

# Testing Plan

**\*\*This document has been updated to reflect changes in the test procedure for sprint 5. It contains new Acceptance Tests planned for sprint 5.**

## 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. **Current code coverage at the end of sprint 4 is 82.65%.**

## 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 5, 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%	5/5
app/components/footer	100%	2/2	100%	0/0
app/components/header	100%	14/14	6/6	4/4
app/models	100%	11/11	4/4	2/2
app/pages/landing	100%	2/2	100%	0/0
app/pages/login/login	100%	26/26	6/6	3/3
app/pages/login/register	59.09%	39/66	56.52%	6/7
app/pages/login/verify-email	100%	6/6	100%	4/4
app/pages/user-profile	94.73%	72/76	74.07%	7/8
app/services	83.87%	130/155	95%	28/33
environments	100%	2/2	100%	0/0

## 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

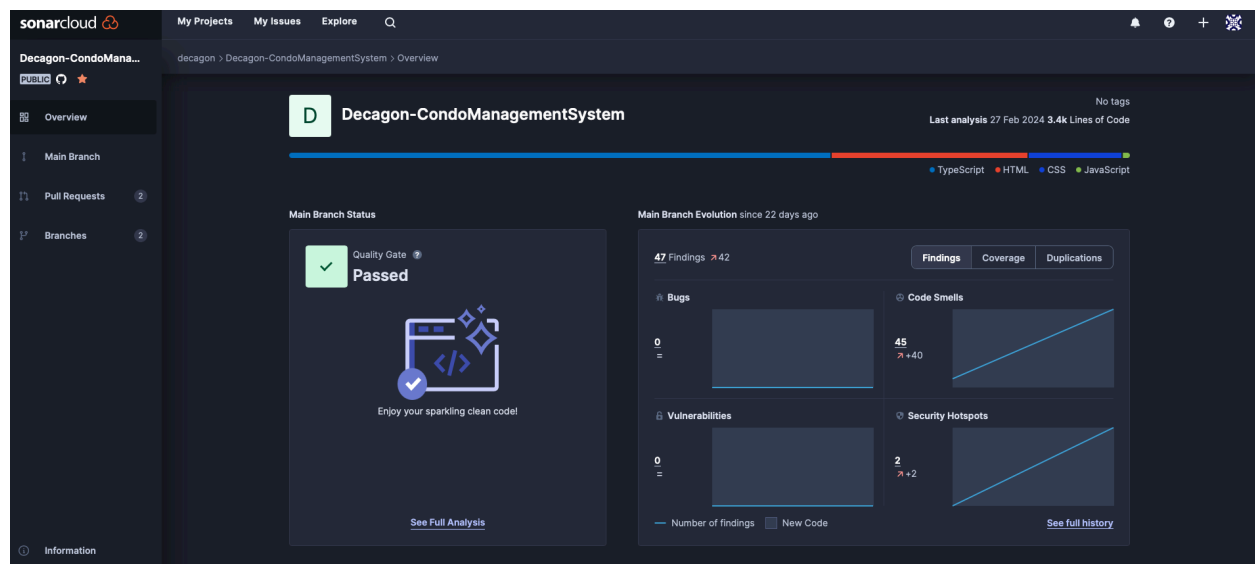
The goal of system testing is to test the system as a whole. It is meant to test all the requirements from start to finish. We will also do manual tests where we test the system requirements manually. We do these tests throughout the implementation and during the review process to ensure that the system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

SonarCloud 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 · 2ceafae3

New Code Overall Code

**Reliability**

**0** Bugs

**Security**

**0** Vulnerabilities

**Maintainability**

**79** Code Smells

**Security Review**

**3** Security Hotspots 0.0% Reviewed

**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 b2a9ac64 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 a2831864 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.

All of these acceptance tests were originally planned for sprint 4 but moved to sprint 5 due to time constraints.

### Planned Acceptance Tests for sprint 5

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