SOEN 390

Team 15

Compilation of previous sprints

Rently

https://rently-management.netlify.app/

Winter 2024

Student	Student ID
Abdelkader Habel	40209153
Adam Boucher	40165035
Adel Bouchatta	40175598
Anes Khadiri	40159080
Chems-Eddine Saidi	40192094
Francesco Ferrato	26642152
Omar Mohammad	40162541
Oussama Cherifi	40212275
Zakaria El Manar El Bouanani	40190432

Table Of Contents

Testing documents

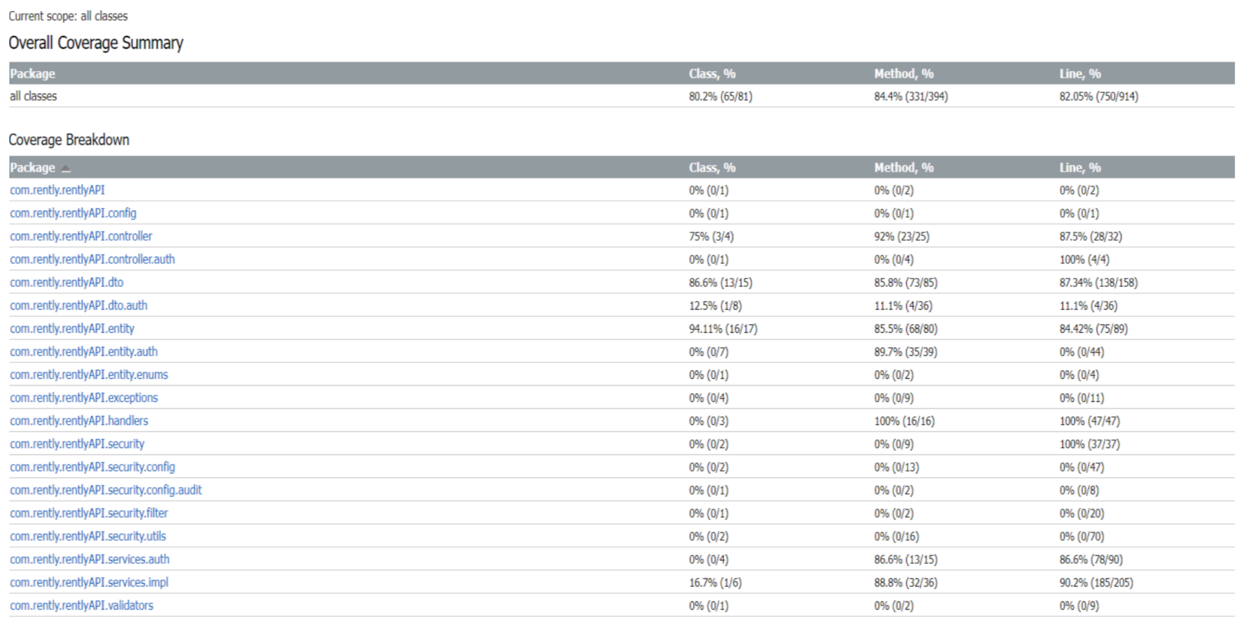
The testing documents here are as they were submitted in their respective sprints. We learned from our errors and updated our testing documents in sprint 4-5.

Sprint 2

Testing plan & coverage

The current test cases are built using Junit. For the next sprint, we plan to integrate JaCoCo, HemCrest and other libraries to improve and facilitate the testing process.

Current overall coverage summary : 80%+ code coverage



Current scope: all classes

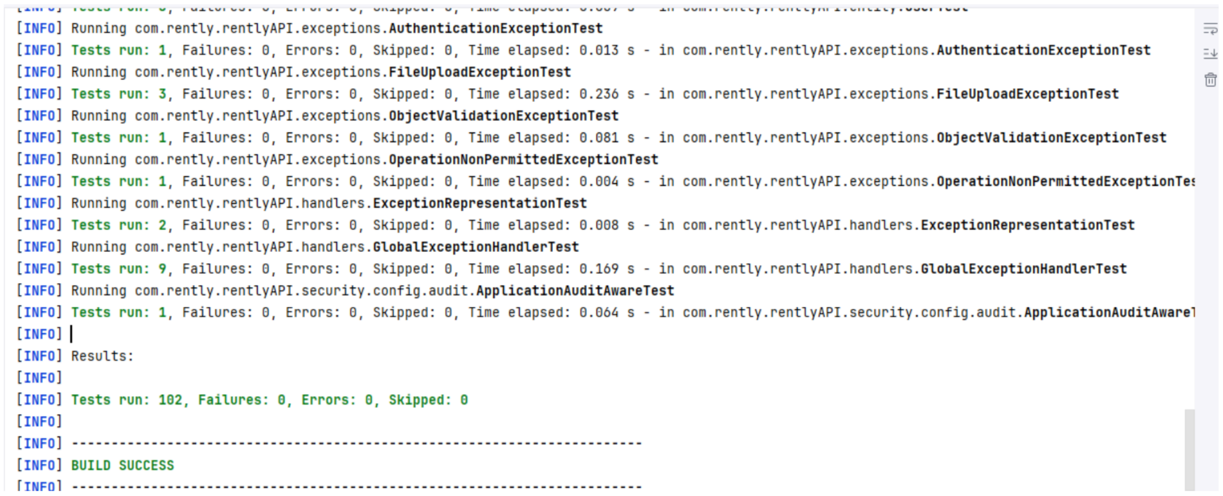
Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	80.2% (65/81)	84.4% (331/394)	82.05% (750/914)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
com.rently.rentlyAPI	0% (0/1)	0% (0/2)	0% (0/2)
com.rently.rentlyAPI.config	0% (0/1)	0% (0/1)	0% (0/1)
com.rently.rentlyAPI.controller	75% (3/4)	92% (23/25)	87.5% (28/32)
com.rently.rentlyAPI.controller.auth	0% (0/1)	0% (0/4)	100% (4/4)
com.rently.rentlyAPI.dto	86.6% (13/15)	85.8% (73/85)	87.34% (138/158)
com.rently.rentlyAPI.dto.auth	12.5% (1/8)	11.1% (4/36)	11.1% (4/36)
com.rently.rentlyAPI.entity	94.11% (16/17)	85.5% (68/80)	84.42% (75/89)
com.rently.rentlyAPI.entity.auth	0% (0/7)	89.7% (35/39)	0% (0/44)
com.rently.rentlyAPI.entity.enums	0% (0/1)	0% (0/2)	0% (0/4)
com.rently.rentlyAPI.exceptions	0% (0/4)	0% (0/9)	0% (0/11)
com.rently.rentlyAPI.handlers	0% (0/3)	100% (16/16)	100% (47/47)
com.rently.rentlyAPI.security	0% (0/2)	0% (0/9)	100% (37/37)
com.rently.rentlyAPI.security.config	0% (0/2)	0% (0/13)	0% (0/47)
com.rently.rentlyAPI.security.config.audit	0% (0/1)	0% (0/2)	0% (0/8)
com.rently.rentlyAPI.security.filter	0% (0/1)	0% (0/2)	0% (0/20)
com.rently.rentlyAPI.security.utils	0% (0/2)	0% (0/16)	0% (0/70)
com.rently.rentlyAPI.services.auth	0% (0/4)	86.6% (13/15)	86.6% (78/90)
com.rently.rentlyAPI.services.impl	16.7% (1/6)	88.8% (32/36)	90.2% (185/205)
com.rently.rentlyAPI.validators	0% (0/1)	0% (0/2)	0% (0/9)

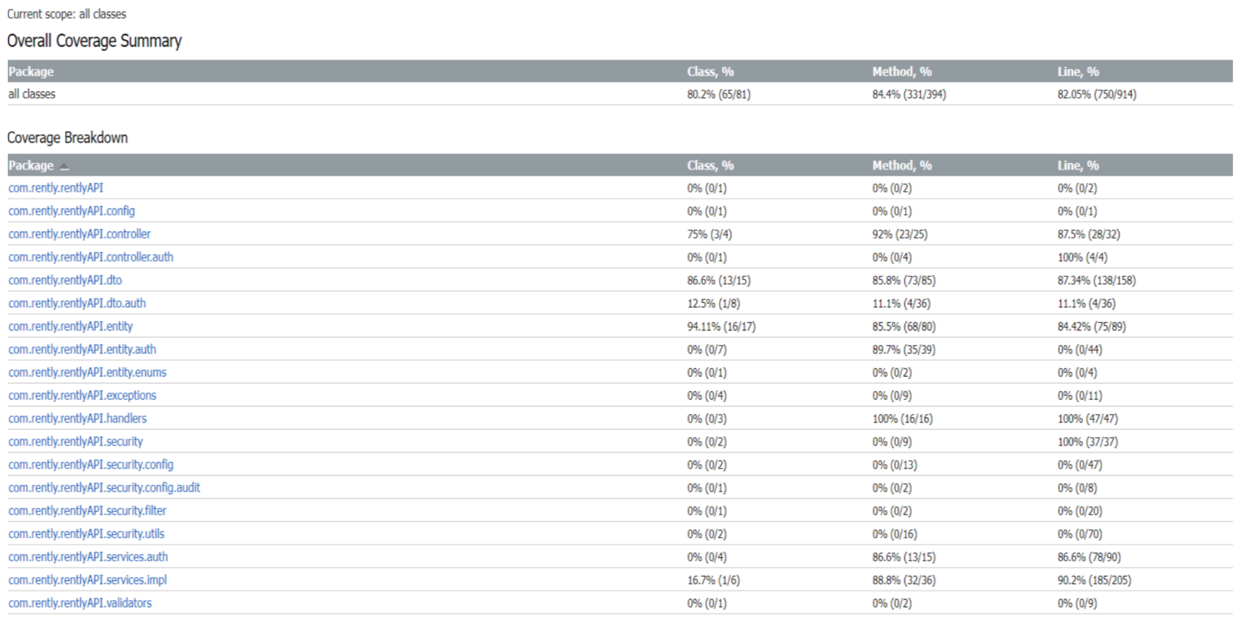
Sample test build :



```
[INFO] Running com.rently.rentlyAPI.exceptions.AuthenticationExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 s - in com.rently.rentlyAPI.exceptions.AuthenticationExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.FileUploadExceptionTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.236 s - in com.rently.rentlyAPI.exceptions.FileUploadExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.ObjectValidationExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.081 s - in com.rently.rentlyAPI.exceptions.ObjectValidationExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.OperationNonPermittedExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.rently.rentlyAPI.exceptions.OperationNonPermittedExceptionTes
[INFO] Running com.rently.rentlyAPI.handlers.ExceptionRepresentationTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 s - in com.rently.rentlyAPI.handlers.ExceptionRepresentationTest
[INFO] Running com.rently.rentlyAPI.handlers.GlobalExceptionHandlerTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.169 s - in com.rently.rentlyAPI.handlers.GlobalExceptionHandlerTest
[INFO] Running com.rently.rentlyAPI.security.config.audit.ApplicationAuditAwareTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.064 s - in com.rently.rentlyAPI.security.config.audit.ApplicationAuditAwareT
[INFO] 
[INFO] Results:
[INFO]
[INFO] Tests run: 102, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Sprint 3

Backend - JUnit & Mockito



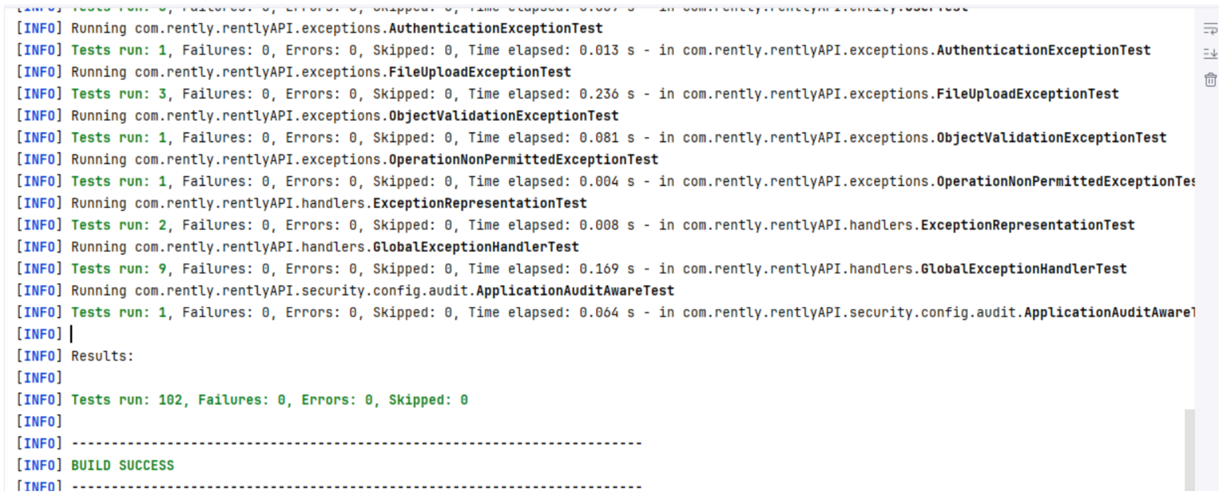
Current scope: all classes

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	80.2% (65/81)	84.4% (331/394)	82.05% (750/914)

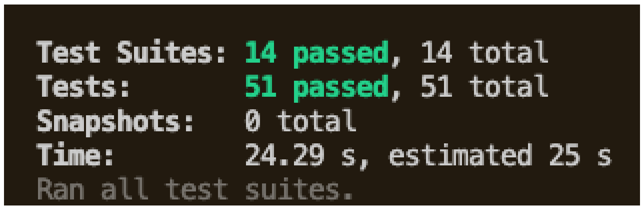
Coverage Breakdown

Package	Class, %	Method, %	Line, %
com.rently.rentlyAPI	0% (0/1)	0% (0/2)	0% (0/2)
com.rently.rentlyAPI.config	0% (0/1)	0% (0/1)	0% (0/1)
com.rently.rentlyAPI.controller	75% (3/4)	92% (23/25)	87.5% (28/32)
com.rently.rentlyAPI.controller.auth	0% (0/1)	0% (0/4)	100% (4/4)
com.rently.rentlyAPI.dto	86.6% (13/15)	85.8% (73/85)	87.34% (138/158)
com.rently.rentlyAPI.dto.auth	12.5% (1/8)	11.1% (4/36)	11.1% (4/36)
com.rently.rentlyAPI.entity	94.11% (16/17)	85.5% (68/80)	84.42% (75/89)
com.rently.rentlyAPI.entity.auth	0% (0/7)	89.7% (35/39)	0% (0/44)
com.rently.rentlyAPI.entity.enums	0% (0/1)	0% (0/2)	0% (0/4)
com.rently.rentlyAPI.exceptions	0% (0/4)	0% (0/9)	0% (0/11)
com.rently.rentlyAPI.handlers	0% (0/3)	100% (16/16)	100% (47/47)
com.rently.rentlyAPI.security	0% (0/2)	0% (0/9)	100% (37/37)
com.rently.rentlyAPI.security.config	0% (0/2)	0% (0/13)	0% (0/47)
com.rently.rentlyAPI.security.config.audit	0% (0/1)	0% (0/2)	0% (0/8)
com.rently.rentlyAPI.security.filter	0% (0/1)	0% (0/2)	0% (0/20)
com.rently.rentlyAPI.security.utils	0% (0/2)	0% (0/16)	0% (0/70)
com.rently.rentlyAPI.services.auth	0% (0/4)	86.6% (13/15)	86.6% (78/90)
com.rently.rentlyAPI.services.impl	16.7% (1/6)	88.8% (32/36)	90.2% (185/205)
com.rently.rentlyAPI.validators	0% (0/1)	0% (0/2)	0% (0/9)

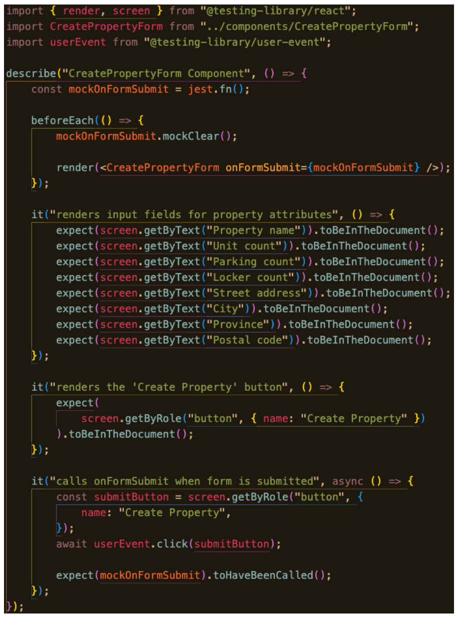


```
[INFO] Running com.rently.rentlyAPI.exceptions.AuthenticationExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 s - in com.rently.rentlyAPI.exceptions.AuthenticationExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.FileUploadExceptionTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.236 s - in com.rently.rentlyAPI.exceptions.FileUploadExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.ObjectValidationExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.081 s - in com.rently.rentlyAPI.exceptions.ObjectValidationExceptionTest
[INFO] Running com.rently.rentlyAPI.exceptions.OperationNonPermittedExceptionTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.rently.rentlyAPI.exceptions.OperationNonPermittedExceptionTes
[INFO] Running com.rently.rentlyAPI.handlers.ExceptionRepresentationTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 s - in com.rently.rentlyAPI.handlers.ExceptionRepresentationTest
[INFO] Running com.rently.rentlyAPI.handlers.GlobalExceptionHandlerTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.169 s - in com.rently.rentlyAPI.handlers.GlobalExceptionHandlerTest
[INFO] Running com.rently.rentlyAPI.security.config.audit.ApplicationAuditAwareTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.064 s - in com.rently.rentlyAPI.security.config.audit.ApplicationAuditAwareT
[INFO] 
[INFO] Results:
[INFO] 
[INFO] Tests run: 102, Failures: 0, Errors: 0, Skipped: 0
[INFO] 
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

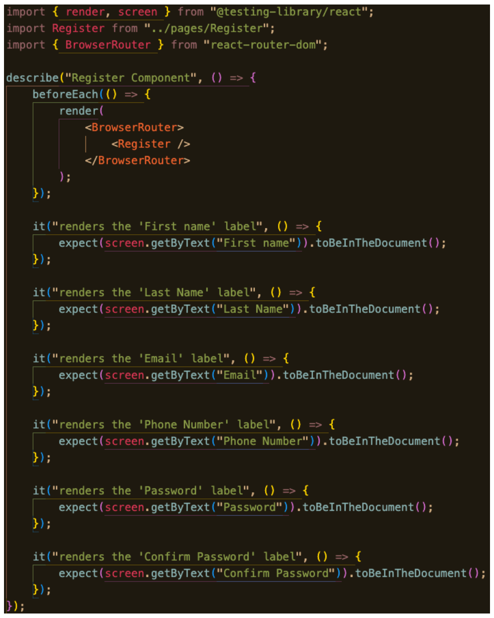
Frontend - Jest



```
Test Suites: 14 passed, 14 total
Tests:       51 passed, 51 total
Snapshots:   0 total
Time:        24.29 s, estimated 25 s
Ran all test suites.
```



```
import { render, screen } from "@testing-library/react";
import CreatePropertyForm from "../components/CreatePropertyForm";
import userEvent from "@testing-library/user-event";

describe("CreatePropertyForm Component", () => {
    const mockOnFormSubmit = jest.fn();

    beforeEach(() => {
        mockOnFormSubmit.mockClear();

        render(<CreatePropertyForm onFormSubmit={mockOnFormSubmit} />);
    });

    it("renders input fields for property attributes", () => {
        expect(screen.getByText("Property name")).toBeInTheDocument();
        expect(screen.getByText("Unit count")).toBeInTheDocument();
        expect(screen.getByText("Parking count")).toBeInTheDocument();
        expect(screen.getByText("Locker count")).toBeInTheDocument();
        expect(screen.getByText("Street address")).toBeInTheDocument();
        expect(screen.getByText("City")).toBeInTheDocument();
        expect(screen.getByText("Province")).toBeInTheDocument();
        expect(screen.getByText("Postal code")).toBeInTheDocument();
    });

    it("renders the 'Create Property' button", () => {
        expect(
            screen.getByRole("button", { name: "Create Property" })
        ).toBeInTheDocument();
    });

    it("calls onFormSubmit when form is submitted", async () => {
        const submitButton = screen.getByRole("button", {
            name: "Create Property",
        });
        await userEvent.click(submitButton);

        expect(mockOnFormSubmit).toHaveBeenCalled();
    });
});
```



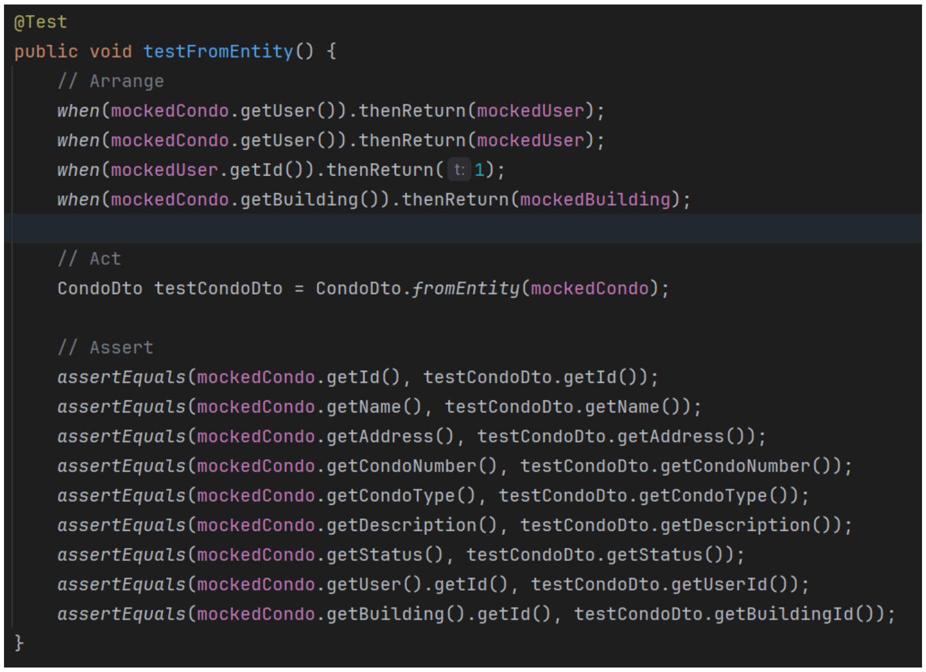
```
import { render, screen } from "@testing-library/react";
import Register from "../pages/Register";
import { BrowserRouter } from "react-router-dom";

describe("Register Component", () => {
    beforeEach(() => {
        render(
            <BrowserRouter>
                <Register />
            </BrowserRouter>
        );
    });

    it("renders the 'First name' label", () => {
        expect(screen.getByText("First name")).toBeInTheDocument();
    });

    it("renders the 'Last Name' label", () => {
        expect(screen.getByText("Last Name")).toBeInTheDocument();
    });

    it("renders the 'Email' label", () => {
        expect(screen.getByText("Email")).toBeInTheDocument();
    });

    it("renders the 'Phone Number' label", () => {
        expect(screen.getByText("Phone Number")).toBeInTheDocument();
    });

    it("renders the 'Password' label", () => {
        expect(screen.getByText("Password")).toBeInTheDocument();
    });

    it("renders the 'Confirm Password' label", () => {
        expect(screen.getByText("Confirm Password")).toBeInTheDocument();
    });
});
```

Moving Forward/Sprint 4

For sprint 4, the team will continue to work with JUnit and Jest for unit testing on the system. As mentioned, SonarQube will also be implemented into the pipeline to provide static analysis on the system prior to deployment. This is an important task to complete to ensure proper deployment of the system, without this there may be possible build issues that are not encountered until demo time to the stakeholders. These processes and tools will help immensely with all the moving parts making the project a reality.

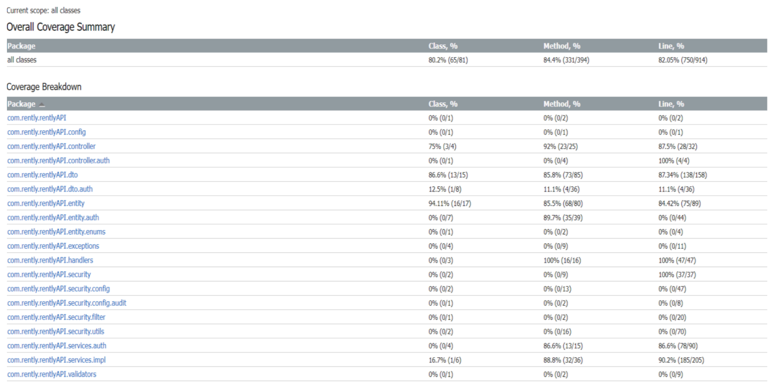
Sprint 4

Backend testing



```java
@Test
public void testFromEntity() {
    // Arrange
    when(mockedCondo.getUser()).thenReturn(mockedUser);
    when(mockedCondo.getUser()).thenReturn(mockedUser);
    when(mockedUser.getId()).thenReturn( t: 1);
    when(mockedCondo.getBuilding()).thenReturn(mockedBuilding);

    // Act
    CondoDto testCondoDto = CondoDto.fromEntity(mockedCondo);

    // Assert
    assertEquals(mockedCondo.getId(), testCondoDto.getId());
    assertEquals(mockedCondo.getName(), testCondoDto.getName());
    assertEquals(mockedCondo.getAddress(), testCondoDto.getAddress());
    assertEquals(mockedCondo.getCondoNumber(), testCondoDto.getCondoNumber());
    assertEquals(mockedCondo.getCondoType(), testCondoDto.getCondoType());
    assertEquals(mockedCondo.getDescription(), testCondoDto.getDescription());
    assertEquals(mockedCondo.getStatus(), testCondoDto.getStatus());
    assertEquals(mockedCondo.getUser().getId(), testCondoDto.getUserId());
    assertEquals(mockedCondo.getBuilding().getId(), testCondoDto.getBuildingId());
}
```

Figure: Example test case, showcasing Arrange, Act and Assert sections of the unit test



Current scope: all classes

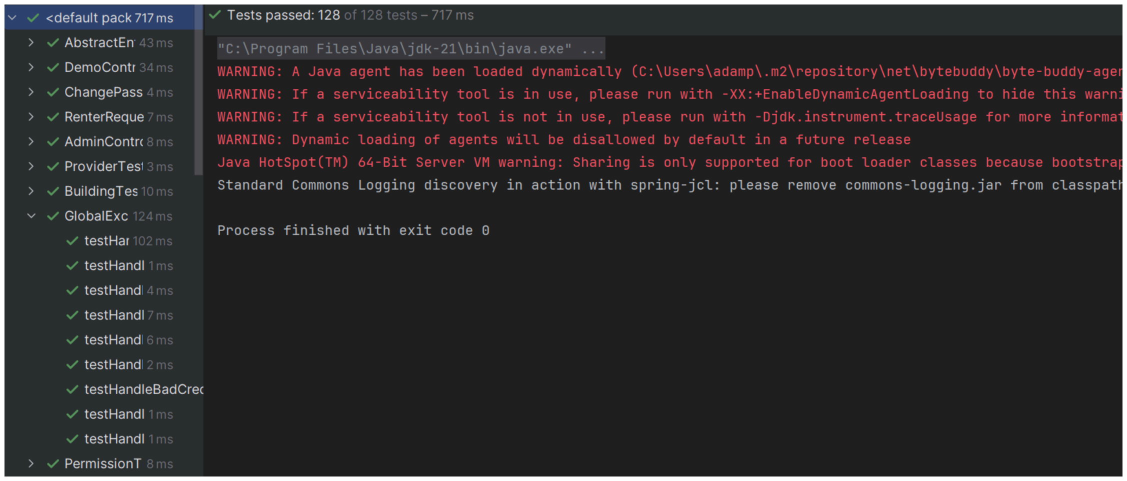
Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	80.2% (65/81)	84.4% (331/394)	82.05% (750/914)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
com.rently.rentlyAPI	0% (0/1)	0% (0/2)	0% (0/2)
com.rently.rentlyAPI.config	0% (0/1)	0% (0/1)	0% (0/1)
com.rently.rentlyAPI.controller	75% (3/4)	92% (23/25)	87.5% (28/32)
com.rently.rentlyAPI.controller.auth	0% (0/1)	0% (0/4)	100% (4/4)
com.rently.rentlyAPI.dto	86.6% (13/15)	85.8% (73/85)	87.34% (138/158)
com.rently.rentlyAPI.dto.auth	12.5% (1/8)	11.1% (4/36)	11.1% (4/36)
com.rently.rentlyAPI.entity	94.11% (16/17)	85.5% (68/80)	84.42% (75/89)
com.rently.rentlyAPI.entity.auth	0% (0/7)	89.7% (35/39)	0% (0/44)
com.rently.rentlyAPI.entity.enums	0% (0/1)	0% (0/2)	0% (0/4)
com.rently.rentlyAPI.exceptions	0% (0/4)	0% (0/9)	0% (0/11)
com.rently.rentlyAPI.handlers	0% (0/3)	100% (16/16)	100% (47/47)
com.rently.rentlyAPI.security	0% (0/2)	0% (0/9)	100% (37/37)
com.rently.rentlyAPI.security.config	0% (0/2)	0% (0/13)	0% (0/47)
com.rently.rentlyAPI.security.config.audit	0% (0/1)	0% (0/2)	0% (0/8)
com.rently.rentlyAPI.security.filter	0% (0/1)	0% (0/2)	0% (0/20)
com.rently.rentlyAPI.security.utils	0% (0/2)	0% (0/16)	0% (0/70)
com.rently.rentlyAPI.services.auth	0% (0/4)	86.6% (13/15)	86.6% (78/90)
com.rently.rentlyAPI.services.impl	16.7% (1/6)	88.8% (32/36)	90.2% (185/205)
com.rently.rentlyAPI.validators	0% (0/1)	0% (0/2)	0% (0/9)

Figure: Run coverage of JUnit tests

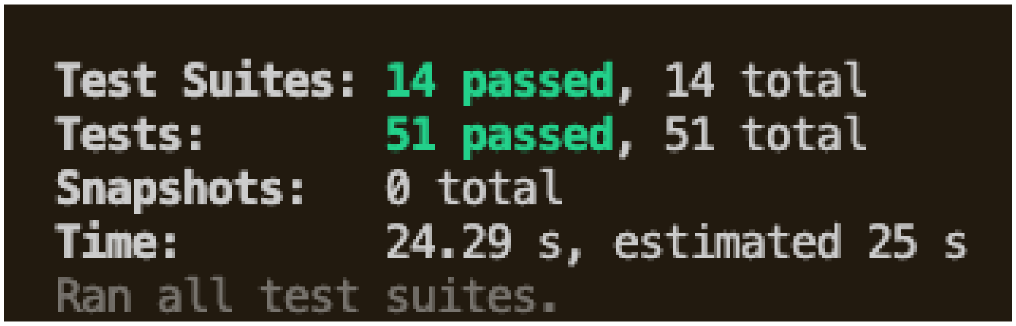


```
<default pack 717 ms
  AbstractEn 43 ms
  DemoContr 34 ms
  ChangePass 4 ms
  RenterReque 7 ms
  AdminContr 8 ms
  ProviderTes 3 ms
  BuildingTes 10 ms
  GlobalExc 124 ms
    testHar 102 ms
    testHandl 1 ms
    testHandl 4 ms
    testHandl 7 ms
    testHand 6 ms
    testHand 2 ms
    testHandleBadCre
    testHandl 1 ms
    testHandl 1 ms
  PermissionT 8 ms
```

```
Tests passed: 128 of 128 tests – 717 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...
WARNING: A Java agent has been loaded dynamically (C:\Users\adamp\.m2\repository\net\bytebuddy\byte-buddy-age
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warni
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more informat
WARNING: Dynamic loading of agents will be disallowed by default in a future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath

Process finished with exit code 0
```

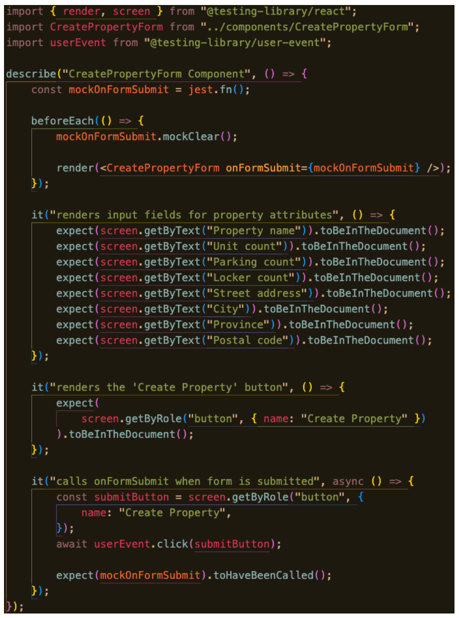
Figure : Run JUnit tests in IntelliJ, showcasing 128 of 128 tests passing

Jest

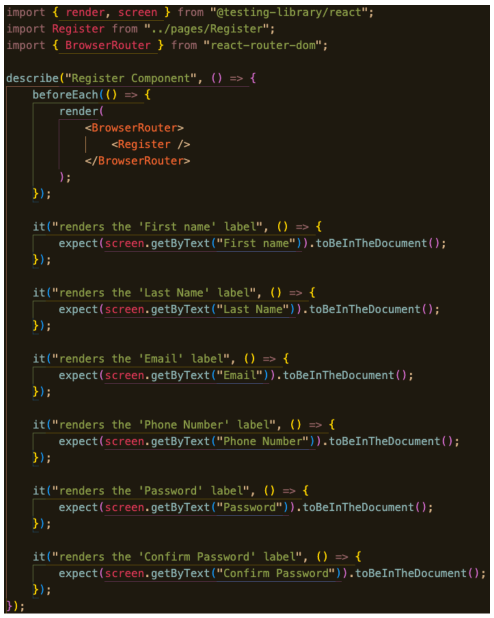


```
Test Suites: 14 passed, 14 total
Tests:       51 passed, 51 total
Snapshots:   0 total
Time:        24.29 s, estimated 25 s
Ran all test suites.
```

Figure: Run of Jest tests developed



```
import { render, screen } from "@testing-library/react";
import CreatePropertyForm from "../components/CreatePropertyForm";
import userEvent from "@testing-library/user-event";

describe("CreatePropertyForm Component", () => {
    const mockOnFormSubmit = jest.fn();

    beforeEach(() => {
        mockOnFormSubmit.mockClear();

        render(<CreatePropertyForm onFormSubmit={mockOnFormSubmit} />);
    });

    it("renders input fields for property attributes", () => {
        expect(screen.getByText("Property name")).toBeInTheDocument();
        expect(screen.getByText("Unit count")).toBeInTheDocument();
        expect(screen.getByText("Parking count")).toBeInTheDocument();
        expect(screen.getByText("Locker count")).toBeInTheDocument();
        expect(screen.getByText("Street address")).toBeInTheDocument();
        expect(screen.getByText("City")).toBeInTheDocument();
        expect(screen.getByText("Province")).toBeInTheDocument();
        expect(screen.getByText("Postal code")).toBeInTheDocument();
    });

    it("renders the 'Create Property' button", () => {
        expect(
            screen.getByRole("button", { name: "Create Property" })
        ).toBeInTheDocument();
    });

    it("calls onFormSubmit when form is submitted", async () => {
        const submitButton = screen.getByRole("button", {
            name: "Create Property",
        });
        await userEvent.click(submitButton);

        expect(mockOnFormSubmit).toHaveBeenCalled();
    });
});
```



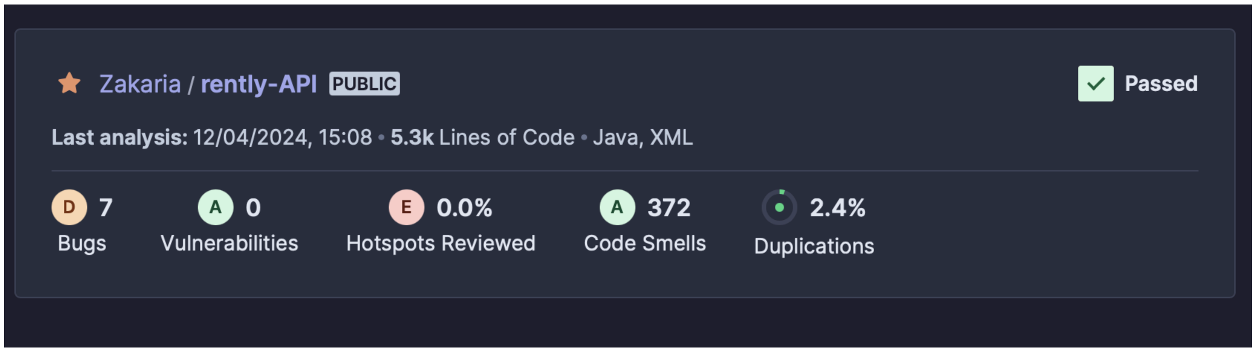
```
import { render, screen } from "@testing-library/react";
import Register from "../pages/Register";
import { BrowserRouter } from "react-router-dom";

describe("Register Component", () => {
    beforeEach(() => {
        render(
            <BrowserRouter>
                <Register />
            </BrowserRouter>
        );
    });

    it("renders the 'First name' label", () => {
        expect(screen.getByText("First name")).toBeInTheDocument();
    });

    it("renders the 'Last Name' label", () => {
        expect(screen.getByText("Last Name")).toBeInTheDocument();
    });

    it("renders the 'Email' label", () => {
        expect(screen.getByText("Email")).toBeInTheDocument();
    });

    it("renders the 'Phone Number' label", () => {
        expect(screen.getByText("Phone Number")).toBeInTheDocument();
    });

    it("renders the 'Password' label", () => {
        expect(screen.getByText("Password")).toBeInTheDocument();
    });

    it("renders the 'Confirm Password' label", () => {
        expect(screen.getByText("Confirm Password")).toBeInTheDocument();
    });
});
```

Figure: Example of test built using Jest

Figur: Another example of test built using Jest

SonarCloud

SonarCloud was integrated in sprint 4 to work with the backend to provide static code analysis of the system as it changes. This provides active feedback on development efforts and ensures that each pull request made does not cause build issues in the system. This can also provide acceptance test standards. The scan is done manually when a specific maven command is run. It is done manually so that there is no need to do it in a pipeline and build the entire project as it increases the complexity of the task as well as increasing the time before getting results.



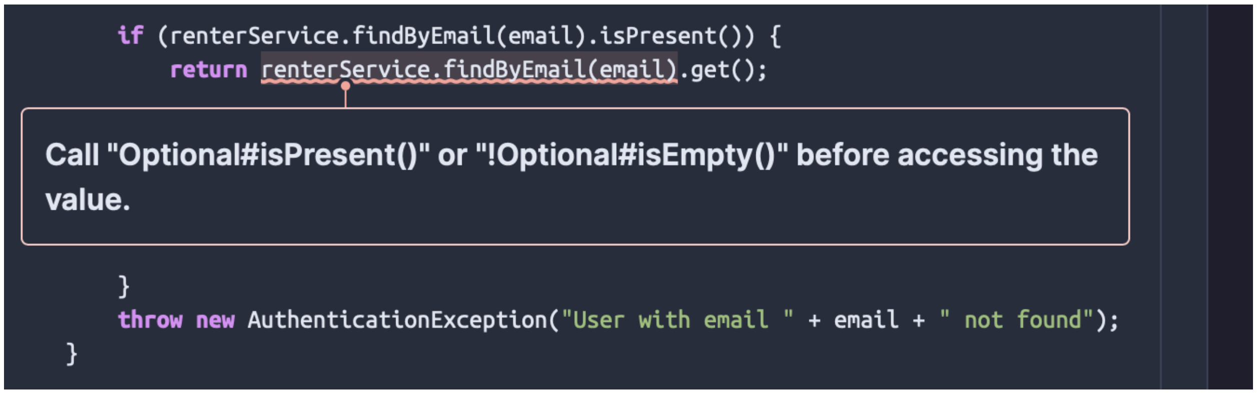
Zakaria / rently-API PUBLIC

Passed

Last analysis: 12/04/2024, 15:08 • 5.3k Lines of Code • Java, XML

D 7	A 0	E 0.0%	A 372	2.4%
Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Duplications

The above image shows the results of a scan. It shows bugs as well as code smells. When diving deeper into the bugs, we notice that there are limitations to what Sonar can do in terms of understanding the code. For example, see the image below:

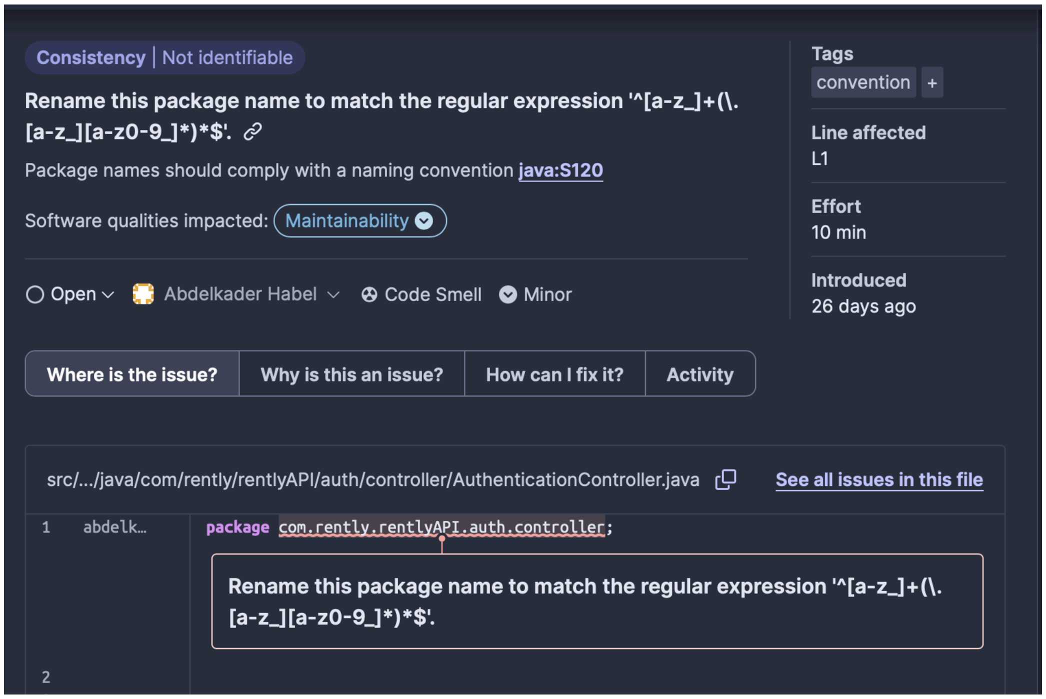


```
    if (renterService.findByEmail(email).isPresent()) {
        return renterService.findByEmail(email).get();
```

Call "Optional#isPresent()" or "!Optional#isEmpty()" before accessing the value.

```
    }
    throw new AuthenticationException("User with email " + email + " not found");
}
```

The bug says that we need to call isPresent() or isEmpty() on an Optional<> object before calling get(). The problem with the “bug” is it does not see the check is being done in the if statement when checking if there is a renter with the given email. Therefore, this bug is not in fact a bug, and thus we choose to ignore it. Similarly for the code smells, a lot of them are misidentifications. Indeed, it is not aware of Spring Boot naming conventions so there are many times when it identifies a smell that shouldn't be there, such as in the below image:


Consistency | Not identifiable

Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.

Package names should comply with a naming convention java:S120

Software qualities impacted: Maintainability

Open · Abdelkader Habel · Code Smell · Minor

Tags: convention

Line affected: L1

Effort: 10 min

Introduced: 26 days ago

src/.../java/com/rently/rentlyAPI/auth/controller/AuthenticationController.java

```
abdelk…  package com.rently.rentlyAPI.auth.controller;
```

Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.

The name of the package is correct, but Sonar identifies it as wrong.

Metrics/Success Criteria

The system will be defined as adequately tested when the coverage using IntelliJ/JUnit's testing coverage software passes with a coverage above 80%. Doing this allows for the team to forgo that last few percent of coverage, something that can be difficult to achieve in general, especially for a codebase with such a short-lived life-cycle.

Current overall coverage summary : 80%+ code coverage

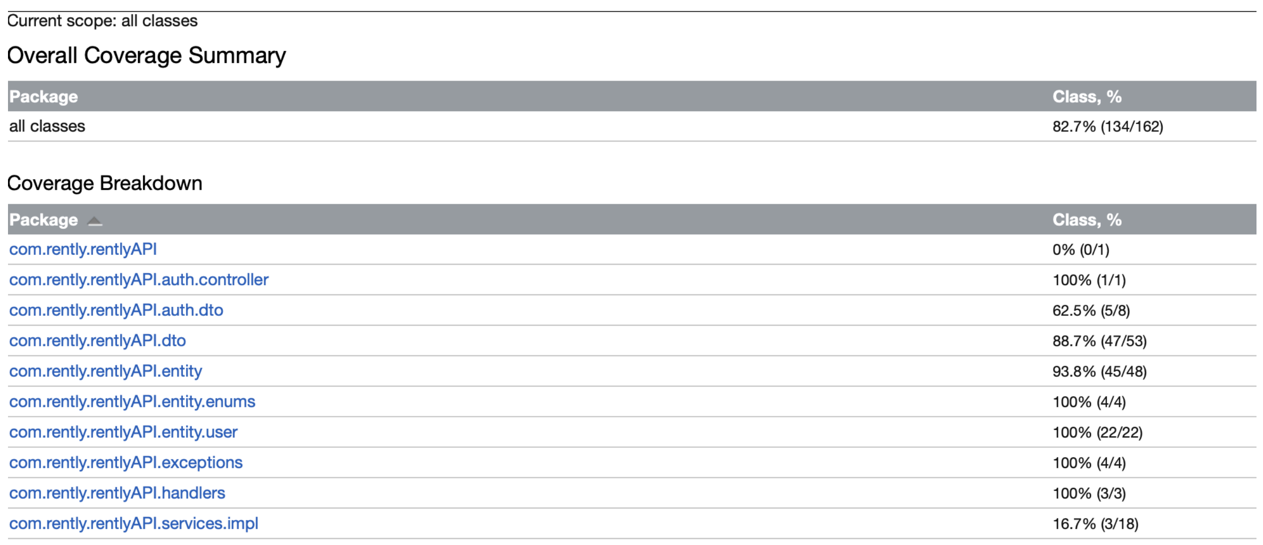
Moving Forward/Sprint 5

For sprint 5, the team plans on continuing work in unit tests for the frontend and the backend, adding tests for each added backend functionality. This will increase system reliability and ensure that the system is built correctly and bug-free. Although with final exam season approaching, there is a higher likelihood of final coding steps being left until the end to ensure a correct schedule and timeline in the area of post-secondary graduation.\

Sprint 5

Backend testing

The results are >80%



Current scope: all classes

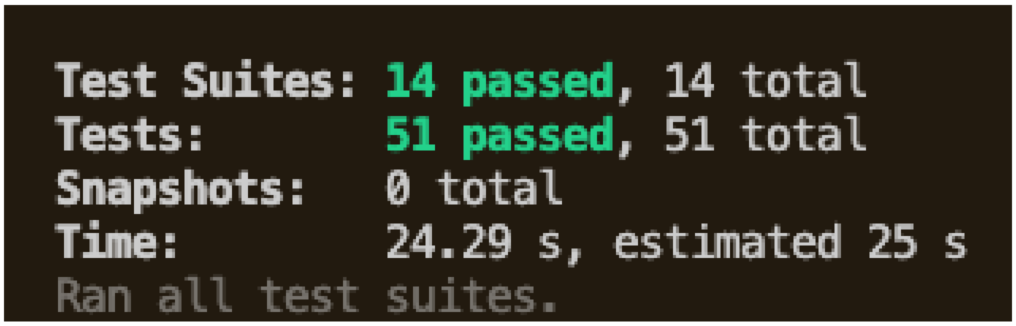
Overall Coverage Summary

Package	Class, %
all classes	82.7% (134/162)

Coverage Breakdown

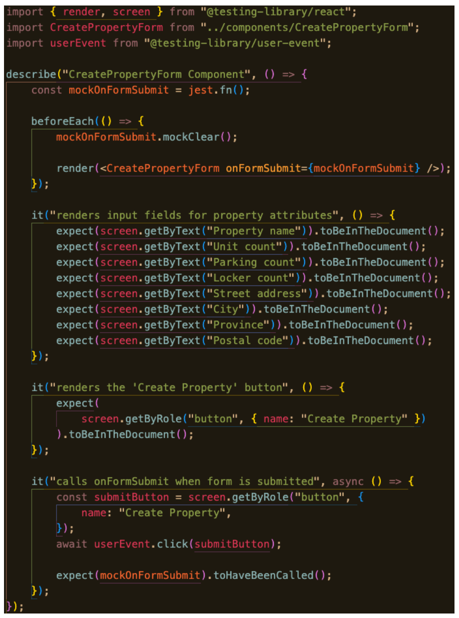
Package	Class, %
com.rently.rentlyAPI	0% (0/1)
com.rently.rentlyAPI.auth.controller	100% (1/1)
com.rently.rentlyAPI.auth.dto	62.5% (5/8)
com.rently.rentlyAPI.dto	88.7% (47/53)
com.rently.rentlyAPI.entity	93.8% (45/48)
com.rently.rentlyAPI.entity.enums	100% (4/4)
com.rently.rentlyAPI.entity.user	100% (22/22)
com.rently.rentlyAPI.exceptions	100% (4/4)
com.rently.rentlyAPI.handlers	100% (3/3)
com.rently.rentlyAPI.services.impl	16.7% (3/18)

Jest (frontend testing)



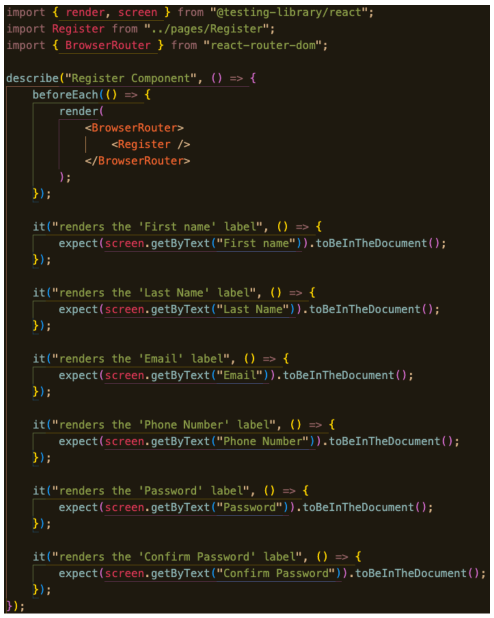
```
Test Suites: 14 passed, 14 total
Tests:       51 passed, 51 total
Snapshots:   0 total
Time:        24.29 s, estimated 25 s
Ran all test suites.
```

Figure: Run of Jest tests developed



```javascript
import { render, screen } from "@testing-library/react";
import CreatePropertyForm from "../components/CreatePropertyForm";
import userEvent from "@testing-library/user-event";

describe("CreatePropertyForm Component", () => {
    const mockOnFormSubmit = jest.fn();

    beforeEach(() => {
        mockOnFormSubmit.mockClear();

        render(<CreatePropertyForm onFormSubmit={mockOnFormSubmit} />);
    });

    it("renders input fields for property attributes", () => {
        expect(screen.getByText("Property name")).toBeInTheDocument();
        expect(screen.getByText("Unit count")).toBeInTheDocument();
        expect(screen.getByText("Parking count")).toBeInTheDocument();
        expect(screen.getByText("Locker count")).toBeInTheDocument();
        expect(screen.getByText("Street address")).toBeInTheDocument();
        expect(screen.getByText("City")).toBeInTheDocument();
        expect(screen.getByText("Province")).toBeInTheDocument();
        expect(screen.getByText("Postal code")).toBeInTheDocument();
    });

    it("renders the 'Create Property' button", () => {
        expect(
            screen.getByRole("button", { name: "Create Property" })
        ).toBeInTheDocument();
    });

    it("calls onFormSubmit when form is submitted", async () => {
        const submitButton = screen.getByRole("button", {
            name: "Create Property",
        });
        await userEvent.click(submitButton);

        expect(mockOnFormSubmit).toHaveBeenCalled();
    });
});
```

Figure: Example of test built using Jest



```javascript
import { render, screen } from "@testing-library/react";
import Register from "../pages/Register";
import { BrowserRouter } from "react-router-dom";

describe("Register Component", () => {
    beforeEach(() => {
        render(
            <BrowserRouter>
                <Register />
            </BrowserRouter>
        );
    });

    it("renders the 'First name' label", () => {
        expect(screen.getByText("First name")).toBeInTheDocument();
    });

    it("renders the 'Last Name' label", () => {
        expect(screen.getByText("Last Name")).toBeInTheDocument();
    });

    it("renders the 'Email' label", () => {
        expect(screen.getByText("Email")).toBeInTheDocument();
    });

    it("renders the 'Phone Number' label", () => {
        expect(screen.getByText("Phone Number")).toBeInTheDocument();
    });

    it("renders the 'Password' label", () => {
        expect(screen.getByText("Password")).toBeInTheDocument();
    });

    it("renders the 'Confirm Password' label", () => {
        expect(screen.getByText("Confirm Password")).toBeInTheDocument();
    });
});
```

Figure: Another example of test built using Jest

Retrospective documents

Sprint 1

Introduction

This postmortem is written in a context where the first sprint and the layout for the whole development of Rently, a condo management system, is completed. The plan for the 5 sprints was devised and some groundwork has been started. Particularly, technologies have been chosen, architecture diagrams have been developed and roles have been separated in the team.

What went wrong

1 - Auth Setup
Initial implementation was not up to the security standards, so we had to reimplement the whole authentication and authorization logic. We realized it had to be changed after we saw the refresh tokens endpoints not behaving in a way we expected it to. Instead of storing the refresh token in the HTTP-only cookies, we were instead exposing it to the user, making auth vulnerable to XSS and CSRF attacks. It was fixed by implementing an HTTP-only mechanism and storing it in the cookies. Although it was fixed in the end, it is a negative point because of the time it took us to fix it, which could have been allocated to other tasks.

2 - Redux setup problem
The Redux setup issue, characterized by an overly complex and inefficient state management system, was primarily due to a lack of optimal use of Redux Toolkit features. This led to performance issues and increased development time. The problem arose from an initial misunderstanding of Redux Toolkit's capabilities, especially around state optimization and asynchronous logic handling. To address this, the team refactored the setup by applying best practices, such as using selector functions and the createSlice and createAsyncThunk utilities. However, a more streamlined solution was found in leveraging React Context, which provided a simpler and more direct way to manage state without the overhead of Redux, significantly improving maintainability and performance. This experience underscored the importance of selecting the right tools for project needs and the benefits of a simpler approach for state management.

3 - Tight timeline
We found the timeline to be very tight to have code implemented, as well as having all the documents ready. We struggled with having all the types of diagrams, some of them we had never worked with before, produced at the same time as we had to handle the JIRA board and the coding. Our good separation of tasks helped us a lot, since everyone was working on something different.

4 - Jira & Confluence Setup

The project's Jira and Confluence setup was initially ineffective, leading to operational inefficiencies. Key issues included default configurations that didn't meet project needs, insufficient training on tool capabilities, and inconsistent adoption across the team.

The setup fell short due to inadequate configuration for the project's specific requirements, lack of comprehensive training for the team, and the absence of a clear strategy for tool adoption.

This resulted in decreased productivity, communication breakdowns, and the perception of resource wastage, as the tools did not deliver the expected improvements in efficiency and collaboration.

Actions taken to address these issues included customizing the configuration of both tools, conducting detailed training sessions, and enforcing usage policies to ensure consistent application across the project.

Better initial planning, including a detailed needs assessment, consultation with tool experts, and regular reviews for adjustments, would have significantly enhanced the effectiveness of Jira and Confluence in the project workflow.

5 - Inconsistent Code Review Processes

The project suffered from inconsistent code review processes, leading to uneven code quality and delays in feature deployment. Without a standardized approach, code reviews varied significantly in thoroughness and effectiveness, allowing bugs and coding standard violations to slip through.

The inconsistency arose from a lack of formalized code review guidelines and training. Reviewers had varying levels of experience and different perspectives on what constituted quality code, and there was no system in place to ensure uniformity in the review process.

This led to several negative outcomes, including increased bug rates in production, longer development cycles due to the need for additional rounds of revision and testing, and frustration among team members over the perceived arbitrariness of the review process.

To address the problem, the team implemented a set of standardized code review guidelines and provided training sessions on best practices. Additionally, a code review checklist was introduced to ensure that all reviews covered essential quality criteria consistently.

A more structured approach from the start, with clear code review standards and regular reviewer training, could have prevented these issues. Integrating automated code quality tools into the development workflow might also have helped maintain consistency and reduce the workload on human reviewers.

What went right

1 - Team role separation

The team was separated appropriately for the project, according to each member's strength. This meant that we were able to all start working on what we need to do without having people be forced to learn new technologies in a short time span. What was also good is that we designated leaders to manage the backend and frontend subteams. This eased the separation of tasks, where instead of one person managing 9 people, we had 1 person managing 4-5 people in their subteam.

2 - Technology selection

It was easy for us to choose the technologies, thanks to our familiarity with some frameworks over others. We chose Spring Boot for the backend due to its great ability to separate concerns. Various layers allow us to keep a structured code and produce maintainable code. For the frontend, we chose React, as it is the framework most people know.

3 - Design of the system

We are happy with how we designed our system, most specifically our class and domain diagrams. We were able to apply our knowledge from previous courses to this project and have something developed fast. Our domain model covered all the core aspects of the project and we kept in mind modularity and future changes. It was straightforward and we didn't struggle on it. The only thing that took a bit more reflexion was on assigning responsibilities to the classes. We did not know for certainty whether to assign some responsibilities to the Company or the Property. We made decisions based on how we thought it would be difficult to implement.

4 - JIRA board handling

After the difficulties faced in the setup, we found it is easy to use JIRA once we look past the initial setup. The JIRA board helped us familiarize more with how this environment works. We assigned 2 people to work on the board to keep it clean, and prepared documents to help the other members of the team navigate through JIRA. We found it really helpful to see that we can include the id of a task in a commit or branch name to have it automatically tracked by JIRA. It was impressive to see. We also enjoyed the automations that change the status of tasks based on whether the id is in a commit, a branch or a pull request. For example, the status goes from "In

Progress” -> “In Review” when there is a pull request that has the id of that task. Similarly, the status changes from ”In review” -> “Done” when a pull request is accepted.

Conclusion

The initial sprint of the Rently project, aimed at creating a comprehensive condo management system, has been a period of significant learning and achievement. Despite facing challenges such as security flaws in authentication setup, inefficiencies in Redux configuration, tight timelines, difficulties with Jira & Confluence setup, and inconsistencies in code review processes, the team has made considerable progress.

Key Learnings:

- The importance of security from the start.
- The value of selecting the right development tools.
- The need for effective task management and team training.
- The benefits of consistent code quality checks.

Achievements:

- Efficient team role separation leveraging individual strengths.
- Successful technology selection enhancing backend and frontend development.
- Effective system design ensuring a solid project foundation.
- Improved project management through streamlined JIRA workflows.

Moving Forward:

This sprint has set a solid foundation and direction for the Rently project. The challenges encountered have provided valuable insights, shaping our strategies for improved efficiency and success in future sprints. With a focus on leveraging our strengths and applying the lessons learned, we are well-positioned to advance the development of Rently as a leading condo management solution.

Sprint 2

Introduction

This postmortem is written in a context where the second sprint is completed, with features already functional. We are using Spring Boot in the backend and React in the frontend to develop Rently, our condo management platform. We deviated from our initial plan for sprint 2, and had to implement features different from those we said we would. We also realized at the end of the sprint our approach is not very efficient on the backend side, and that we will have some major refactoring to do in the 3rd sprint. Specifically, we did not implement the class diagram and then from there worked on our features, but rather we chose features and worked on implementing the class diagram as we went. Some things went well, like the team collaboration and a good separation of tasks.

What went wrong

1 - Setting up the AWS S3 bucket to store files

We faced a few problems when setting up the AWS S3 bucket to store files. Firstly, we had issues creating an MAI account, which is a more secure sub account with particular privileges. Then after creating the account, we had to find the appropriate service (S3 bucket) and then find the access key and the secret key. Basically, the setup for this service was problematic and took a lot of time for us to be done with it. After the setup, we had another issue to allow external users to read and write to the storage bucket. There were also a few problems on the backend when trying to use the AWS dependency within Spring Boot. The online tutorial Spring gave was outdated, so we had to use the AWS documentation, which was not very clear.

2 - Problems using online shareable resources

We encountered problems related to online resources that are free but with restrictions. For example, when using Postman to test our endpoints, we made a collection of endpoints with the bodies of requests so that for future tests we just run them without wasting time. We wanted to share the collection with other teammates, but when we reached 5 people in the Postman workspace, we got put into a free trial for 2 weeks and we do not know yet what will happen after. We will likely have to manually share the collections after any endpoint is modified, which is very inconvenient, keeping track of how 9 people are using them. Similarly, we needed a database that would allow us to work collaboratively, but it was hard to find. We decided that for those working on the backend, they would each use their own separate database, so they need to create their account on Neon, and those working on the frontend would use the same database, since they do not need it as extensively as the backend developers. Overall, it was a bit hard to navigate the resources and their limitations.

3 - Implementation of the class diagram

One major problem we realized we had caused was related to the class diagram. We did not start the coding process by implementing the class diagram and then working our way up from there. Instead, we created the basic classes, like condo, building and user, and ignored a lot of other components of the class diagram. For example, lockers and parkings are associated to building and condos, but they appeared nowhere in our code. We thought we could update the structure as we go, but we realized later that it increases the complexity by a lot, due to the amount of refactoring that was required whenever we needed to introduce a new component. We realized our approach was wrong and decided to refactor our code and implement the full diagram to have our structure ready at the beginning of sprint 3.

4 - Understanding of user roles

A problem we faced was about the different roles that are given to Rently users. For example, some members of the backend team understood a specific set of users, while others understood a different set. It was striking when the time came to make endpoints for companies to create buildings and condos. The user class was initially coded with lists of buildings and employees as attributes, to accommodate for company users. The problem with that is that regular users, like owners or renters are given attributes that are irrelevant to them. We basically considered the “company” entity to be a user. We decided that we would change it in the refactoring process at the beginning of sprint 3. This implementation made us realize that we needed to communicate more closely with each other to be sure of how things work for the most crucial parts of the software.

5 - Unclear set of user stories to be implemented

When we made the plan for sprint 2, we had decided on a few user stories to work on during sprint 2. When we started working, we realized that some of those stories needed to be pushed to later sprints because they were dependent on other parts being fully functional. For example, we saw that it does not really make sense to start thinking about common facilities and reservations before buildings, condos and basic operations with them are fully settled. We also had to change a lot of our user stories because we realized they did not cover the full set of functionalities of the project.

What went right

1 - Good frontend development process

The frontend development part was better structured than the backend part. There were not that many changes that were made, only additions to existing code. There also wasn’t a major problem that was encountered that forced the frontend team to rethink their entire process, unlike in the backend. There was close supervision of the subteam by their team leader, which meant that any concerns were quickly addressed.

2 - Clearing up expectations for future sprints
The problems we faced in this sprint allowed us to discuss more seriously how we would structure the rest of the project. We decided how the classes would be coded, their responsibilities and we decided to adhere as closely to our design as we could. This was a good thing because the discussion we had forced everyone to seriously think about the project in practical terms and gave everyone a better understanding of what needs to be done. If we did not face those problems mentioned above, they might have arisen in later sprints.

3 - Development flow
Everyone respected the rules we had set for how we use JIRA and how git actions are performed. We did not face any problems related to major commits not being linked to a branch or being untracked on JIRA. This showed us that the communication at the beginning of sprint 1 related to JIRA was clear and understood by everyone. We know that we will not have any problems related to this in the future, and that we can focus our attention on more time-consuming tasks.

4 - Sprint extension
We greatly benefitted from the unexpected extension to sprint 2 at the end of the sprint. We were able to allocate more time on the documents and really plan how to handle sprint 3 so that we do not face the same problems we did. We realized that because we were being squeezed by the deadline, we were not able to take a step back and look at all the problems we had. We were working and navigating problems at the same time, which was not done in an efficient way. The extension helped us clear everything and reassess our progress to correct ourselves.

Conclusion

Our takeaway from this sprint is that poor communication in certain areas can really quickly derail a project. We experienced it in the backend and took the necessary steps and decisions to correct ourselves in the future sprints. We also realized that staying true to a design is crucial to stay on the right track so that everyone can refer to the same thing and work coherently, instead of each person having a different view of different parts of the system, and then introducing incoherence when everyone’s work is brought together. For future sprints, we now know that the plan needs to be as clear as possible, and very thought out so that the proper stories can be implemented by members of the team.

Sprint 3

Introduction

This postmortem is written in a context where the third sprint is completed, with features already functional. As mentioned in the retrospective for sprint 2, this sprint involved a great amount of refactoring. The entire backend system was redone, going from the authentication services to the persistence layer. The code that was written in the first two sprints did not fully adhere to the design we had made and also did not meet code quality standards, especially in terms of cohesion and coupling.

What went wrong

1 - Immense refactoring challenge
We had to refactor the entire backend code that was written in the first two sprints, which was a very tedious task. There were a lot of problems in terms of code quality and respect of coding conventions that we needed to address. Some classes were doing many different things, and we needed to break them down to reduce the levels of coupling and make them more cohesive.

2 - Problems using SonarCloud
We set up SonarCloud to find code issues and help us improve the quality of our code. We thought the set up was going to be easy and not tedious. We had already used SonarCloud in previous courses and the setup was easy, so our expectations were not met. When we ran the analysis the first time, we realized it did not work on .java classes, and it only takes compiled classes. The compiled classes are not present in the repository, and we did not want to push them. What this means is we need to build the project “online” before running the scan. This forces us to use a CI/CD pipeline, like CircleCI.

3 - Refactoring broke our tests
Something that went very bad was linked to the tests. In doing our refactoring, we broke all the tests that were for the backend. Our top priority was to get the system working to catch up on the activities of the 3rd sprint. Not having these tests makes us vulnerable to problems we might have missed.

4 - One branch held most changes
We initially started working on the common facility feature before we decided to refactor. We made the mistake of making the changes on the same branch as the feature and were not in a position to merge it to the main branch because it would have broken everything, with a long time before we fix the problems. We were forced to always make changes on different branches that diverge from the feature branch instead of the main branch. This meant that for a good

while, the branch that was initially for the feature effectively became our main branch. It was bad because it forced us to finish all our changes before merging to main and starting other features.

What went right

1 - Application of design patterns
We used the facade design pattern, which made our code clearer and made it so that the right services were being called from a few endpoints. For example, when registering an employee or a company admin, the controller calls the user service, which then retrieves the type of user and calls the appropriate constructor. So the data would go to the user service who then calls the employeeService or the adminService. This was done for several functionalities. It allowed us to easily structure the code and makes development easier.

2 - Backend development speed
Since we had the design patterns, it was easy to implement the features once a few of them were done. It was like a template for us to follow and helped us avoid spending too much time thinking about where to place certain functions. Also, following the GRASP patterns gave us a lot of cases of reusability, which in turn saves us a lot of time.

3 - Better use of Postman CI/CD
We also improved our use of Postman. With the growing number of endpoints, it was becoming difficult to perform a specific action without performing a series of other actions beforehand. For example, to create a common facility, a system admin must first be created, then create a company and add to it a system admin, then authenticate the admin, make it create a building and then from there create a common facility. Needless to say it is difficult to test a growing system if all these requests depend on a lot of them. We wrote scripts to help us perform these actions so we do not have to do them.

4 - Smaller number of merge conflicts
Due to the great communication between team members, a more coherent code and a better organization, the merge conflicts were lesser and of small complexity to solve than in the first sprints. This was true even with 3-4 members working on the same branch. This was a great advantage to us because we did not spend time solving the conflicts. It was also a sign for us that our new approach is working.

Conclusion

In conclusion, Sprint 3 had its ups and downs. Despite the tough task of refactoring and some hiccups with SonarCloud and broken tests, we made it through. Using design patterns like the facade pattern really helped clean up our code and speed up development, although we were stuck working from a feature branch that was acting like the main branch in the backend. Postman was a lifesaver for testing our endpoints, especially with complex sequences. Plus, our improved teamwork led to fewer merge conflicts, making everything smoother.

Sprint 4

Introduction

This postmortem is written in a context where the fourth sprint is completed, with features functional. This sprint was generally positive

What went wrong

1 - Nothing significant
Sprint 4 did not have problems that we consider important. There were a few moments where the coding took some time to figure out how it needed to be done, but never did we face a real obstacle. The previous sprints have set us up to perform well in this sprint.

What went right

1 - Backend development speed
Since we had the design patterns, it was easy to implement the features once a few of them were done. It was like a template for us to follow and helped us avoid spending too much time thinking about where to place certain functions. Also, following the GRASP patterns gave us a lot of cases of reusability, which in turn saves us a lot of time.

2 - Better use of Swagger UI
We also improved our use of Swagger. We were not fully using it in the previous sprints, we were only using it to know what endpoints we have. In this sprint, the frontend team was using it more frequently to know what the output of requests looks like. Indeed, the body sent to an endpoint is already defined in our postman suites, but the output is not if a valid request isn't sent. Swagger provides a sample output which helps with implementing the frontend.

3 - Deployment ease

Deploying the project was very easy and straightforward. It took less time and effort than we expected. We used Railways to deploy the backend and Netlify to deploy the frontend. It took less than an hour to deploy and fix the problems that were showing up.

4 - Good development in the frontend
Most of the coding activities in this sprint were frontend related, as a lot of the backend endpoints had to be consumed by the frontend. It was easy as there was great collaboration between developers.

Conclusion

In conclusion, Sprint 4 was very positive. There were no wrong aspects to it and everything was mostly easy. The backend development was very easy as most features were already mostly done. The good practices we took in the previous sprint helped us a lot for sprint 4 when it came to coding. We also used Swagger more seriously which helped the people in the frontend understand better what they are using. It was also very easy to deploy the project and hassle free. Overall, this sprint was very positive.

Sprint 5

Introduction

This postmortem is written in a context where the fifth and final sprint is completed, with features functional. This sprint was generally positive.

What went wrong

1 - API Endpoints that had not been updated in the frontend
One problem we faced was that the website was not fully working when we released it in sprint 4. We only realized that after the end of the sprint and had to look for what it was. More precisely, when registering and logging in, the system was showing error messages even though the given credentials were correct. We looked at the network section of the develop feature on our browser and saw that we were calling a specific endpoint. Then when we looked at the frontend code, we realized we had not updated the endpoint at that level. Changing that fixed the problem, but it took a lot of time to pinpoint.

2 - We were a little late on the frontend development
We realized we had a few things to change and fix in the frontend, a task that was more challenging and required more work than needed in the backend. It was especially bothersome because it was at the end of the project and there was no possibility to move anything to a future sprint. It was not an easy task but we still got it in time.

What went right

1 - Backend development
The work that needed to be done in the backend was minimal, so we did not have to worry much about code there. Most of the work had been done in previous sprints in a fast manner thanks to our approach to the development. We were able to focus on other tasks while the frontend was being completed.

2 - Team collaboration
Given this is the end of the project and it happened during final exams, the collaboration between members needed to be very good, and it was. We held a meeting where we made clear the responsibilities of everyone as well as where we discussed our progress. We fixed a few problems during that meeting and any confusion anyone had was cleared.

3 - Deployment ease
Deploying the project was very easy and straightforward. In sprint 4, we deployed the website on railways and netlify. The link to the website we had was a randomly generated one and we needed to change it to something recognizable and related to our platform. We thought it would be complicated to rename it and would probably require us to redeploy and play around a lot in the settings, but none of that happened, we just needed to update a field in netlify settings.

Conclusion

In conclusion, Sprint 5 was overall positive. We faced some problems related to endpoints and status of the frontend code, but it was resolved well enough thanks to the good collaboration between everyone. Most delays were in the frontend because of the supplemental amount of work there is to do compared to the backend. However we also had a few good points in this sprint, like the backend development, which was fast and easy. The collaboration and communication between team members was also very good and helped us get the project through completion. Lastly, the ease of deployment was a very good advantage to us. It confirmed to us that we chose the right services to host our backend and frontend, as opposed to other services that would have required much more configurations.

User stories backlog documents

The user stories backlog here are as they were submitted in their respective sprints. We learned from our errors and updated our USB in sprint 4-5.

Sprint 2 USB

REN-4 As a user I want to login to the system. 30 USP

- Setup an authentication system with JWT and basic roles
- Setup frontend connection to AUTH
- Create authentication unit tests
- Implement OAuth2 with Google

REN-53 As a public user, I want to link my profile to my unit to have access to the services associated to my unit (become an occupant). 8 USP

- BE implementation of key generation per condo for management company
- BE implement key activation for user to become occupant or renter
- FE implementation of the related forms for activation keys

REN-54 As a public user, I want to create a profile to have access to the website 18 USP

- FE Connect setting user profile page to the developed endpoints
- FE Make a profile page
- BE Setup image storage system for profile pictures for public users (S3 Bucket/Flickr)
- BE Add relevant attributes to user class and implement relevant services

REN-55 As an owner I want to have a dashboard of my properties to see everything related to them, like fees, status of requests and more. 16 USP

- BE Implement the relevant services and endpoints
- FE Implement user dashboard and connect it to the endpoints

REN-86 As a management company, I want to add, modify and delete my buildings. 13 USP

- FE Create form related to adding new buildings
- FE Make form to modify building
- BE Implement related endpoints for adding a building
- BE Implement related endpoints to modify/delete buildings

REN-91 As a management company, I want to add, modify and delete my condos in my buildings. 13 USP

- BE Implement related endpoints for adding condos
- BE Implement related endpoints to modify/delete condos
- FE Create form related to adding condos in a building
- FE Make form to modify condo

Sprint 3 USB

Green: Acceptance tests pass/Minor changes needed but functional feature
Yellow: User stories in sprint 4
Blue: User stories in sprint 5

- REN-78 As a public user, I want to create an account and login
- REN-78 As a system administrator, I want to create a company
- REN-78 As a system administrator, I want to create company admins and link them to a company
- REN-78 As a company administrator, I want to login
- REN-78 As a company administrator, I want to register employees
- REN-78 As an employee, I want to login
- REN-53 As a public user, I want to use an activation key to become an owner/renter
- REN-86 As a company administrator, I want to create a building
- REN-91 As a company administrator, I want to add condos to the buildings
- REN-58 As a company administrator, I want to set up common facilities in my buildings for users to reserve
- REN-59 As an owner/renter, I want to make a reservation on a common facility in a calendar-like interface
- REN-78 As a company administrator, I want to send a registration key to public users to link them to a unit
- REN-54 As an owner/renter, I want to create a profile
- REN-55 As an owner, I want to see a dashboard of my properties to see everything relating to them, like fees and status of requests.
- REN-60 As an owner/renter, I want to see availabilities of common facilities
- REN-134 As a company administrator, I want to add lockers and parking to my building
- REN-61 As a company administrator, I want to upload condo files for my properties
- REN-135 As a company administrator, I want to enter condo fees to units
- REN-63 As a company administrator, I want to set different roles for my employees so they can do the work they are supposed to do
- REN-64 As a company administrator, I want to see the status of all the requests owners made.

- REN-66 As an owner, I want to make special requests to be treated by employees
- REN-67 As an owner, I want to see the status of my requests in a notification page
- REN-68 As an employee, I want to see the owner requests that are assigned to me
- REN-69 As an employee, I want to see the status of my assigned requests on a notifications page
- REN-136 As an employee, I want to update assignments I am working on, like changing the status or adding comments
- REN-62 As a company administrator, I want to enter operational costs for all operations made to know my expenses and budget
- REN-70 As a company administrator, I want to have access to a financial annual report to know what my expenses/budget were
- REN-65 As an owner, I want to access the condo files my management company made available for all units in my building

Sprint 4 USB

Green: DONE
Blue: TODO
Mauve: IN PROGRESS

User Story	JIRA ID	USP	Priority	Status
As a public user, I want to create an account and login	REN-78	5	HIGH	DONE
As a system administrator, I want to create a company	REN-78	4	MEDIUM	DONE
As a system administrator, I want to create company admins and link them to a company	REN-78	5	MEDIUM	DONE
As a company administrator, I want to login	REN-78	3	HIGH	DONE
As a company administrator, I want to register employees	REN-78	5	HIGH	DONE

As an employee, I want to login	REN-78	3	HIGH	DONE
As a public user, I want to use an activation key to become an owner/renter	REN-53	5	HIGH	DONE
As a company administrator, I want to create a building	REN-86	5	HIGH	DONE
As a company administrator, I want to add condos to the buildings	REN-91	3	HIGH	DONE
As a company administrator, I want to set up common facilities in my buildings for users to reserve	REN-58	3	LOW	DONE
As an owner/renter, I want to make a reservation on a common facility in a calendar-like interface	REN-59	6	MEDIUM	DONE
As a company administrator, I want to send a registration key to public users to link them to a unit	REN-78	6	HIGH	DONE
As an owner/renter, I want to create a profile	REN-54	5	HIGH	DONE
As an owner, I want to see a dashboard of my properties to see everything relating to them, like fees and status of requests	REN-55	4	MEDIUM	DONE
As an owner/renter, I want to see availabilities of common facilities	REN-60	6	MEDIUM	DONE

As an employee, I want to update assignments I am working on, like changing the status or adding comments	REN-136	5	MEDIUM	DONE
As a company administrator, I want to have access to a financial annual report to know what my expenses/budget were	REN-70	5	MEDIUM	IN PROGRESS
As a company administrator, I want to enter operational costs for all operations made to know my expenses and budget	REN-62	4	LOW	TODO
As an owner, I want to access the condo files my management company made available for all units in my building	REN-65	4	LOW	IN PROGRESS
As a company administrator, I want to add lockers and parking to my building	REN-134	4	LOW	IN PROGRESS
As a company administrator, I want to upload condo files for my properties	REN-61	7	High	IN PROGRESS
As a company administrator, I want to enter condo fees to units	REN-135	5	High	DONE
As a company administrator, I want to set different roles for my employees so they can do the work they are supposed to do	REN-63	5	High	DONE
As a company administrator, I want to see the status of all the requests owners made.	REN-64	4	Medium	DONE

As an owner, I want to make special requests to be treated by employees	REN-66	4	Medium	DONE
As an owner, I want to see the status of my requests in a notification page	REN-67	5	High	DONE
As an employee, I want to see the owner requests that are assigned to me	REN-68	6	High	DONE
As an employee, I want to see the status of my assigned requests on a notifications page	REN-69	4	Medium	DONE
Total USP		130		

Sprint 5 USB

Green: DONE

User Story	JIRA ID	USP	Priority	Status
As a public user, I want to create an account and login	REN-78	5	HIGH	DONE
As a system administrator, I want to create a company	REN-78	4	MEDIUM	DONE
As a system administrator, I want to create company admins and link them to a company	REN-78	5	MEDIUM	DONE
As a company administrator, I want to login	REN-78	3	HIGH	DONE
As a company administrator, I want to register employees	REN-78	5	HIGH	DONE

As an employee, I want to login	REN-78	3	HIGH	DONE
As a public user, I want to use an activation key to become an owner/renter	REN-53	5	HIGH	DONE
As a company administrator, I want to create a building	REN-86	5	HIGH	DONE
As a company administrator, I want to add condos to the buildings	REN-91	3	HIGH	DONE
As a company administrator, I want to set up common facilities in my buildings for users to reserve	REN-58	3	LOW	DONE
As an owner/renter, I want to make a reservation on a common facility in a calendar-like interface	REN-59	6	MEDIUM	DONE
As a company administrator, I want to send a registration key to public users to link them to a unit	REN-78	6	HIGH	DONE
As an owner/renter, I want to create a profile	REN-54	5	HIGH	DONE
As an owner, I want to see a dashboard of my properties to see everything relating to them, like fees and status of requests	REN-55	4	MEDIUM	DONE
As an owner/renter, I want to see availabilities of common facilities	REN-60	6	MEDIUM	DONE

As an employee, I want to update assignments I am working on, like changing the status or adding comments	REN-136	5	MEDIUM	DONE
As a company administrator, I want to have access to a financial annual report to know what my expenses/budget were	REN-70	5	MEDIUM	DONE
As a company administrator, I want to enter operational costs for all operations made to know my expenses and budget	REN-62	4	LOW	DONE
As an owner, I want to access the condo files my management company made available for all units in my building	REN-65	4	LOW	DONE
As a company administrator, I want to add lockers and parking to my building	REN-134	4	LOW	DONE
As a company administrator, I want to upload condo files for my properties	REN-61	7	High	DONE
As a company administrator, I want to enter condo fees to units	REN-135	5	High	DONE
As a company administrator, I want to set different roles for my employees so they can do the work they are supposed to do	REN-63	5	High	DONE
As a company administrator, I want to see the status of all the requests owners made.	REN-64	4	Medium	DONE

As an owner, I want to make special requests to be treated by employees	REN-66	4	Medium	DONE
As an owner, I want to see the status of my requests in a notification page	REN-67	5	High	DONE
As an employee, I want to see the owner requests that are assigned to me	REN-68	6	High	DONE
As an employee, I want to see the status of my assigned requests on a notifications page	REN-69	4	Medium	DONE
Total USP		130		