Design Checklist
Teddy Marchildon & Matthew O'Boyle

DUKE COMPUTER SCIENCE

Project

Abstract

- The Design Checklist is a tool for **visualizing a static analysis of code**
  - Highlights poor design choices that may make the code less extensible
  - Code duplication, magic values, long methods, cyclomatic complexity, etc.
- Used previously in CompSci308, now additionally in CompSci307
  - These courses are designed to teach students the architecture of "reliable, maintainable, and useful software systems."
- This semester we worked hard to:
  - Make the tool available to both CompSci 308 as well as CompSci307
  - Ensure TA's can effectively use the tool for their students to give **quality feedback**
  - Decrease site load times to increase user retention
  - Allow students to **interact with the app** so it can help them organize their project

Challenges

- Handling the large amounts of Gitlab data on each repository
  - Filtering out what we think is important feedback for the students and TA's was very important to ensure they would use the site
- Handling the large amounts of SonarQube data on each repository
  - In the beginning, we presented students with a view of all the issues, sorted by category and sub category
  - According to the Google Analytics, users weren't using our tool to the full potential
  - So we **created a dashboard view** (Figure 1) which immediately presents the user with the most valuable information
- Ensure TA's can effectively use the tool for their students to give quality feedback

Methods

- We further optimized the site through our use of **caching**
  - We initially tried a polling system, where we would poll SonarQube every 2 minutes for new information.
  - We later switched to subscribing to a **webhook** that alerts our server when SonarQube is done analyzing a project
  - This reduced the time of almost all API requests
  - Handling the large amounts of SonarQube data on each repository
- Made the API more **modular** by creating a separate endpoint for lines of code, rather than including it in already large responses
- **Threading** various caching pipelines on the server
- Getting student feedback during class lab sessions

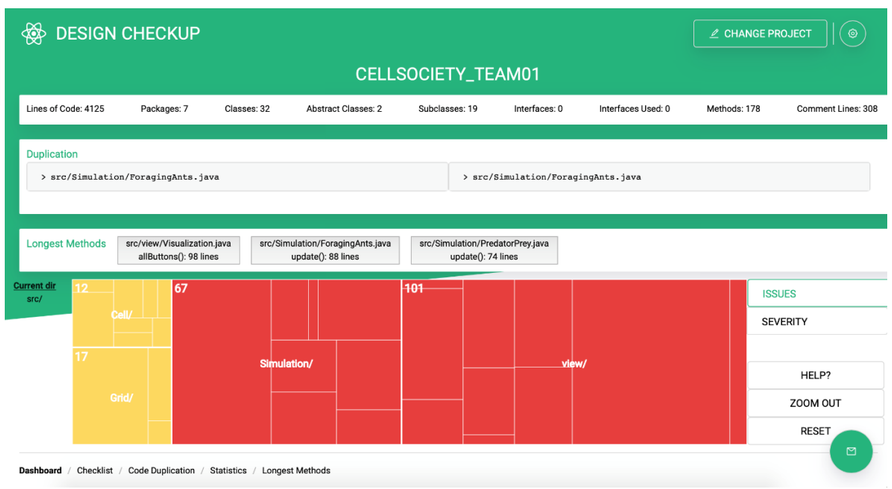

Figure 1: Dashboard view with prioritized information

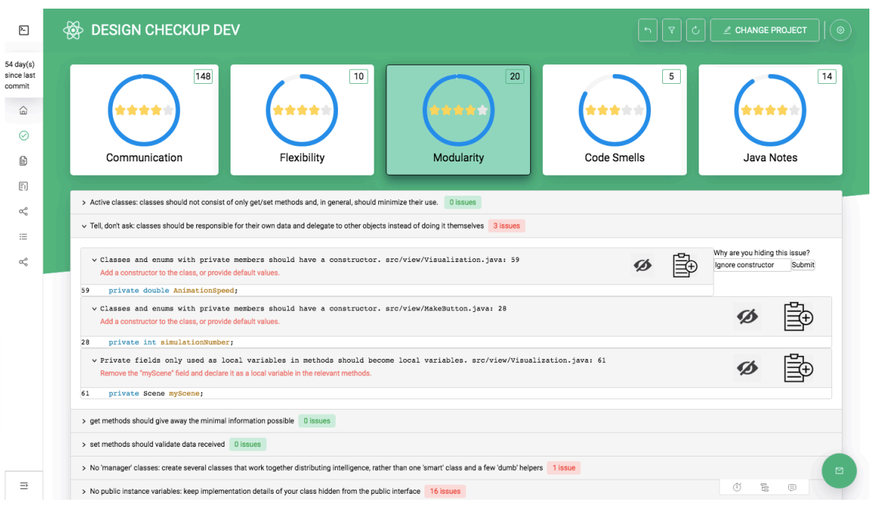

Figure 2: By Category view shows all issues sorted by categories, as well as hide feature and Todo list feature

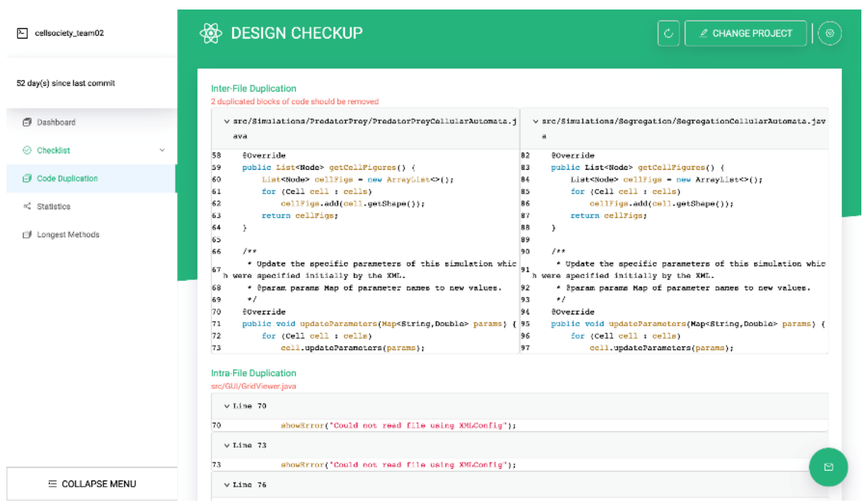

Figure 3: All Duplication of code within the project

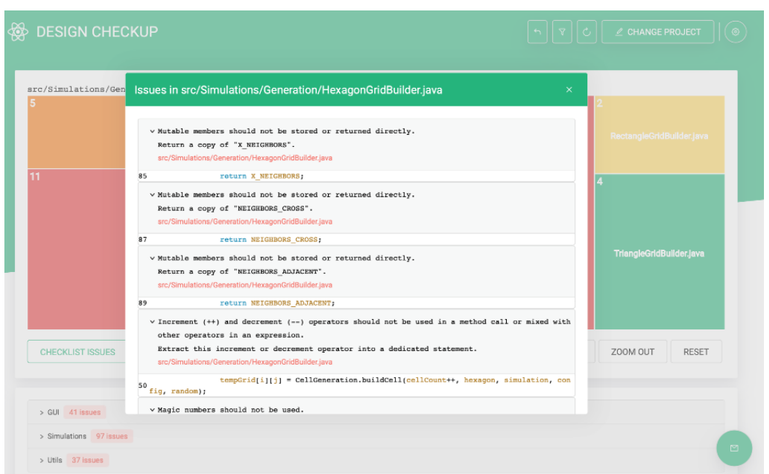

Figure 4: Viewing issues in a particular file from the tree map of issues

Results

- We have presented this tool to students in CompSci 308 the last two semesters, and **have also opened it up to CompSci 307 students this semester** (the first semester 307 is offered).
- During lab feedback sessions, many students found the project very **useful**, and showed eagerness to reduce issues
  - "I didn't even realize we were using so many magic values"
- One issue that appeared was load time when many students accessed the site at once
  - This was worsened as students began to commit their code to master, triggering the webhook from SonarQube and starting a caching process on the server
- We received other feedback such as **making the tool interactive**
  - Students wanted to prioritize some issues more than others

Future Plans

- Our number one priority moving forward is to **decrease the load time** of the site, especially when numerous students are using it at once
  - To fix this, we are considering spinning up VM instances for caching, and another for answering routing queries
- We have responded to the feedback we received and are in the process of making the site **interactive**
  - We have debuted a **Todo list feature**, as well as the ability to hide certain issues that the team may not prioritize
- Viewing **dependency graphs** in a first step toward more code visualizations
- We would love to create a metric to compare different issues. This would allow students to see the the most important issues in their project, and which files have the most issues.

Conclusion

- We believe that this project has been **very successful**
- Student usage has increased as we have executed on our goals
- In-class demos have been helpful in allowing students to see the full potential of the site before they use it
- This semester, we have increased the amount of caching which has reduced API response time
- We have expanded the tool to more than one course by simply adding files to the backend
  - **This allows any java-based course to use the tool**
- We created a dashboard that presents users with the most pertinent information immediately
- We have added user interaction to expand the tools that the site provides for students and project management