**Team <Scentaur>**

**Ga Jun Young** (ga.young@ucdconnect.ie)

Royal Thomas (royal.thomas@ucdconnect.ie)

Zheng Ju (zheng.ju@ucdconnect.ie)

William Ikenna-Nwosu (William.ikenna-nwosu@ucdconnect.ie)

Use a splash page image here [optional]



Use *LaTeX* if you wish, but use the general spacing and font/style you find here (1.5 spacing, 12 point font for text, etc.).

Be sure to submit a PDF (not a .DOC file) as your report. Overall it should be **15 to 20 pages**, including diagrams.

**1. Introduction**

**1.1 Vision of the Project**

Scentaur is a Java-specific “code smell” detector. The team behind Scentaur believes that many current code smell detectors are not particularly easy to use and understand (i.e. JDeodrant and Infusion). Thus, the team envisions Scentaur to be a user-friendly code smell detector tool. We want Scentaur to be accessible, performant, have clear data visualization and finally, to be reliable.

**1.2 What Scentaur Set Out to Achieve**

* **Web-based:** Team Scentaur has set out to achieve a web-based code smell detector tool. Providing a quick and easy way to detect smells with a simple quick drag and drop functionality. The web-based application brings forth a simple UI design and easy accessibility to reach a wide range of audience in hopes of detecting smells within their code.
* **Performant:** Scentaur had set out to reduce the amount of time required to detect smells unlike JDeodrant which takes a significant amount of time to detect smells; as it contains many types of detectors (22) along with the ability to refractor smelly code. As a result, Scentaur is currently providing up-to 4 sub smells for each smell category. This proves to increase performance and in addition, users are enabled to check smells they want to detect within their code.
* **Go-To Product:** When Scentaur first started, the team believed that in order to encourage an audience to code, we should provide a software to help beginners to think about the software design phase of development. To give them an understanding an appreciation for writing maintainable code. To allow them to think about code on a higher level so they can make an immediate impact when working in teams together. Scentaur has achieved this capability by providing users useful tool tips on the smells involved. After analysis Scentaur provides a general but impactful summary to users on the code at hand. This includes smell definitions, colored pieces of code that smells along with its line number, smell category and file location.
* **Reliability:** Testing can give one confidence that the code is **functioning** correctly, Source control gives one confidence that all team members are working on the same version of the code (**Communication** is up to date). Scentaur give users confidence that the system’s (software) **design** is adaptable to change and if a change is made to the system that breaks functionality, it will be caught.

**1.3 Main Goals Achieved**

Team Scentaur has achieved several main goals. Here are the following:

1. To learn and understand the different types of code smells.
2. To understand and take responsibility in a larger team.
3. To enable Scentaur for future uses outside of the assignment background.
4. To enable easy implementations and execution of other code smells in the future.

**1.4 Core Components and Bonuses**

Throughout the project lifetime, Scentaur has become an amazing web-based application. Users are enabled to drag and drop repositories and Scentaur will respond by providing any detailed smells detected within a matter of seconds. This is all thanks to the following core components of Scentaur. The following is a list of core components:

1. **JavaParser:** Without JavaParser Scentaur would be incapable of sniffing Java files in a matter of seconds. The capabilities of JavaParser is immense, from parsing files to creating Abstract Syntax Tree to detecting smells. JavaParser is a huge component of Scentaur, where it is the one that helps provide the smells we wish to detect.
2. **HTML, CSS:** HTML and CSS is yet another core component of Scentaur. They represent the visual aspect that we behind Scentaur provide to users. The easy to interpret drag and drop box is easily spotted when you enter the homepage of Scentaur followed by a simple, user-friendly Interface where end-users can understand the detected problems that Scentaur has provided for them. Without HTML and CSS, Scentaur would not be able to visually represent information to its target audience. Thus, HTML and CSS play a major role in helping Scentaur to communicate the back-end solutions to users.
3. **Tomcat:** Tomcat plays the role of the server for Scentaur. The two components above are not able to complete any task individually. Thus, Tomcat serves as a link between the two (links Java to HTML). Not only that Tomcat allows users to store their temporary files within a server. These files can only be accessed by the user during their session on the webpage. That is because each user is given a unique session ID and granted only permission to contents, they have stored to the server. As a result, Scentaur is a secure web-based application.
4. **Bonuses:** In addition to the above core components, Scentaur contains many plugin-and-play modules. New smells can be easily added into the Software without causing any problems or give dependency issues. We have also used interfaces to follow up the plugin-and-play module whilst using Generics. Statistics is another unique feature that Scentaur uses to give users a more comprehensive detail of the figures involved with the smells detected.

**1.5 Unique Selling Point**

Scentaur strives for uniqueness. The software overall uses JavaParser which greatly helps reduce the amount of coding required to parse and explore classes to find smells. However, that is not the main uniqueness about this program. Its Unique Selling Point is the fact that Scentaur can be run on a web browser available to users who do not wish to download and install a plugin for their favorite IDE. Scentaur overall has many capabilities one such is the fact that it can have a side by side comparison where one side contains the original java file and the other containing pieces of code that contains the code smell. Scentaur also approaches users by providing them with a simple User Interface with some interesting design choices. This allows for easy navigation and use of the web application.

**2. System Features**

**2.1 Features Implemented**

* **Smells:** The first feature that Scentaur focused on was the most simplistic code smells that Scentaur can detect. This was “Bloaters” and its sub-categories such as “Large Class, Long Method, Long Parameter List and Primitive Obsession”. This helped in the decision of the plug-and-play system that we wanted to use. Which had some incorporation of JavaParser as it enabled the team to visit specific nodes that is only required for a specific smell. E.g. Long Parameter List would only need to visit Method Declarations (JavaParser) and look at its parameter size. A unique feature about smells is that since every other smell i.e. Long Method is an extension of smell (An abstract class). We can initialize a smell class with for example large class. This can further then be stored in a list where this list can be all sorts of smell.
* **Report:** This is one of the major features of Scentaur as it produces information and output to the users of Scentaur. In order to do such a thing, it must analyze the project and store this information in its class. This information can be reused, and it is not necessary to reanalyze the project a second time. This report can generate a report for both the web server and onto a text file where users can keep for themselves. The fact that the report does not need to be reanalyzed as it has private variables that store these kinds of data allows Scentaur to specifically pick java files that the user might only want to investigate. E.g. User only wants to detect smells for Car.Java and the Report can do such actions.
* **Statistics:**
* **Server:**
* **HTML,CSS:**

**2.2 Not Implemented Features & Planned Features**

* **Some Smells:** Scentaur works off a plug-and-play system to enable smells to be implemented at any stage of the designing process. And it was planned to have at least 4 smells for each main category (Bloater, Abuser, Coupler, Dispensable). However, due to the plug-and-play system and the fact that each smell is required to be polymorphed to the Smell class for further usage. It was rather complicated to implement some smells for a specific category E.g. Temporary fields and Refused Bequest. This was because of how JavaParser worked as it was not capable of comparing things outside of the class scope. i.e. if the class was a sub class, it would have no information about its super class, thus it wasn’t sure if the sub-class had overridden a method in the super class or not. This reduced the amount of possible code smells we could’ve implemented into our system. Thus, the goal of having at least 4 smells per category was not achieved but only half satisfied.
* **Java Source Code onto HTML WebPage:**
* **Color:**

**2.3 Distinguish Your Project**

What features did you implement and why? What features did you deliberately *not* implement? What features did you plan to implement but not actually do in the end? What features do you believe distinguish *your* project from the projects of your classmates.

**3. Project Design**

**3.1 Model**

- Smell Model

- Report Model

- Web Framework

- Software Pattern -> Plug and play, generics, abstraction

- Third party librarys – maven, javaparser

- Proud of aspect [Website, Server, Smells]

-Benefit Output

Tell us about the design of your system. What model does it follow? Does it use a web framework or is it launched as a standalone application from the desktop or the console? What software patterns did you use in its design? What third party libraries or components did you build upon? What aspects are you most proud of? What aspects would benefit from further development, or even refactoring? Feel free to use software diagrams here to provide a schematic overview of the project’s design, but only if they offer value for the space they take up.

**4. Successes and Failures**

Working as a group is a learning process; it is an explicit learning outcome of this course. So tell us what you learned from working in a team with your fellow team members. What problems arose, and how did you deal with them? How would you do things differently in the future, or if you had to do it all over again. Do not be afraid to discuss failures too: you will not be penalized for being open about the downs as well as the ups of your project.

Who did what in the end? Who deserves special mention for going above and beyond the call of duty? Who did less, or nothing at all? If you feel that a team member did not contribute, you may say so here, but do give that member the right of response within the report.

**5. Team Communication**

Your interim report you laid out your expected communication channels and an optimistic plan for in-group interactions. How did that work out? What was your experience with specific tools of software development, whether relating to version control, planning or communication?

**6. Your Project In Detail**

**- Self diagnosing ( probably 2-3 pages of images, data and explanation)**

Present a worked example of your *NoseJob* application in action, working upon itself to report on its own code smells. What does your system say about your code? How does this align with your own insights into the code? Is the analysis fair, or useful? What does it get right? What does it get wrong? Feel free to use screenshots here.

If your project requires any non-obvious steps to launch/activate, outline those steps here. We should be able to launch your application successfully from what you tell us here.

**Acknowledgements**

Every team member should contribute an equal effort to this report. Use this optional section to provide report credits, or to highlight a special contribution by a given team member.

**References ­**

List any bibliographical citations here [optional]