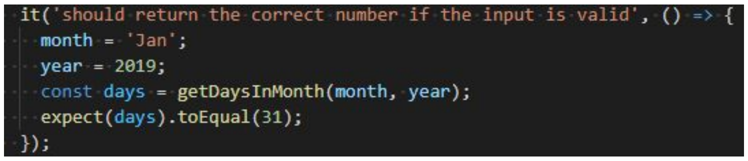
Time helper function tests (utils/time-helper.test.js)	5 unit tests



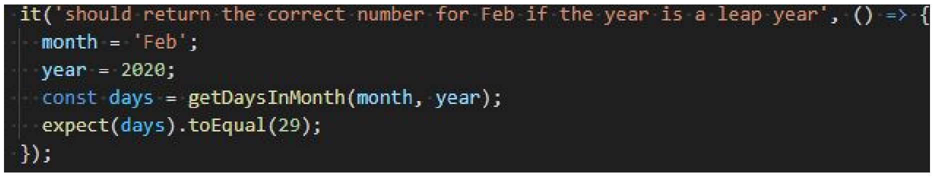
```
it('should return the correct date information', () => {
  timestamp = 1605470892000;
  const date = timestampToDate(timestamp);
  const expectedDate = {
    year: 2020,
    month: {
      monthDesc: 'Nov',
      monthIndex: 10,
      monthNumber: 11,
    },
    date: 15,
    day: {
      dayDesc: 'Sun',
      dayIndex: 0,
      dayNumber: 1,
    },
  };
  expect(date).toMatchObject(expectedDate);
});
```



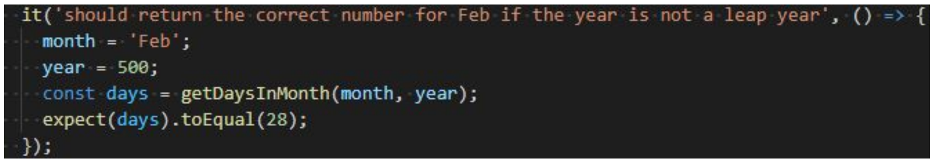
```
it('should return -1 if the input is invalid', () => {
  month = 'invalid';
  year = 2020;
  const days = getDaysInMonth(month, year);
  expect(days).toEqual(-1);
});
```



```
it('should return the correct number if the input is valid', () => {
  month = 'Jan';
  year = 2019;
  const days = getDaysInMonth(month, year);
  expect(days).toEqual(31);
});
```

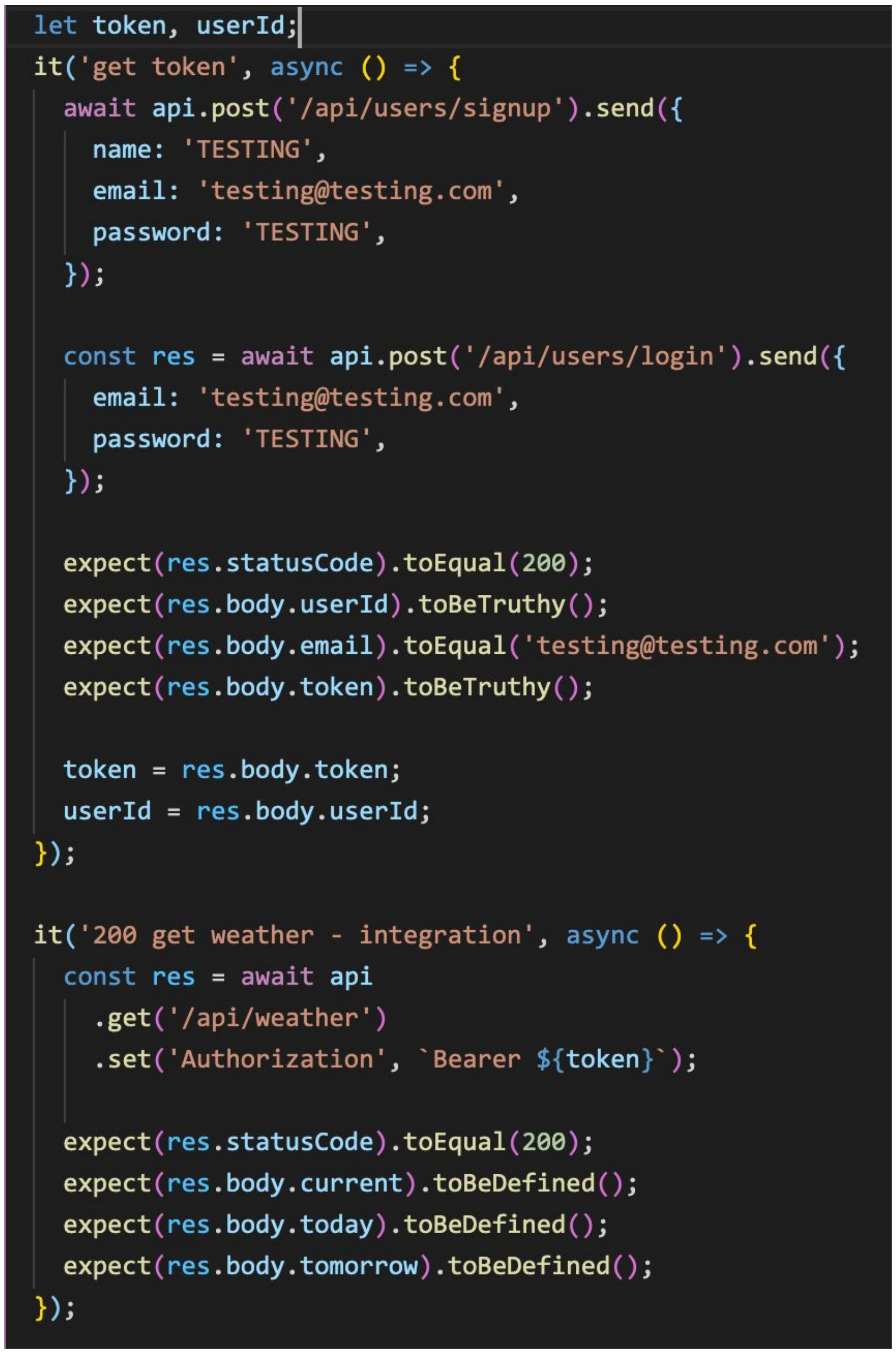


```
it('should return the correct number for Feb if the year is a leap year', () => {
  month = 'Feb';
  year = 2020;
  const days = getDaysInMonth(month, year);
  expect(days).toEqual(29);
});
```



```
it('should return the correct number for Feb if the year is not a leap year', () => {
  month = 'Feb';
  year = 500;
  const days = getDaysInMonth(month, year);
  expect(days).toEqual(28);
});
```

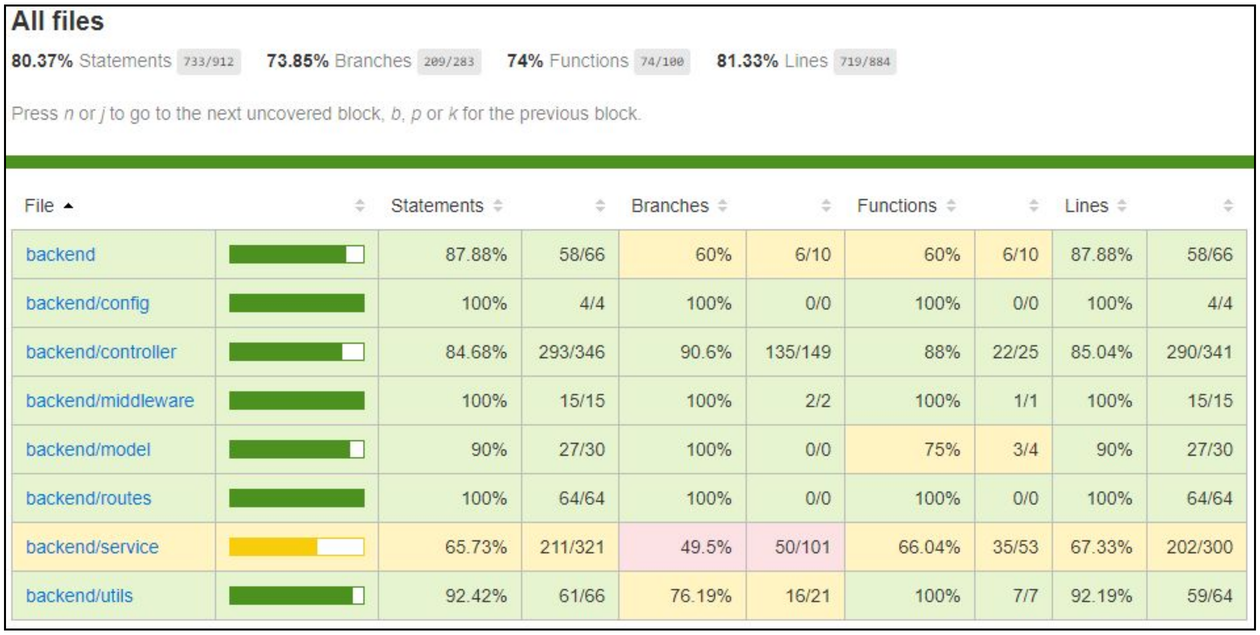
Get weather integration test (integration/weather.test.js)	1 integration test



```javascript
let token, userId;
it('get token', async () => {
  await api.post('/api/users/signup').send({
    name: 'TESTING',
    email: 'testing@testing.com',
    password: 'TESTING',
  });

  const res = await api.post('/api/users/login').send({
    email: 'testing@testing.com',
    password: 'TESTING',
  });

  expect(res.statusCode).toEqual(200);
  expect(res.body.userId).toBeTruthy();
  expect(res.body.email).toEqual('testing@testing.com');
  expect(res.body.token).toBeTruthy();

  token = res.body.token;
  userId = res.body.userId;
});

it('200 get weather - integration', async () => {
  const res = await api
    .get('/api/weather')
    .set('Authorization', `Bearer ${token}`);

  expect(res.statusCode).toEqual(200);
  expect(res.body.current).toBeDefined();
  expect(res.body.today).toBeDefined();
  expect(res.body.tomorrow).toBeDefined();
});
```

A statement- and method-level coverage report for tests of each of the two modules under test

All unit tests



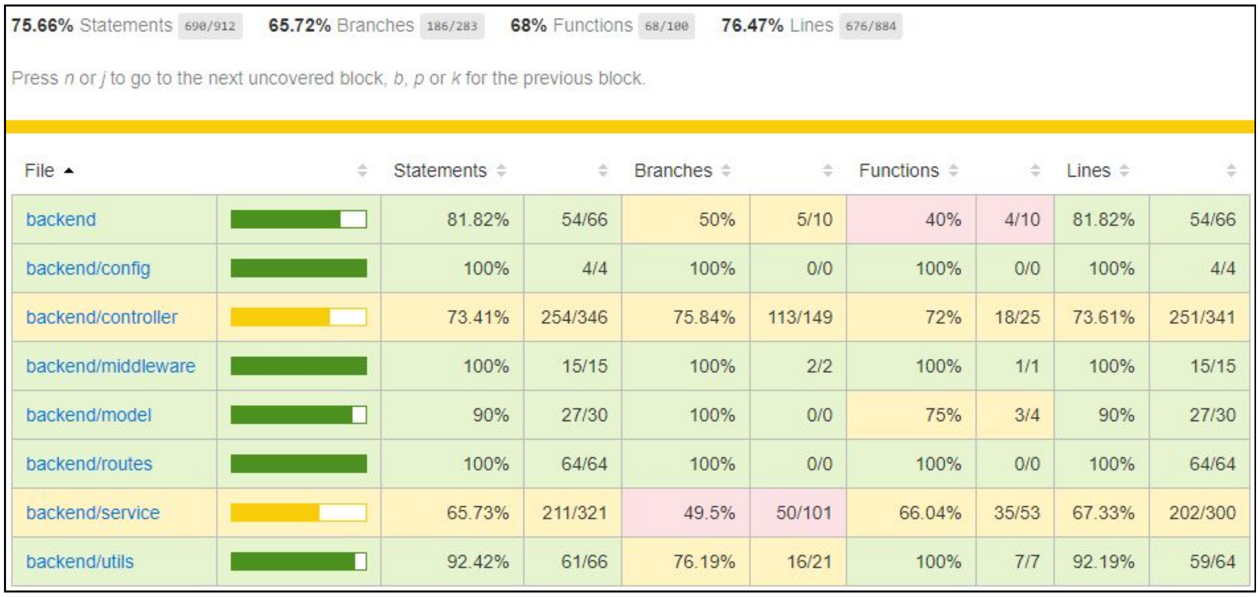
All files

80.37% Statements 733/912 **73.85%** Branches 209/283 **74%** Functions 74/100 **81.33%** Lines 719/884

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements		Branches		Functions		Lines	
backend		87.88%	58/66	60%	6/10	60%	6/10	87.88%	58/66
backend/config		100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/controller		84.68%	293/346	90.6%	135/149	88%	22/25	85.04%	290/341
backend/middleware		100%	15/15	100%	2/2	100%	1/1	100%	15/15
backend/model		90%	27/30	100%	0/0	75%	3/4	90%	27/30
backend/routes		100%	64/64	100%	0/0	100%	0/0	100%	64/64
backend/service		65.73%	211/321	49.5%	50/101	66.04%	35/53	67.33%	202/300
backend/utils		92.42%	61/66	76.19%	16/21	100%	7/7	92.19%	59/64

Only listed modules (11 files)

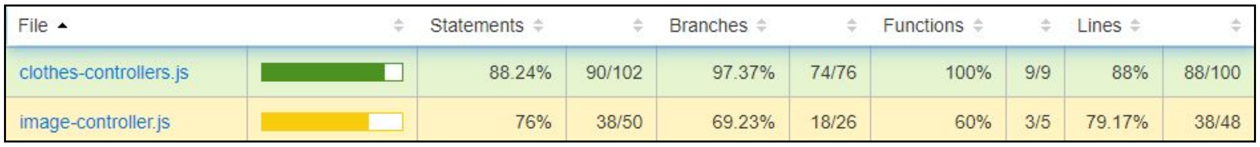


75.66% Statements 690/912 **65.72%** Branches 186/283 **68%** Functions 68/100 **76.47%** Lines 676/884

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

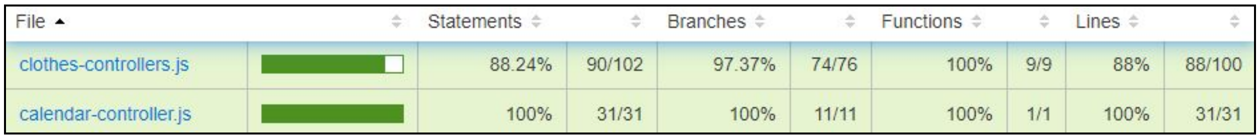
File ▲		Statements		Branches		Functions		Lines	
backend		81.82%	54/66	50%	5/10	40%	4/10	81.82%	54/66
backend/config		100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/controller		73.41%	254/346	75.84%	113/149	72%	18/25	73.61%	251/341
backend/middleware		100%	15/15	100%	2/2	100%	1/1	100%	15/15
backend/model		90%	27/30	100%	0/0	75%	3/4	90%	27/30
backend/routes		100%	64/64	100%	0/0	100%	0/0	100%	64/64
backend/service		65.73%	211/321	49.5%	50/101	66.04%	35/53	67.33%	202/300
backend/utils		92.42%	61/66	76.19%	16/21	100%	7/7	92.19%	59/64

Clothes module (2 files)

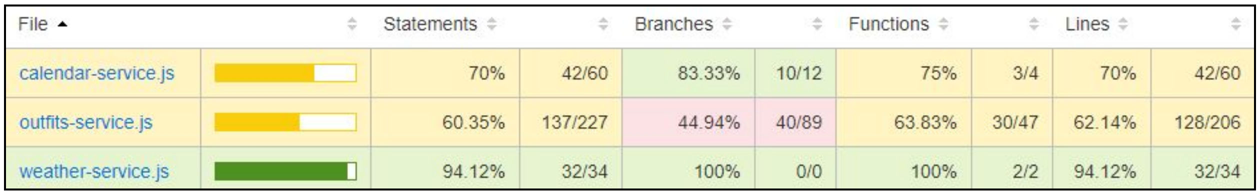


File ▲		Statements		Branches		Functions		Lines	
clothes-controllers.js		88.24%	90/102	97.37%	74/76	100%	9/9	88%	88/100
image-controller.js		76%	38/50	69.23%	18/26	60%	3/5	79.17%	38/48

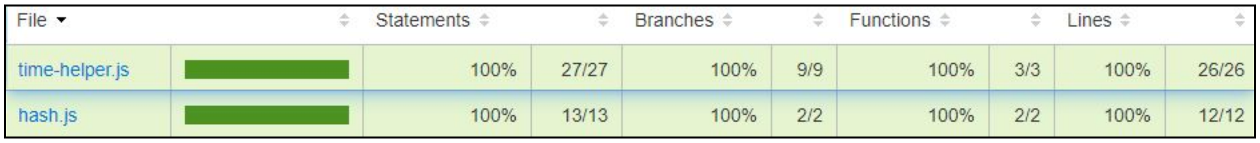
Outfit module (7 files)



File ▲		Statements		Branches		Functions		Lines	
clothes-controllers.js		88.24%	90/102	97.37%	74/76	100%	9/9	88%	88/100
calendar-controller.js		100%	31/31	100%	11/11	100%	1/1	100%	31/31



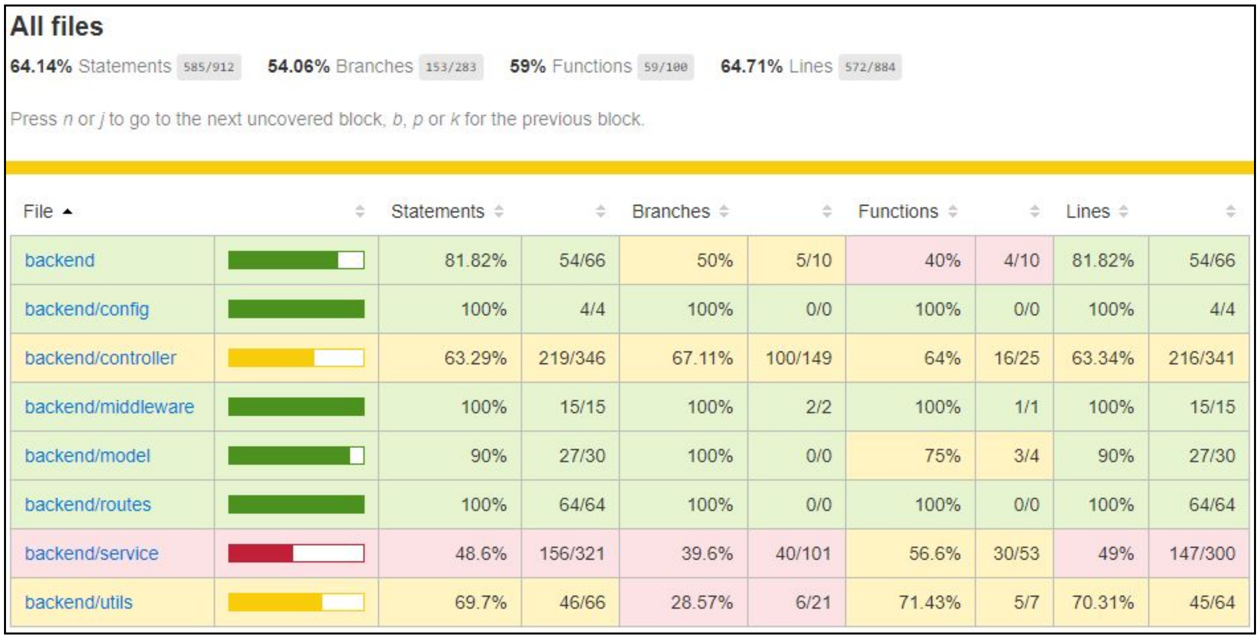
File ▲		Statements		Branches		Functions		Lines	
calendar-service.js		70%	42/60	83.33%	10/12	75%	3/4	70%	42/60
outfits-service.js		60.35%	137/227	44.94%	40/89	63.83%	30/47	62.14%	128/206
weather-service.js		94.12%	32/34	100%	0/0	100%	2/2	94.12%	32/34



File ▼		Statements		Branches		Functions		Lines	
time-helper.js		100%	27/27	100%	9/9	100%	3/3	100%	26/26
hash.js		100%	13/13	100%	2/2	100%	2/2	100%	12/12

A statement- and method-level coverage report for each integration test separately and all integration tests together

Overall



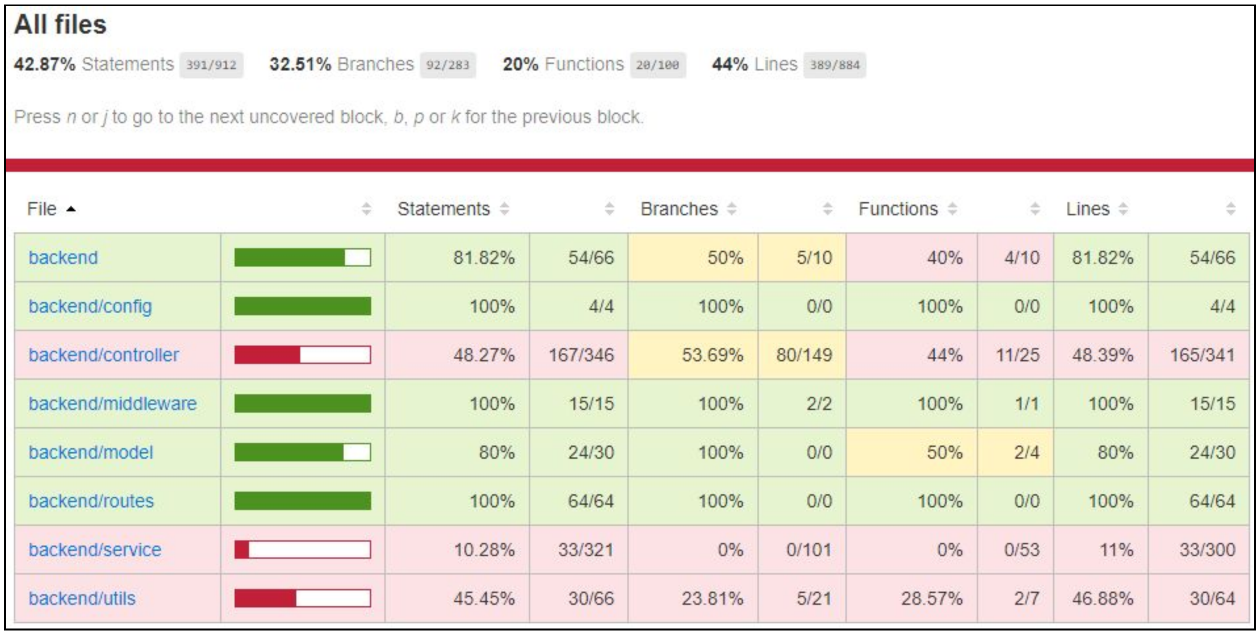
All files

64.14% Statements 585/912 **54.06%** Branches 153/283 **59%** Functions 59/100 **64.71%** Lines 572/884

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements		Branches		Functions		Lines	
backend		81.82%	54/66	50%	5/10	40%	4/10	81.82%	54/66
backend/config		100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/controller		63.29%	219/346	67.11%	100/149	64%	16/25	63.34%	216/341
backend/middleware		100%	15/15	100%	2/2	100%	1/1	100%	15/15
backend/model		90%	27/30	100%	0/0	75%	3/4	90%	27/30
backend/routes		100%	64/64	100%	0/0	100%	0/0	100%	64/64
backend/service		48.6%	156/321	39.6%	40/101	56.6%	30/53	49%	147/300
backend/utils		69.7%	46/66	28.57%	6/21	71.43%	5/7	70.31%	45/64

Clothes Module



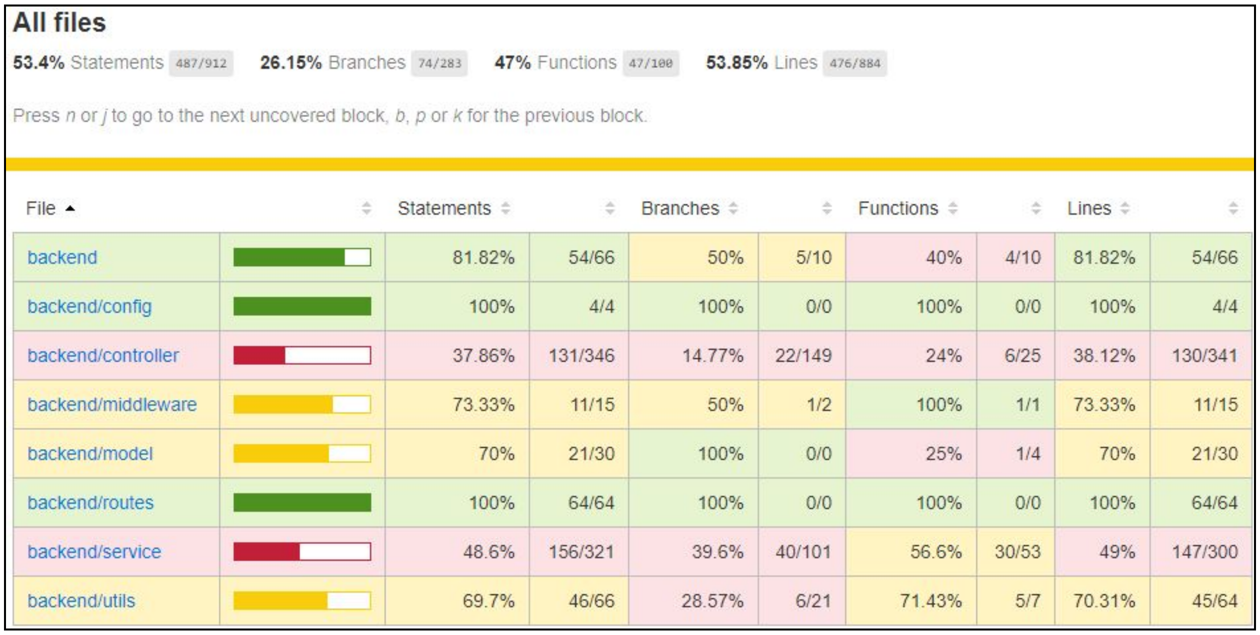
All files

42.87% Statements 391/912 **32.51%** Branches 92/283 **20%** Functions 20/100 **44%** Lines 389/884

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements		Branches		Functions		Lines	
backend	🟩	81.82%	54/66	50%	5/10	40%	4/10	81.82%	54/66
backend/config	🟩	100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/controller	🟥	48.27%	167/346	53.69%	80/149	44%	11/25	48.39%	165/341
backend/middleware	🟩	100%	15/15	100%	2/2	100%	1/1	100%	15/15
backend/model	🟩	80%	24/30	100%	0/0	50%	2/4	80%	24/30
backend/routes	🟩	100%	64/64	100%	0/0	100%	0/0	100%	64/64
backend/service	🟥	10.28%	33/321	0%	0/101	0%	0/53	11%	33/300
backend/utils	🟥	45.45%	30/66	23.81%	5/21	28.57%	2/7	46.88%	30/64

Outfit Module



All files

53.4% Statements 487/912 **26.15%** Branches 74/283 **47%** Functions 47/100 **53.85%** Lines 476/884

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements		Branches		Functions		Lines	
backend	🟩	81.82%	54/66	50%	5/10	40%	4/10	81.82%	54/66
backend/config	🟩	100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/controller	🟥	37.86%	131/346	14.77%	22/149	24%	6/25	38.12%	130/341
backend/middleware	🟨	73.33%	11/15	50%	1/2	100%	1/1	73.33%	11/15
backend/model	🟨	70%	21/30	100%	0/0	25%	1/4	70%	21/30
backend/routes	🟩	100%	64/64	100%	0/0	100%	0/0	100%	64/64
backend/service	🟥	48.6%	156/321	39.6%	40/101	56.6%	30/53	49%	147/300
backend/utils	🟨	69.7%	46/66	28.57%	6/21	71.43%	5/7	70.31%	45/64

Automated front-end UI testing

A list of UI tests with how we encoded success/failure criteria for each UI tests and screenshots

1. RegisterWithExistingAccountTest

First when just starting the test, we check that the register screen is correctly displayed by checking all the UI components on the register screen is displayed, as *figure 1* shown. We check that the user cannot register with an already existing account by verifying the toast message in *figure 2* is displayed and matched with text and all the UI components on the register screen are still displayed on the screen. Finally, we check that the user can jump to the login screen from the register screen by clicking the text "already have an account? Login here" by verifying all the UI components on the login screen is displayed as *figure 3* shown.

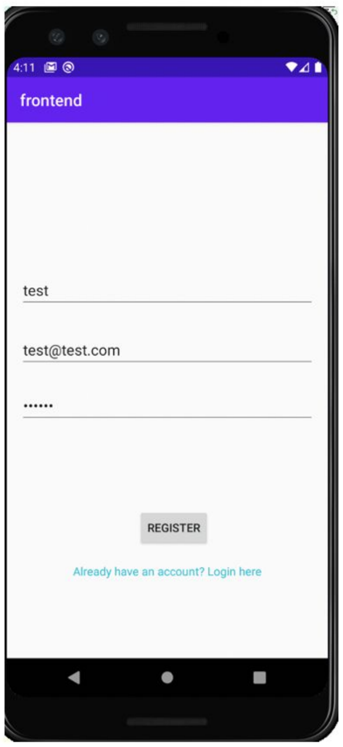
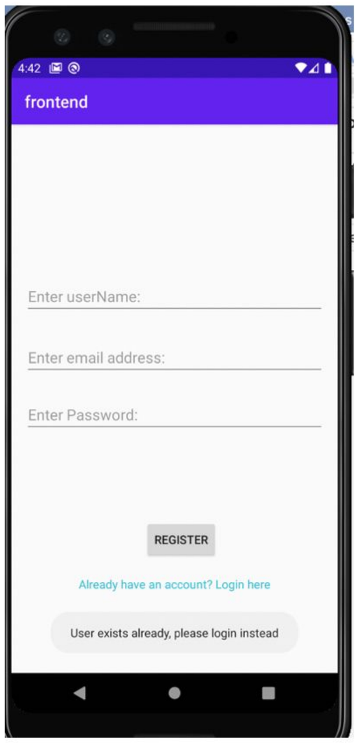


Figure 1: A screenshot right before clicking the REGISTER button



User exists already, please login instead

Figure 2: A screenshot after clicking the REGISTER button

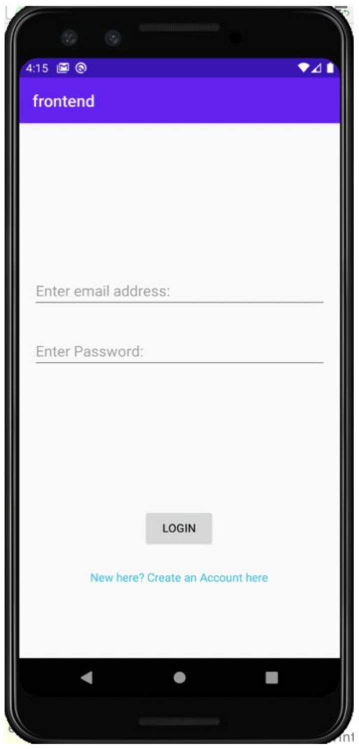


Figure 3: A screenshot after clicking "Already have an account? Login here" in register screen

2. Login Test

2.1. loginWithInvalidAccountTest

First when just starting the test, we check that the login screen is correctly displayed by checking all the UI components(text ,buttons, and editable text boxes) on the login screen is displayed, as *figure 1* shown. We then check that the user can't login with invalid account information by verifying the toast message in *figure 2* is displayed and matched with text, and also all the UI components on the login screen are still displayed. Finally, we check that the user can jump to the register screen from the login screen by clicking the text "New here? Create an Account here" by verifying all the UI components on the register screen is displayed as *figure 3* shown.

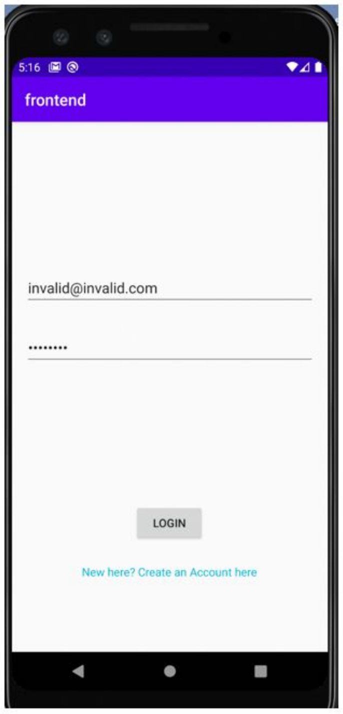
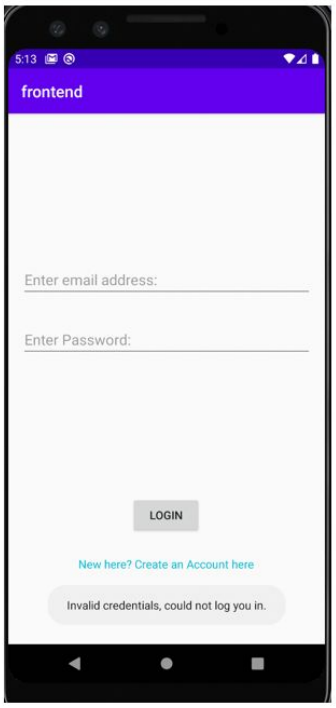


Figure 1: A screenshot right before clicking the LOGIN button



Invalid credentials, could not log you in.

Figure 2: A screenshot after clicking the LOGIN button

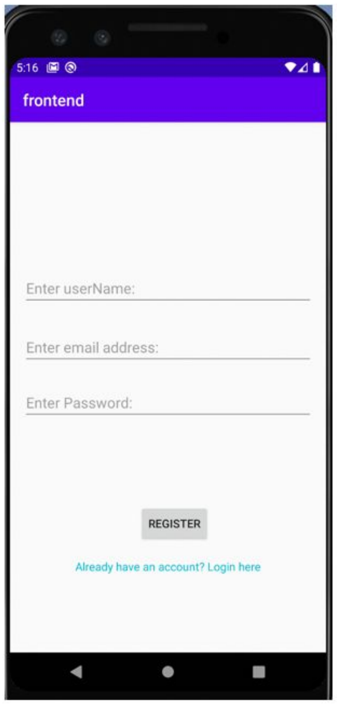


Figure 3: A screenshot after clicking "New here? Create an Account here" in login screen

2.2. loginSuccessTest

First when just starting the test, we check that the login screen is correctly displayed by checking all the UI components on the login screen is displayed, as *figure 1* shown. We then check that the user can login successfully with an valid account and jump to the main screen by verifying the toast message in *figure 2* is displayed and matched with text, and a view on the main activity is displayed on the screen as *figure 3* shown.

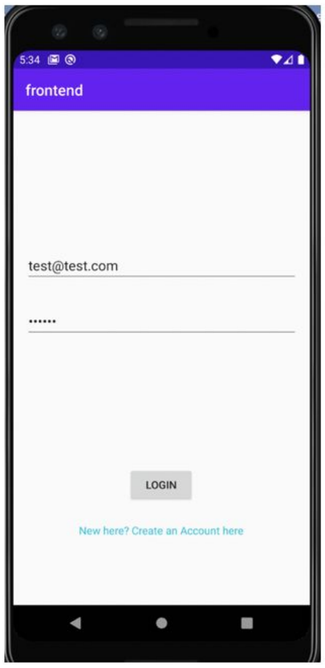


Figure 1: A screenshot right before clicking the LOGIN button

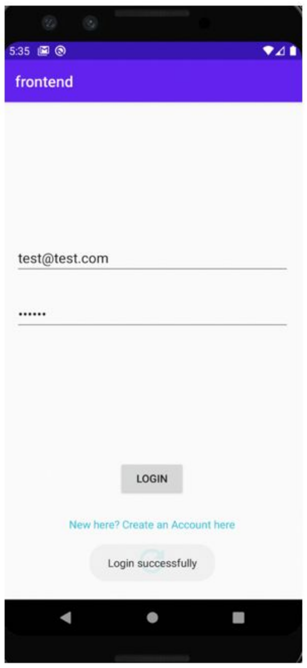


Figure 2: A screenshot after clicking the LOGIN button; A successful- login toast message is shown

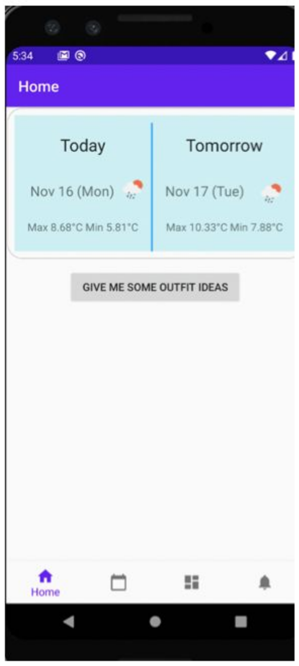


Figure 3: A screenshot after clicking the LOGIN button and jump to the main screen

3. OutfitTest

On the start, we check the view we used to display an outfit is not present and the get outfit button "GIVE ME SOME OUTFIT IDEA" is enabled. We then click the get outfit button and check the view to display an outfit is displayed on the screen as well as the like/dislike buttons are present and enabled, as *figure 1* shown. Now the user should be able to specify their opinion about the outfit. We click the dislike button, and verify the user's opinion has been recorded by checking the undo view is shown, the text "We will not suggest this outfit any more, OR " is displayed, the undo button is enabled, and the like/dislike buttons are not enabled as *figure 2* shown. We then click the undo button, and verify successfully undo by verifying undo view is not displayed on the screen anymore and the like/dislike buttons are enabled for users to specify opinion again. Finally we click the like button, and verify the user's opinion has been recorded by checking a toast with text "Your preference has been recorded" is displayed as *figure 3* shown.

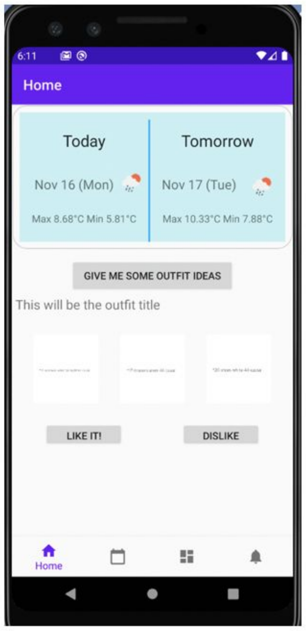
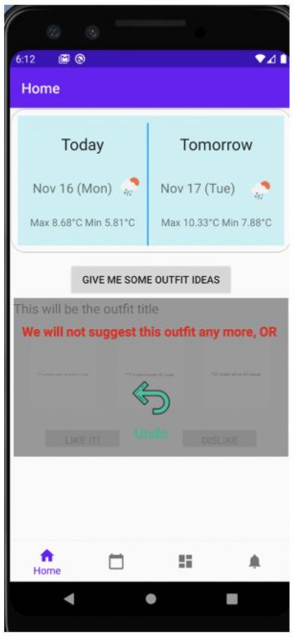
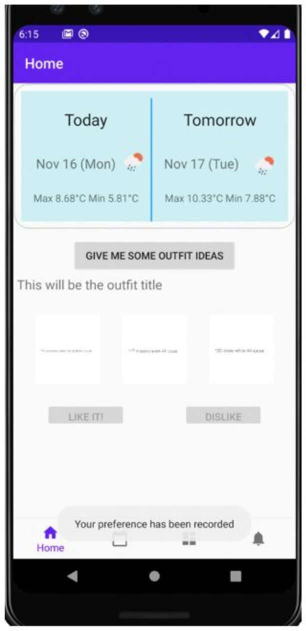


Figure 1: A screenshot after clicking the GIVE ME SOME OUTFIT IDEAS button



We will not suggest this outfit any more, OR

Figure 2: A screenshot after clicking the DISLIKE button; The undo view and undo button are shown

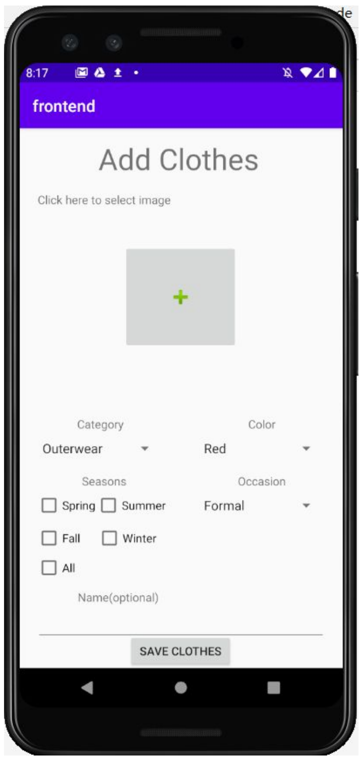


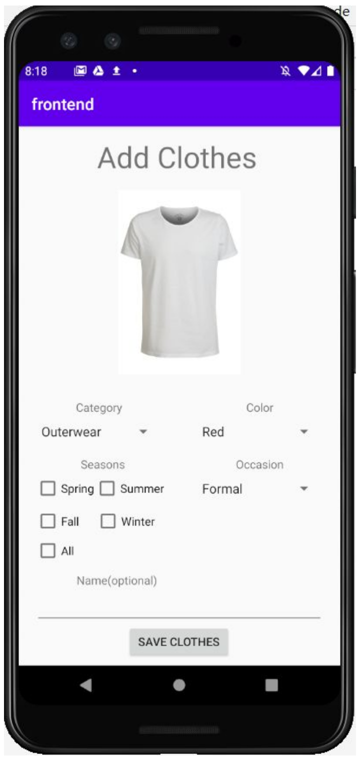
Your preference has been recorded

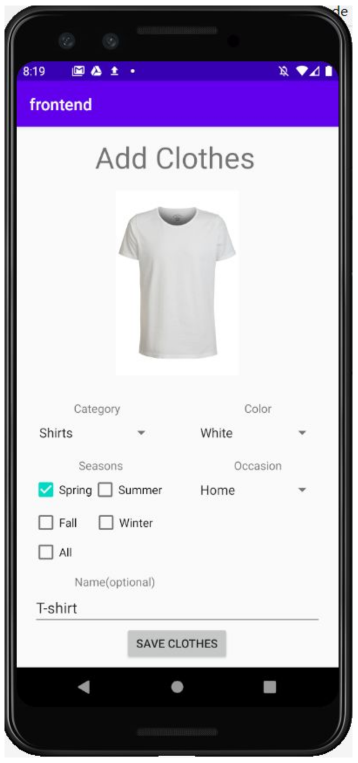
Figure 3: A screenshot after clicking like button; A preference-record toast message is shown

4. AddClothTest

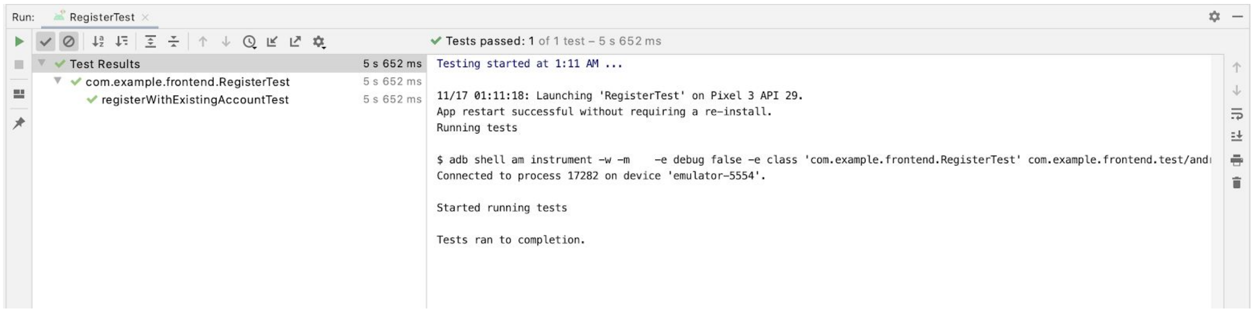
On start, we click the add clothes button in the clothes navigation, then we should be navigated to add clothes activity. In the activity, we click the add image button, and when we come back from gallery, the imageview should display the image we selected, and the add image button should not present. Then we click and select information in the spinner and check boxes, and enter information in the text box. Finally, we click the save button and we should be navigated back to the clothes navigation with a new clothes item displayed.







A log of automated execution and passed status

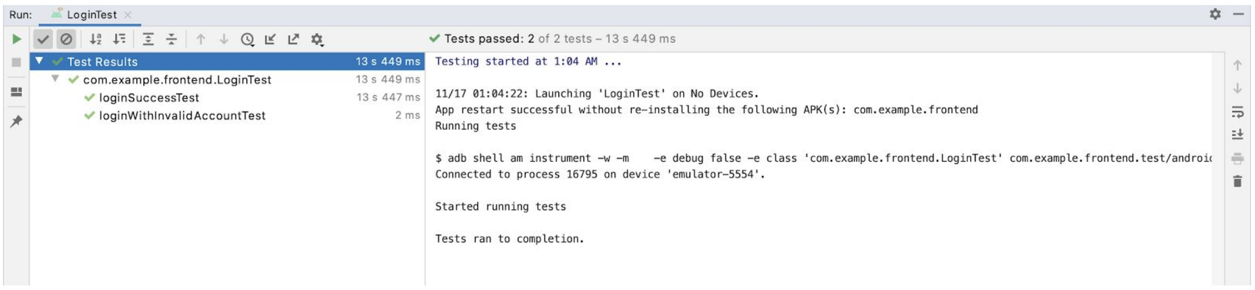


Run: RegisterTest

✓ Tests passed: 1 of 1 test – 5 s 652 ms

```
✓ Test Results                                  5 s 652 ms
  ✓ com.example.frontend.RegisterTest           5 s 652 ms
    ✓ registerWithExistingAccountTest           5 s 652 ms
```

```
Testing started at 1:11 AM ...

11/17 01:11:18: Launching 'RegisterTest' on Pixel 3 API 29.
App restart successful without requiring a re-install.
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'com.example.frontend.RegisterTest' com.example.frontend.test/andı
Connected to process 17282 on device 'emulator-5554'.

Started running tests

Tests ran to completion.
```

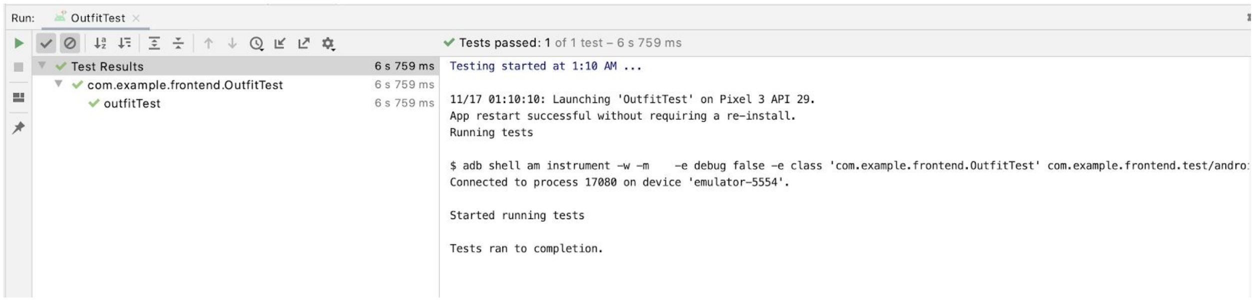


Run: LoginTest

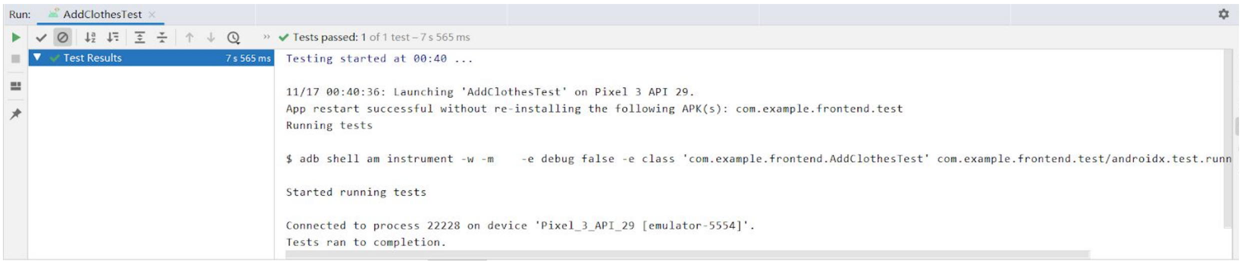
✓ Tests passed: 2 of 2 tests – 13 s 449 ms

```
✓ Test Results                                  13 s 449 ms
  ✓ com.example.frontend.LoginTest              13 s 449 ms
    ✓ loginSuccessTest                          13 s 447 ms
    ✓ loginWithInvalidAccountTest                     2 ms
```

```
Testing started at 1:04 AM ...

11/17 01:04:22: Launching 'LoginTest' on No Devices.
App restart successful without re-installing the following APK(s): com.example.frontend
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'com.example.frontend.LoginTest' com.example.frontend.test/androic
Connected to process 16795 on device 'emulator-5554'.

Started running tests

Tests ran to completion.
```



```
✓ Tests passed: 1 of 1 test – 6 s 759 ms

✓ Test Results                      6 s 759 ms
  ✓ com.example.frontend.OutfitTest 6 s 759 ms
    ✓ outfitTest                    6 s 759 ms

Testing started at 1:10 AM ...

11/17 01:10:10: Launching 'OutfitTest' on Pixel 3 API 29.
App restart successful without requiring a re-install.
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'com.example.frontend.OutfitTest' com.example.frontend.test/andro
Connected to process 17080 on device 'emulator-5554'.

Started running tests

Tests ran to completion.
```

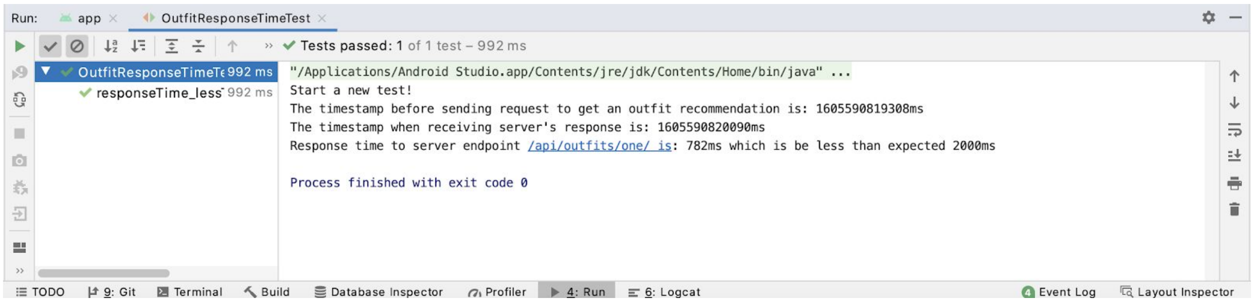


```
✓ Tests passed: 1 of 1 test – 7 s 565 ms

✓ Test Results                      7 s 565 ms

Testing started at 00:40 ...

11/17 00:40:36: Launching 'AddClothesTest' on Pixel 3 API 29.
App restart successful without re-installing the following APK(s): com.example.frontend.test
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'com.example.frontend.AddClothesTest' com.example.frontend.test/androidx.test.runn

Started running tests

Connected to process 22228 on device 'Pixel_3_API_29 [emulator-5554]'.
Tests ran to completion.
```

Testing Non-functional requirements

1. Users should wait less than 2 seconds when requesting for an outfit suggestion.

The log with verification results:

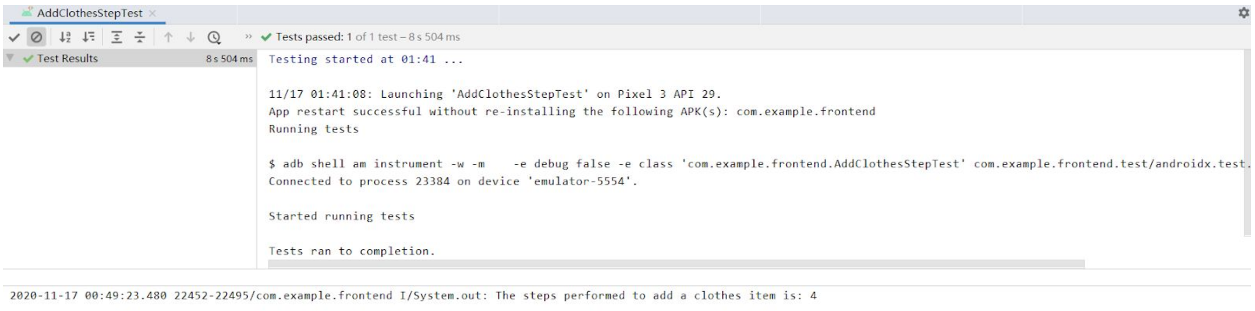

Run: app OutfitResponseTimeTest
Tests passed: 1 of 1 test – 992 ms
OutfitResponseTimeTe 992 ms
responseTime_less 992 ms
TODO 9: Git Terminal Build Database Inspector Profiler 4: Run 6: Logcat Event Log Layout Inspector

```
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Start a new test!
The timestamp before sending request to get an outfit recommendation is: 1605590819308ms
The timestamp when receiving server's response is: 1605590820090ms
Response time to server endpoint /api/outfits/one/ is: 782ms which is be less than expected 2000ms

Process finished with exit code 0
```

We test the "/api/outfits/one" endpoint's response time in Android Studio by writing a UnitTest OutfitResponseTimeTest. In the test, we record the timestamp(through the function System.*currentTimeMillis*()) before and after sending a get-an-outfit request to the server and subtract the 2 timestamps to get the response time,verifying the responseTime requirement is met by *assertTrue*(elapsedTime<=2000);

2. Users do not need more than 5 steps to add a cloth item

Log with verification result:



```
Testing started at 01:41 ...

11/17 01:41:08: Launching 'AddClothesStepTest' on Pixel 3 API 29.
App restart successful without re-installing the following APK(s): com.example.frontend
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'com.example.frontend.AddClothesStepTest' com.example.frontend.test/androidx.test.
Connected to process 23384 on device 'emulator-5554'.

Started running tests

Tests ran to completion.
```

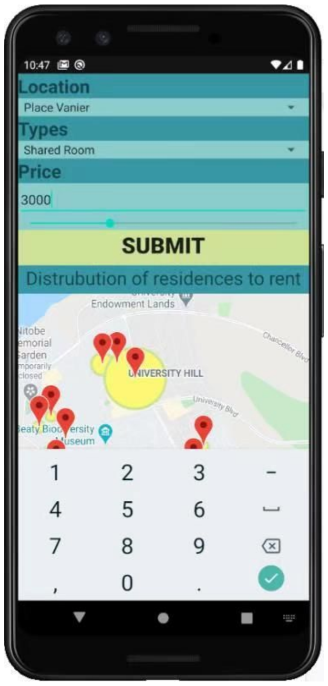
```
2020-11-17 00:49:23.480 22452-22495/com.example.frontend I/System.out: The steps performed to add a clothes item is: 4
```

We count the steps performed in adding clothes by the category of the actions: 1. Click add clothes button; 2. Add clothes image; 3. Input clothes information; 4. Click the save button. The requirement is met if the number of steps is less or equal to 5.

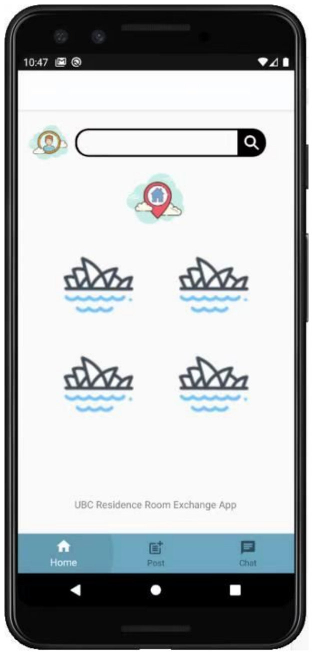
Manual customer acceptance testing

Report one major fault you found in your partner's team app

When submit search request to find a list of rooms that meets certain criterias as following:



The results of queries, such as price, image, host contact information, are supposed to show up on the main page. However, when returning back to the main page, no results are available as the following image indicates:



Report one major fault that your peer-team found in your app

When we have different clothes in the closet page and click "edit" on any piece of clothes, the clothes image displayed in the "edit clothes" page is always the last clothes in the closet.