Sprint 1: A Look Back

Introduction

The Condo Management System project aimed to create an Enterprise Resource Planning solution targeted for condominium management. An ERP is aimed at creating “software that helps plan, budget, predict, and report on an organization’s financial results”, described by Oracle.

Entering the project, the team expected to deliver a small part of the Condo Management System featuring clean and easy user interfaces touching the login/signin/authentication implementation. Furthermore, we expected to perform the analysis and design of the entire system by producing several diagrams including a class diagram, context diagram, etc.

This sprint successfully delivered a functioning part of the Condo Management System built with Angular and Firebase that can be seen during the demo on Friday 9th February 2024. It also included all the documentation related to the system which was successfully accomplished.

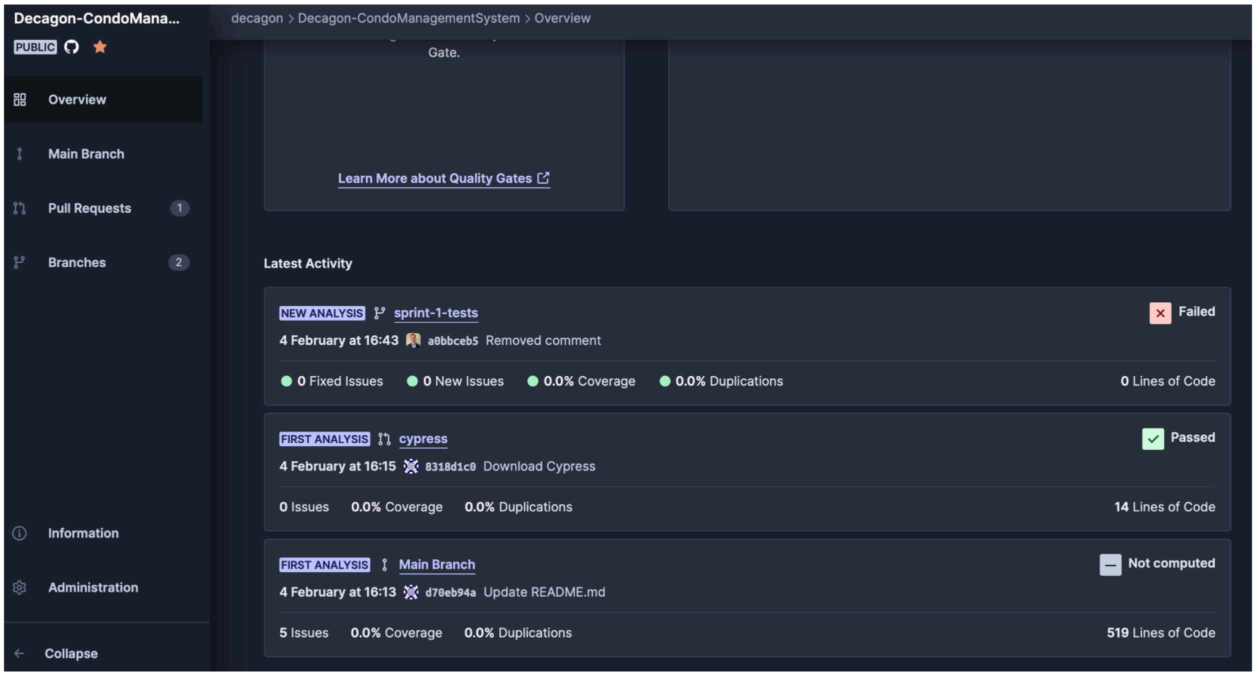
The system will allow condo residents (owners and renters), property managers, and staff to interact easily. This includes registering a condo in your name and account, creating properties and condos in the system by condo management, reserving facilities, etc.

Version Control: GitHub for collaborative development and version control.
IDE: Visual Studio Code (VS Code) for code development.
Diagram Creation: Draw.io for creating diagrams.
Prototyping: Figma for designing and prototyping user interfaces.
Testing: Cypress for end-to-end testing, along with Angular's spec tests for unit tests using the Jasmine/Karma testing framework.
Continuous Integration/Continuous Deployment (CI/CD): Utilized GitHub Actions for automated CI/CD pipelines to ensure a smooth and efficient transition to the customer.
Architecture: Client-Server with Angular and Firebase.

What went wrong

1 - Testing Tool

We wanted to use SonarQube/Cloud to receive feedback on the coverage of tests. However, when we did get the platform to scan our repo, it was confusing and misleading. What it had scanned, why there wasn’t a coverage percentage, etc.



Latest Activity

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

FIRST ANALYSIS Main Branch — Not computed
4 February at 16:13 d70eb94a Update README.md
5 Issues · 0.0% Coverage · 0.0% Duplications · 519 Lines of Code

This impacted our testing abilities by not knowing how much testing we were missing.

We addressed it by implementing the coverage into the pipeline.

Here is a comment by Guillaume Lachapelle “UPDATE: We decided to use "npm run test-headless" which runs the command "ng test --no-watch --no-progress --browsers=ChromeHeadless --code-coverage". This allows us to get the code coverage for the unit tests, and we check if that value is over 80%.
This is done automatically upon every push and every PR on the repository since it is integrated into the CD/CI pipeline. The pipeline build will fail if that coverage condition is not met or if any test fails, including the cypress tests.”

2 - Access issues

We experienced access issues with the Firebase platform.

This impacted progress on our back-end development.

We ensured that all collaborators were marked as “Admin” yet it was only the owner who could do the initial steps like adding a real-time database to the project.

This was confusing since most of the team has used the Firebase platform and never encountered this issue.

We agreed to contact the owner of the Firebase project in a timely manner.

3 - Task documentation

We believed the documentation of tasks and milestones was only done on GitHub.

Therefore, it took longer to document since it was unclear to us why and how we should specify these things without being too redundant or overly detailed.

We discussed with the TA the worries and they assured us a general overview of the task was enough.

We could have asked these questions sooner in any case, since we were sure it was only on Github, but it never hurts to ask.

4 - Risk assessment and analysis

We had some confusion about which documents to fill in and what was needed for the Excel sheet. Therefore, we asked the professor for clarifications on which documents were needed, as well as read the example provided on the Excel sheet to have a better understanding. We could have asked earlier and shared our concerns with the team.

What went right

1 - Division of tasks

We had a meeting early in the sprint to divide the tasks.

This allowed everyone to start working immediately.

We agreed to have a meeting after every demo to discuss the division of tasks for the next sprint.

Since we had not assigned user story points, we could have discussed more the intensity of each task to properly determine the effort it would take. However, everyone worked on this sprint. And it was a success in our eyes.

2 - Collaborative Team Effort

We have all been in projects where we were disappointed and let down by other teammates. Therefore, this motivated us to keep in touch. We fostered a collaborative team culture with open communication. Namely, we are all connected in a group chat that we participate in often. In consequence, we improved problem-solving and overall team satisfaction. We conducted regular team-building activities and encouraged open discussions. Continue emphasizing a culture of collaboration and communication for sustained success.

3 - Software Architecture

A clear depiction of the software architecture was produced due to extensive user requirements analysis. As a result, the software architecture diagrams give the team a clear idea as to how the system will be implemented. Therefore, the team will be able to continue using the diagrams to implement a correct and robust system that meets user needs. The software architecture was reviewed by all team members. However, more frequent review of the changes in the software architecture would benefit the team in later sprints (especially as we start coding more) to be certain that we're all on the same page.

4 - Sprint 1 Tasks Completed

All Sprint 1 tasks were successfully completed on time because all team members worked on their respective tasks in a timely and efficient manner. Thus, we are on schedule with the progression of the project. As such, we are ready to build upon our current code to add more pages and functionality to our Condo Management System in Sprint 2. The completion of tasks was addressed by dividing the work and fostering a good working relationship with members of the team (as mentioned above). An area of improvement could be to review each other's work more regularly so as to not have to make too many changes all at once.

Conclusion

After completing the first sprint of this project, the team got to learn a little about each other and how we worked on previous projects. Some teammates had prior experiences with the tools we are currently using, so the preliminary meeting was also about sharing this information. We learned how to manage a relatively big team while dividing the tasks equally and efficiently. We made sure to have good communication among the team members and to ensure that if any modifications were planned, to notify the team as this could impact their tasks. We will definitely be bringing that good communication into the following sprints.