Testing Plan

**This document has been updated to reflect changes in the test procedure for Sprint 4. It contains new Acceptance Tests planned for sprint 4.

Overview

This document goes over how the team will conduct unit tests, integration tests, and system tests over the course of the development of the Decagon Condo Management System. It will also outline the tools used for each type of test.

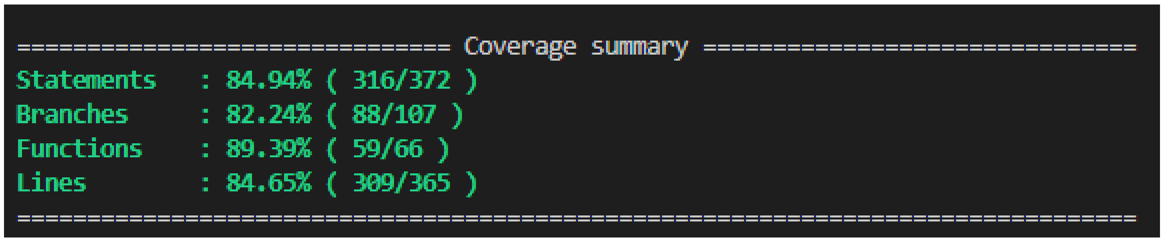
Unit Testing

Unit tests test small units of code, isolated from the rest. Its goal is to test only the basic functionality of a function or class. Unit tests test the behavior of the code itself and by themselves are not capable of validating that the requirements and specifications of the system set by the stakeholders are being respected.

The Decagon Condo Management System is being developed with the Angular framework using TypeScript and a Firebase backend. The Jasmine testing framework is installed by Angular for testing JavaScript and TypeScript code. Each Angular component is generated with a test file where all the unit tests for that component will be written. The “ng test” command from the Angular CLI will run all the Jasmine tests using the Karma test runner which will show which tests pass or fail and the error when a test fails. Adding the “--code-coverage” parameter to the command will report how many many statements, branches, functions, and lines are covered with a percent for each of them. We want to have at least 80% coverage for statements. Adding the “--no-watch” parameter will generate a html file outlining which statements from each class is covered by a test and which are not which will make achieving this test coverage easier.

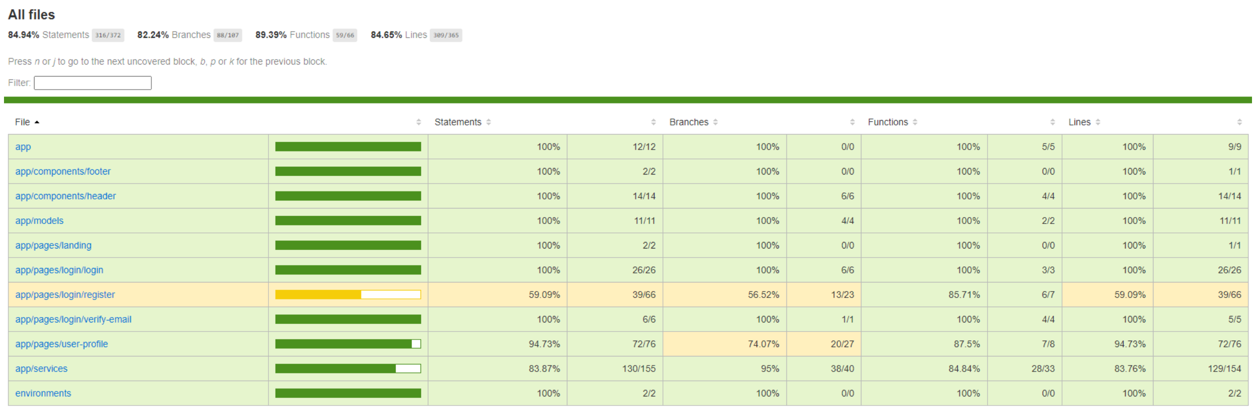
Continuing on to sprint 4, each developer will write and run unit tests for the components that they work on. Unit tests are also configured to run on GitHub automatically after every push to a branch and every pull request. The test run is configured to fail if the statement coverage is less than 80%, enforcing this requirement.

Coverage example



```
=============================== Coverage summary ===============================
Statements   : 84.94% ( 316/372 )
Branches     : 82.24% ( 88/107 )
Functions    : 89.39% ( 59/66 )
Lines        : 84.65% ( 309/365 )
================================================================================
```

Visualization of coverage



All files

84.94% Statements 316/372 **82.24%** Branches 88/107 **89.39%** Functions 59/66 **84.65%** Lines 309/365

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

Filter:

File		Statements		Branches		Functions		Lines	
app		100%	12/12	100%	0/0	100%	5/5	100%	9/9
app/components/footer		100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/components/header		100%	14/14	100%	6/6	100%	4/4	100%	14/14
app/models		100%	11/11	100%	4/4	100%	2/2	100%	11/11
app/pages/landing		100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/pages/login/login		100%	26/26	100%	6/6	100%	3/3	100%	26/26
app/pages/login/register		59.09%	39/66	56.52%	13/23	85.71%	6/7	59.09%	39/66
app/pages/login/verify-email		100%	6/6	100%	1/1	100%	4/4	100%	5/5
app/pages/user-profile		94.73%	72/76	74.07%	20/27	87.5%	7/8	94.73%	72/76
app/services		83.87%	130/155	95%	38/40	84.84%	28/33	83.76%	129/154
environments		100%	2/2	100%	0/0	100%	0/0	100%	2/2

Integration Testing

The goal of integration tests is to test multiple related components to ensure they work well together. We are using cypress end to end testing to test multiple related components.

For the Decagon Condo Management System, we can create integration tests with cypress to test multiple components on the same page, and switch between pages required to perform an action. We can assert that the components cooperate the way they are intended to.

These tests will ensure that different systems work together properly. Mainly, the team wants to assure that the front-end system (Angular and TypeScript) and back-end system (Firebase services) work together well. Cypress provides a way to intercept API calls easily which makes writing tests simple.

The cypress tests are configured to run using the command “npm run cypress:run” or “npm run cypress:open” which will open the tests in a browser which will visualize the process. Cypress is also configured to run on GitHub for every push to a branch and pull request.

System Testing

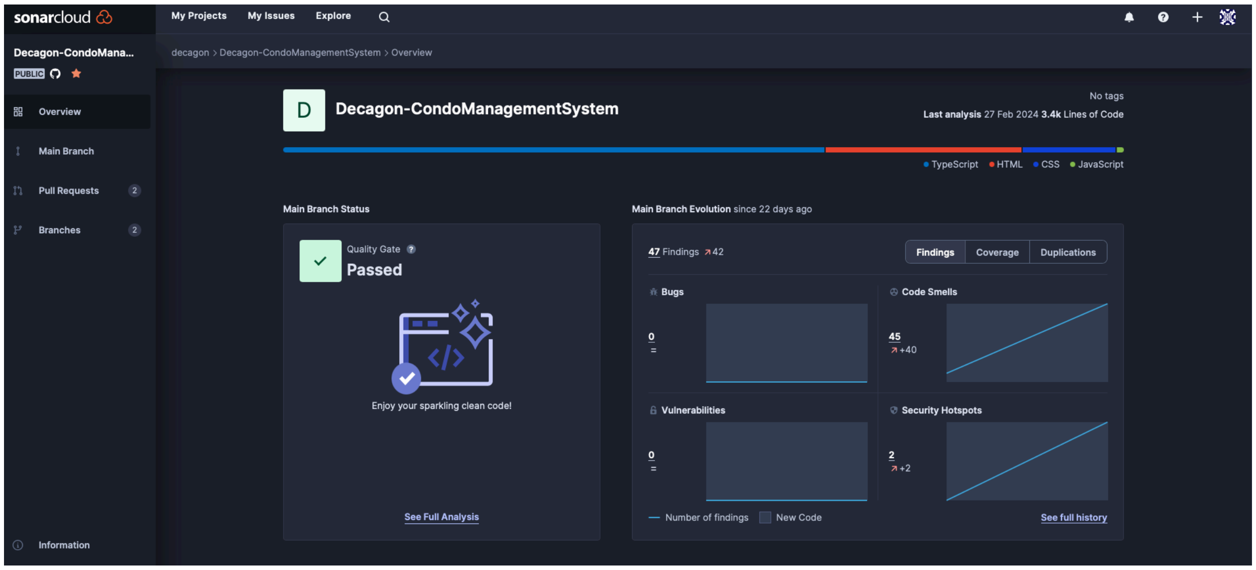
The goal of system testing is to test the system as a whole. It is meant to test all the requirements from start to finish. We will also do manual tests where we test the system requirements manually. We do these tests throughout the implementation and during the review process to ensure that the system is performing well.

Cypress offers the team end-to-end testing which is exactly what we need to perform for system testing. It is convenient since these tests are automated and interact with the software application on its own. Furthermore, since these tests are automated, they can be introduced into the cd/ci pipeline, which enables regression. Regression testing ensures that requirements tested in the past, still work to this day and in the future. All in all, this ensures that system requirements are being tested fast and efficiently.

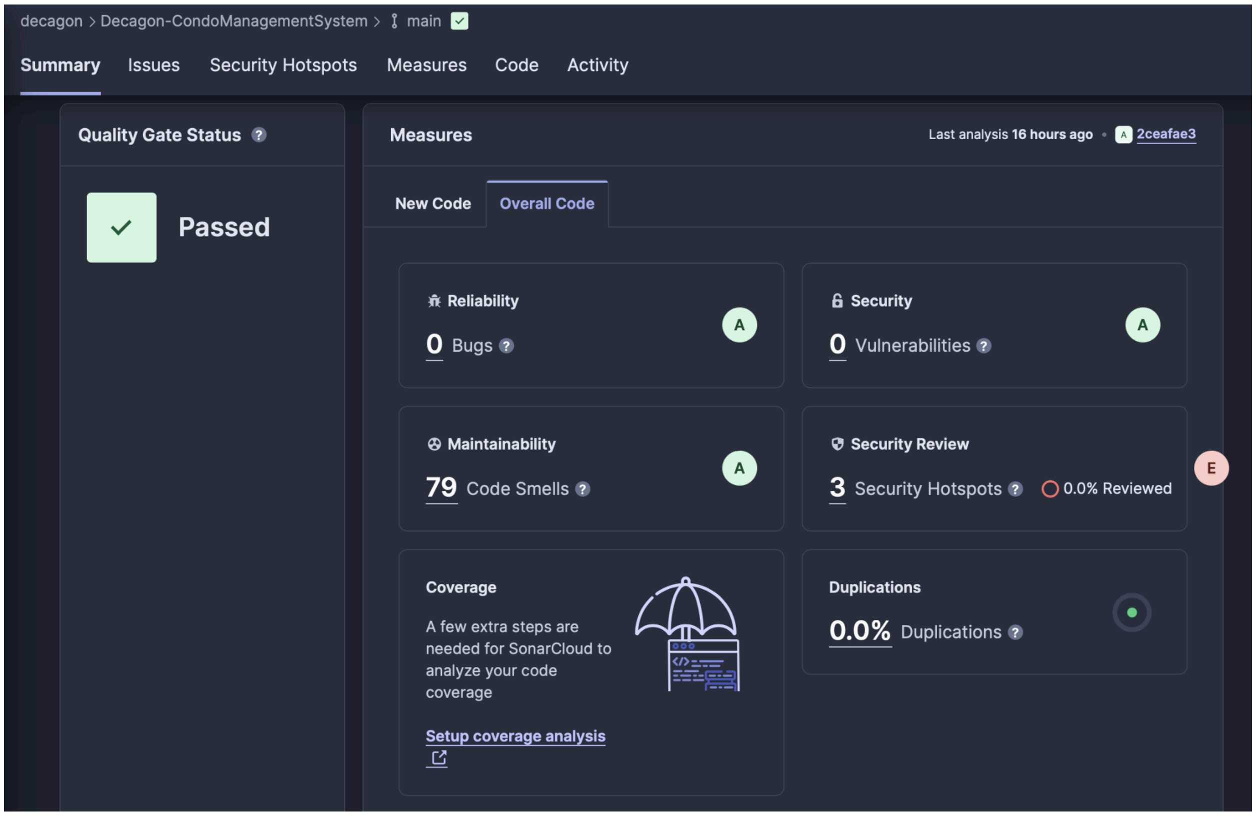
Code Quality

SonarCould is an automated code quality inspection platform. Metrics measured by SonarCloud are bugs, code smells, vulnerabilities, security hotspots and duplicate code. SonarCloud also allows measuring test coverage but we are already measuring this using other methods as mentioned above. SonarCloud is configured to run automatically when changes are pushed to a branch and a pull request is made. The service is set up to report a failure when duplicate code is greater than 5%. SonarCloud will also calculate different quality ratings and will fail the test if a rating is below an ‘A’ grade. This consists of maintainability rating, which is calculated based on the number of code smells, and the reliability rating, which is calculated based on the number and severity of bugs.

Overview



Main Branch



decagon > Decagon-CondoManagementSystem > main

Summary Issues Security Hotspots Measures Code Activity

Quality Gate Status

Passed

Measures Last analysis **16 hours ago** • A 2ceafae3

New Code | Overall Code

Reliability — 0 Bugs — A

Security — 0 Vulnerabilities — A

Maintainability — 79 Code Smells — A

Security Review — 3 Security Hotspots — 0.0% Reviewed — E

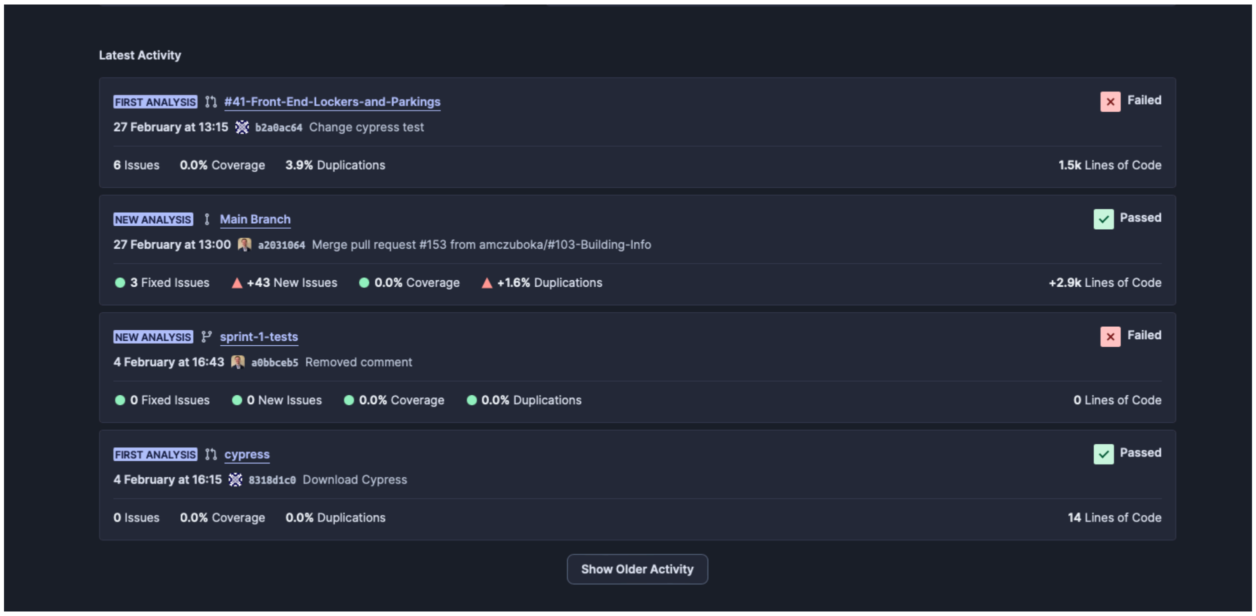
Coverage

A few extra steps are needed for SonarCloud to analyze your code coverage

Setup coverage analysis

Duplications — 0.0% Duplications

Recent commits



Latest Activity

FIRST ANALYSIS #41-Front-End-Lockers-and-Parkings — Failed
27 February at 13:15 b2a0ac64 Change cypress test
6 Issues 0.0% Coverage 3.9% Duplications — 1.5k Lines of Code

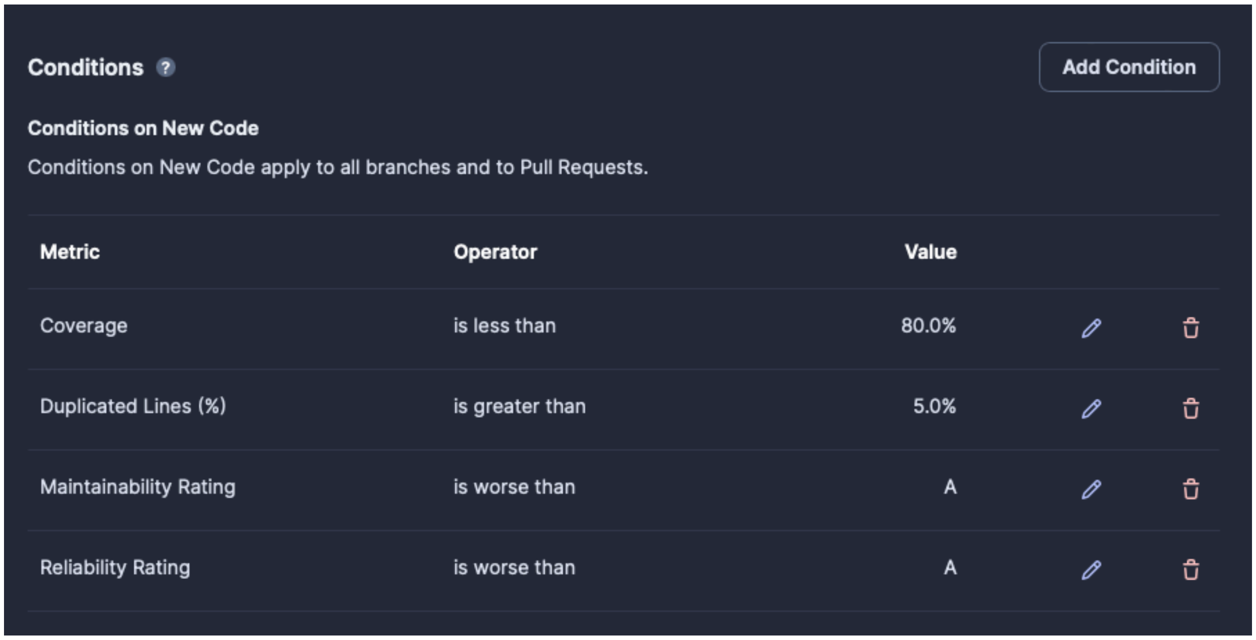
NEW ANALYSIS Main Branch — Passed
27 February at 13:00 a2031064 Merge pull request #153 from amczuboka/#103-Building-Info
● 3 Fixed Issues ▲ +43 New Issues ● 0.0% Coverage ▲ +1.6% Duplications — +2.9k Lines of Code

NEW ANALYSIS sprint-1-tests — Failed
4 February at 16:43 a0bbceb5 Removed comment
● 0 Fixed Issues ● 0 New Issues ● 0.0% Coverage ● 0.0% Duplications — 0 Lines of Code

FIRST ANALYSIS cypress — Passed
4 February at 16:15 8318d1c0 Download Cypress
0 Issues 0.0% Coverage 0.0% Duplications — 14 Lines of Code

Show Older Activity

Failure Conditions



Conditions ?

Add Condition

Conditions on New Code

Conditions on New Code apply to all branches and to Pull Requests.

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	5.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A

Acceptance Tests

Acceptance tests are done using cypress end to end testing that simulates a user interacting with the webpages.

Some acceptance tests were originally planned for sprint 3 but moved to sprint 4.

Planned Acceptance Tests for sprint 4

Create acceptance test for Request page. #43

Sign in as a public user
User navigates to the building-info page of a building
User clicks on “Requests” tab
User selects the request type and writes a comment. Clicks submit.
Company user that owns that building signs in
Result: A new notification is visible
User clicks on the icon
Result: The request message is displayed

Create acceptance test for editing Profile page for unit. #46

Sign in as a company user
Navigate to my properties page
Click on a building
Click on a unit and click edit
Change condo unit fields and save changes
Refresh the page
Result: Condo unit is updated

Create user acceptance for "Budget Report" page. #74

Sign in as a company user with assigned condos
Navigate to budget report page
Result: Sees breakdown of revenue, costs and profit on each building
Result: Sees total revenue, costs and profit.

Create user acceptance test for "Add New Building Operation" page. #75

Sign in as a company user with properties
Navigate to the Add New Building Operation page
Chose a building and fill in the fields
Click submit
Navigate to my properties page and click on the building
Result: New building operation is displayed

Create user acceptance test for "Reservations" page for owners and renters. #79

Sign in as a public user
Navigate to building info page
Navigate to reservation page

Make a reservation on a condo
Result: Reservation is sent

Create user acceptance test for "Reservations" page for managers. #80

Sign in as a employee / manager user
Navigate to building info page
Navigate to reservation page
Make a reservation on a condo
Result: See reservation requests

Create user acceptance test for "Bookings" page for owners and renters. #81

Sign in as a public user
Navigate to building info page
Navigate to bookings
Make a booking
Result: Booking is sent

Create user acceptance test for registering a unit with an account with key. #112

Sign in as a public user
User navigates to the building-info page of a building
User clicks on “Condo” tab
User clicks on “Request for rent” or “Request for ownership” of a listed condo
User enters a valid registration key
User navigates to their properties page
Result: See unit in their list of rented / owned condos

Create user acceptance test for registering a parking and locker with an account with key. #113

Sign in as a public user
User navigates to the building-info page of a building
User clicks on “Lockers” tab
User clicks on “Request for Rent” on an available locker
User clicks on “Parking” tab
User clicks on “Request for Rent” on an available parking lot
User navigates to their properties page
Result: See locker and parking spot in list

Create user acceptance test for individual page of unit #115

Sign in as a company user
User goes to their properties page
User publishes a new property with condo units
Sign out and sign in as a public user
Click on new property and go to building-info page
Click on condo unit
Result: See condo unit information

Create user acceptance test for individual page of building #116

Sign in as a company user
User goes to their properties page
User publishes a new property with a description
Sign out and sign in as a public user
Click on new property and go to building-info page
Result: See building-info page with description, info and company info

Create user acceptance test for requesting a unit/locker/parking #151

Sign in as a public user
User navigates to the building-info page of a building
User clicks on “Condo” tab
User clicks on “Request for rent” or “Request for ownership” of a listed condo
User clicks on “Lockers” tab
User clicks on “Request for Rent” on an available locker
User clicks on “Parking” tab
User clicks on “Request for Rent” on an available parking lot
Sign out and sign in as a company user that owned the same building
Result: Manager sees a notification from the public user requesting a condo / locker / parking

Create user acceptance test for "Pay Balance" page. #164

Sign in as a public user that owns a condo
Navigate to pay balance page
Result: Balance sheet with breakdown of costs for condo fees and extras (lockers, parking, etc)
Click on submit payment
Result: Payment submitted to company