# OOAD project 6 final report

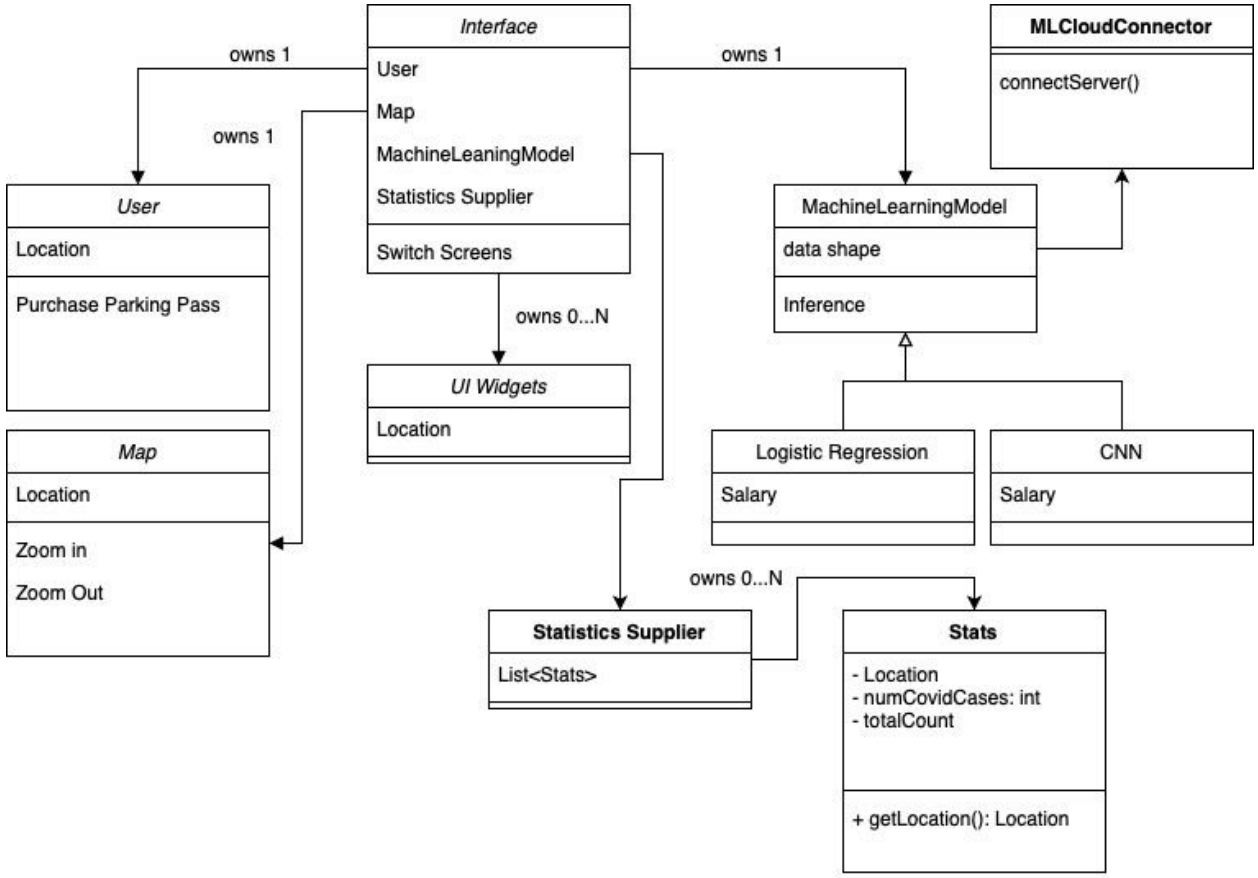
Knowid: The Coronavirus Pandemic Android Application
Lucas Hayne
Abikamet Anbunathan

- Final State of System Statement

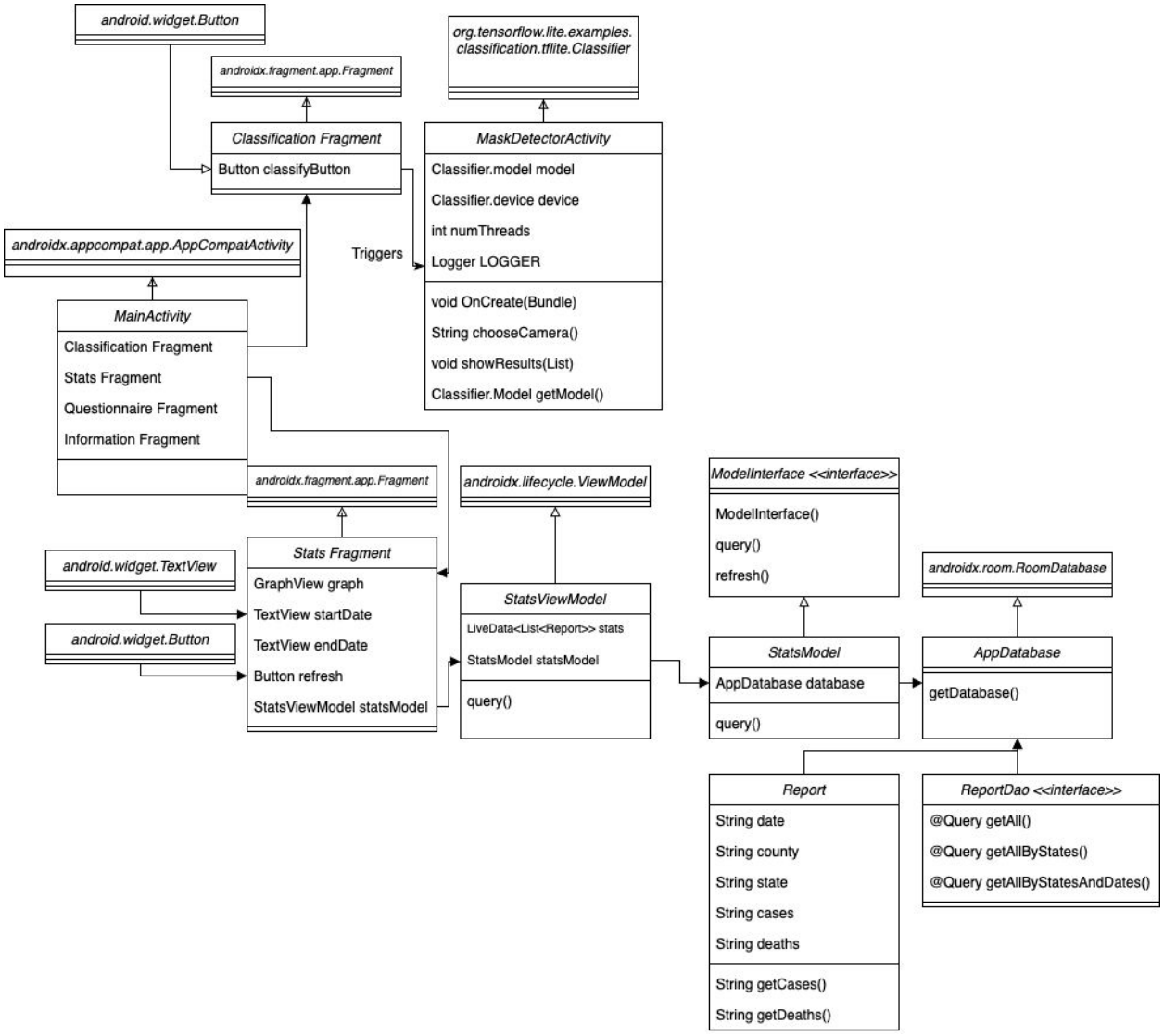
  The implementation of the semester project for us changed quite a bit from the initial design. These changes occurred for a number of reasons. Firstly, neither of us had too much experience with Android Development and as a result changed our implementation a bit to fit the needs of that platform. Additionally, the scope of our initial design was slightly more ambitious than we realized, and as a result scaled back our efforts, instead focusing on implementing object oriented solutions to a smaller set of problems than we imagined. In the end we implemented within our project four use cases: a mask detector, a covid-19 diagnostic questionnaire, an information page, and a statistics browser.

- Final Class Diagram and Comparison Statement
  - A thorough UML class diagram with your final set of classes and key relationships of the system
  - Highlight and document in that diagram any patterns that were included (in whole or part) in your design
  - Include the class diagram submitted in Project 4, and use it to show what changed in your system up to the final submission
  - Support these diagrams with a written paragraph identifying key changes in your system since your design was submitted in Projects 4 and 5

Project 4 UML Diagram

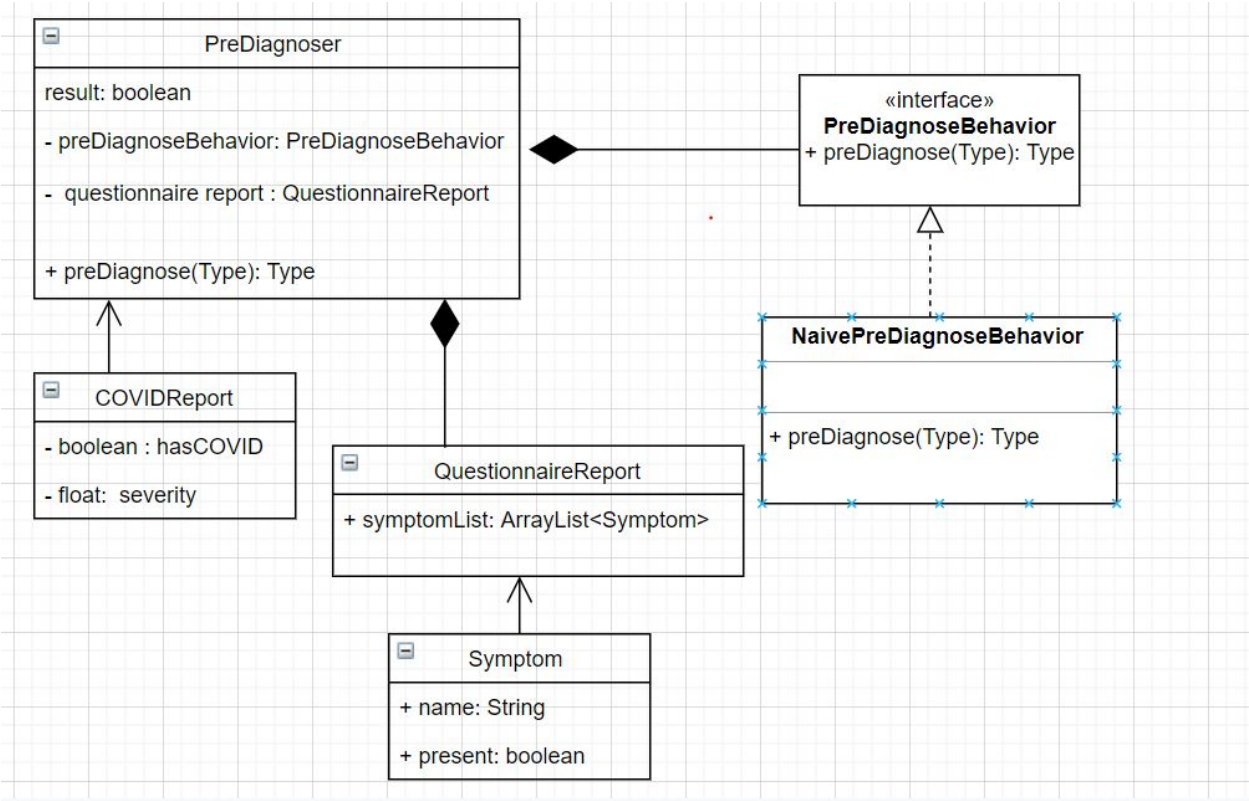


Project 6 UML Diagram(Stats and MaskDetector Fragments)




Here, the StatsModel, android widgets, and StatsFragment represent a MVC compound pattern. The user interacts with the inherited android widgets (view) which alert the StatsFragment (Controller) to query the database through statsModel (Model). Updates flow back from the model to the screen.

UML diagram for Questionnaire fragment




The Main Purpose of the Questionnaire fragment was to collect some information about the user
in order to preDiagnose(Basically an educated guess) and find out the odds of having COVID. Apart from the Android internal frameworks used (mentioned in the previous section) added some custom classes in order to apply the OOP principles and decouple the system as much as possible.
Here **Strategy Pattern** is used to encapsulate the preDiagnosis method. For Example in the future a sophisticated machine learning based preDiagnosing system could be added by just implementing the PreDiagnose behavior without having to recompile the PreDiagnoser class
User fills out a Questionnaire which is stored in the QuestionnaireReport which is part of the PreDiagnoser which then returns COVIDResult based on chosen PreDiagnose Behavior.

- Third-Party code vs. Original code Statement
    - A clear statement of what code in the project is original vs. what code you used from other sources – whether tools, frameworks, tutorials, or examples – this must be present even if you used NO third-party code - include the sources (URLs) for your third-party elements
    - Statement on the OOAD process for your overall Semester Project
    - List three key design process elements or issues (positive or negative) that your team experienced in analysis and design of the OO semester project

| Original Code | Third-Party Code |
| --- | --- |
| • StatsFragment.java*<br>• StatsModel.java<br>• StatsViewModel.java<br>• ModelInterface.java<br>• Report.java<br>• ReportDao.java<br>• MaskDetectorActivity.java*<br>*(except where comments indicate inspiration drawn from other sources)<br><br>• PreDiagnoser.java<br>• COVIDReport.java<br>• QuestionnaireReport.java<br>• Symptom.java<br>• PreDiagnoseBehavior.java<br>• NaivePreDiagnoseBehavior.java | Everything not listed in Original Code. We relied heavily on examples from developer.android.com. |

## Statement on the OOAD process for your overall Semester Project

The OOAD process was initially hard since we did not have an accurate estimation of how much effort was required, and we got a lot of ideas during the development stage. Realized more opportunities for decoupling the system.
It was hard to figure out what design patterns to use at the Start. But towards the end things made more sense.

1) Should have used a more iterative approach to design and development
   Using a quick prototype would have allowed us to make necessary changes earlier.

2) Tried to force object orientedness to the application, instead of just developing a simpler version of the application and allowing design pattern ideas to come naturally.

3) Being able to work on separate parts of the application independently, our progress in work did not interfere with each other, and could work on our own schedule and combine them easily( beauty of Android Activities being independent and yet so cohesive).