題目：基於DevOps之程式作業品質測試分析系統

(一)摘要

大多數的程式批改軟體，像是瘋狂程設、online judge等等，他們批改的要求大多著重在程式執行的結果是否正確。較少著重於程式撰寫品質的檢查，此情況容易造成學生只在乎結果是否正確，而不在乎程式碼撰寫的品質。

因此在本計畫中，我們預計開發一個具有程式自動批改功能的教學輔助系統，系統的功能除了擁有一般批改軟體有的單元測試來判斷答案對錯之外，會再加入檢測程式品質的功能，並且在檢測軟體品質的同時提供使用者改善的建議，藉此讓使用者能根據系統給予的提示，提升程式碼撰寫的品質。根據Pulasthi等人的研究[2]，使用DevOps開發模式有助於提升軟體程式碼品質，所以我們決定將DevOps的開發模式套用在學生繳交程式作業之上，在學生熟悉軟體開發流程的同時，也能提升撰寫之程式碼的品質。

關於程式品質的檢測重點，本計畫預計採用CISQ(Consortium for Information & Software Quality)組織對於程式品質(Code Quality)的規範當中的安全性(Security)、可靠性(Reliability)以及可維護性(Maintainability)這三項指標作為主要的焦點，因此，所開發的系統於程式品質檢測的重點也會放在這裡。

除此之外，為了能夠更方便、直觀地了解學生的學習狀況，我們會將檢測結果統整到儀表板(Dashboard)上顯示出來，儀表版會擁有題目分數、程式碼品質檢測結果以及學生程式碼的品質曲線圖等資訊，讓系統能提供使用者如何針對所撰寫的程式碼進行品質提升的參考範例。

(二)研究動機與研究問題

在大一程式設計這堂課程中，使用了系上老師自己開發的線上自動評分系統，這套系統能正確的判斷程式執行的結果，當時的我們也就直覺的認為程式只要結果對就好。到了大二，在軟體工程這堂課中，我們了解到了軟體程式品質的重要性，如果只在乎結果，程式中可能存在許多不必要的變數或結構，而不好的變數的宣告方式及糟糕的函式寫法都有可能產生安全性問題、影響可讀性，從而降低程式的品質。

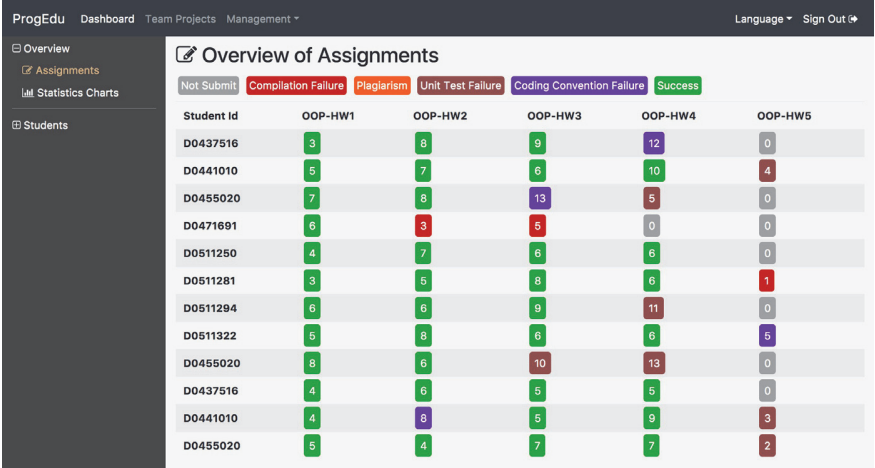
藉由系上資訊專題課程的契機，我們預計開發出一套能判斷程式執行結果正確性，且能夠分析程式品質的教學輔助系統。對於這套系統，我們將重點放在如何檢測程式品質以及如何提升使用者程式碼品質上。檢測品質的部分，會利用SonarQube來檢測上面所使用的三項指標：安全性、可靠性以及可維護性。提升程式碼品質的部分，在檢測完後，系統會將可改進的部分標記起來，並提供範例給使用者參考，此外，系統也會將每次結果紀錄下來，生成程式碼的品質曲線圖，以便了解學習狀況。

(三)文獻回顧與探討

在本章節中，我們將針對相關的參考文獻及本計畫預計會使用到的開源工具進行介紹。在參考文獻部分，我們參考了現有的程式批改系統ProgEdu[1]及其系統架設方式，並預計於本計畫中，使用Git作為學生繳交程式碼的工具，並且加入持續整合(CI)軟體Jenkins，讓我們能即時構建並檢測學生繳交的程式碼。在品質檢測的部分，亦會再加入更多的品質檢測項目。此部份我們參考了CISQ - ISO/IEC 5055，針對程式的安全性、可靠性以及可維護性進行評量。針對這三項性質，系統能透過SonarQube檢測，並且針對特定的檢測結果能產生可直接供使用者使用的建議程式碼，這個部分參考了Diego[4]等人所提出的研究。在老師出題方面，預期將使用SCORM[5]規格一致化的概念，建立一個能匯入匯出系統的題庫，方便老師出題。使用工具的部分，將會在下方與上面提到的參考文獻一同介紹。

3-1.參考文獻回顧

1. ProgEdu：[1]是一套能自動檢查、編譯、單元測試的系統，並提供Checkstyle、Unit test等程式碼的檢驗結果，並將結果呈現在儀表板(圖一)供老師和學生檢視。



圖一 ProgEdu將各個學生的各個作業之狀態顯示在儀表板

1. CISQ - ISO/IEC 5055:[3]根據四個關鍵因素來衡量軟體產品的內部結構： 安全性、可靠性、性能效率和可維護性。這些是決定軟體系統的可信度、可靠性和彈性的因素。在ISO 5055之前，沒有國際標準透過分析其內部結構來衡量軟體系統的品質和完整性。這相當於只透過房屋的外觀來評估房屋，而沒有檢查其內部結構是否有木材腐爛。ISO 5055 提供了在開發中產品的事前檢查措施，以在發生問題之前識別和消除結構缺陷。
2. SCORM: SCORM(Sharable Content Object Reference Model)[5]是一套電子學習產品的技術標準。 它提供了能讓電子學習內容和學習管理系統（learning management system，簡稱LMS）協同工作的通信方法和數據模型。 它告訴程序員如何編寫代碼，以便他們構建的內容能夠與其他電子學習軟件“很好地配合”。 SCORM 是可用的最廣泛使用的電子學習標準。
3. 在Diego[4]等人的研究中，依據SonarQube中最常被手動更正的11個靜態代碼分析規則進行建議程式碼的撰寫，並將結果作為Pull Request提交到專案中，結果顯示高達84%的請求被合併，這樣的結果顯示此生成修復建議的方法是可行的，並且可以幫助增加靜態代碼分析工具的應用性。

以下為該研究中提到的11個最常被檢測出違反的規則：

每項規則前的B和C分別代表Bugs及code smell。

B1 Strings and boxed types should be compared using equals()

B2 BigDecimal(double) should not be used

C1 String literals should not be duplicated

C2 String functions use should be optimized for single characters

C3 Strings should not be concatenated using + in a loop

C4 Parsing should be used to convert strings to primitive types

C5 Strings literals should be placed on the left-hand side when checking for equality

C6 Constructors should not be used to instantiate String,BigInteger, BigDecimal, and primitive wrapper classes

C7 "entrySet()" should be iterated when both the key and value are needed

C8 Collection.isEmpty() should be used to test for emptiness

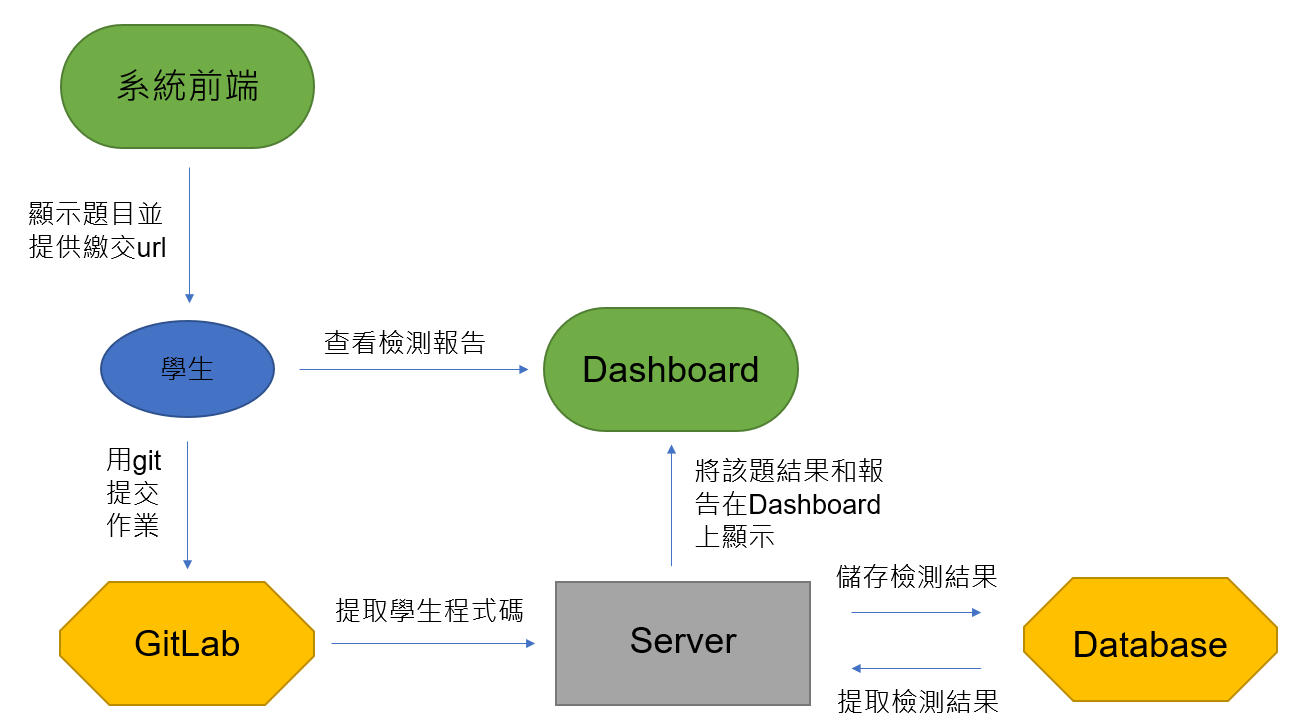
C9 "Collections.EMPTY\_LIST", "EMPTY\_MAP", and "EMPTY\_SET" should not be used

3-2.使用工具

1. 系統前端：預計從當今較為普遍的三種JavaScript框架Vue、React和Angular中選擇其一來建立前端使用者介面。
2. Vue：是一套以視圖層為基礎發展的 JavaScript 漸進式框架。Vue的目標是透過簡單的 API 提供開發者實作資料綁定與操作網頁上的元件。
3. React：是一個高效且具有彈性的 JavaScript 函式庫，用以建立使用者介 面。它讓你使用小巧而獨立的「component」，來建立複雜的 UI。
4. Angular：是一個基於TypeScript的開發平台。它將核心功能和可選功能作為一組 TypeScript 函式庫進行實現。
5. GitLab[7]：利用Git作為版本控制系統，除了讓學生方便繳交作業，還能讓學生熟悉系統開發的流程。GitLab 是DevOps 平臺，它使組織能夠通過更快、更高效的交付軟體，同時增強安全性和合規性，最大限度地提高軟體開發的整體回報。
6. 系統後端：使用RESTful API來連接系統的前端和資料庫。使用RESTful API能讓API接口簡單且統一，且能讓後端API開發上好維護且有彈性。
7. 系統資料庫：將根據之後所決定的資料儲存方式來決定使用
8. RDBMS資料庫(ex:MySQL)或是NoSQL資料庫(ex:MongoDB)。
9. MongoDB：是一個文件導向 (Document-oriented database)的資料庫管理系統的資料庫。
10. MySQL：是一種關聯數據庫管理系統，關聯數據庫將數據保存在不同的表中可有效增加速度並提高靈活性。
11. 程式品質檢測平台：使用SonarQube[8]對學生程式碼進行品質檢測。SonarQube提供程式碼品質、風險檢測解決方案，目前提供了超過27種以上程式語言的開發分析。
12. 系統整合工具：系統將建置Jenkins來協助學生能快速建置作業。Jenkins提供了軟體開發的持續整合服務，除此之外也提供相當多插件來支援不同的專案開發。
13. 系統整合容器：系統將透過Docker[6]來建置，以方便管理。Docker是一個用於開發、發佈和運行應用程式的開放平臺。擁有輕量化、提高資源效率、更快建置等優勢。

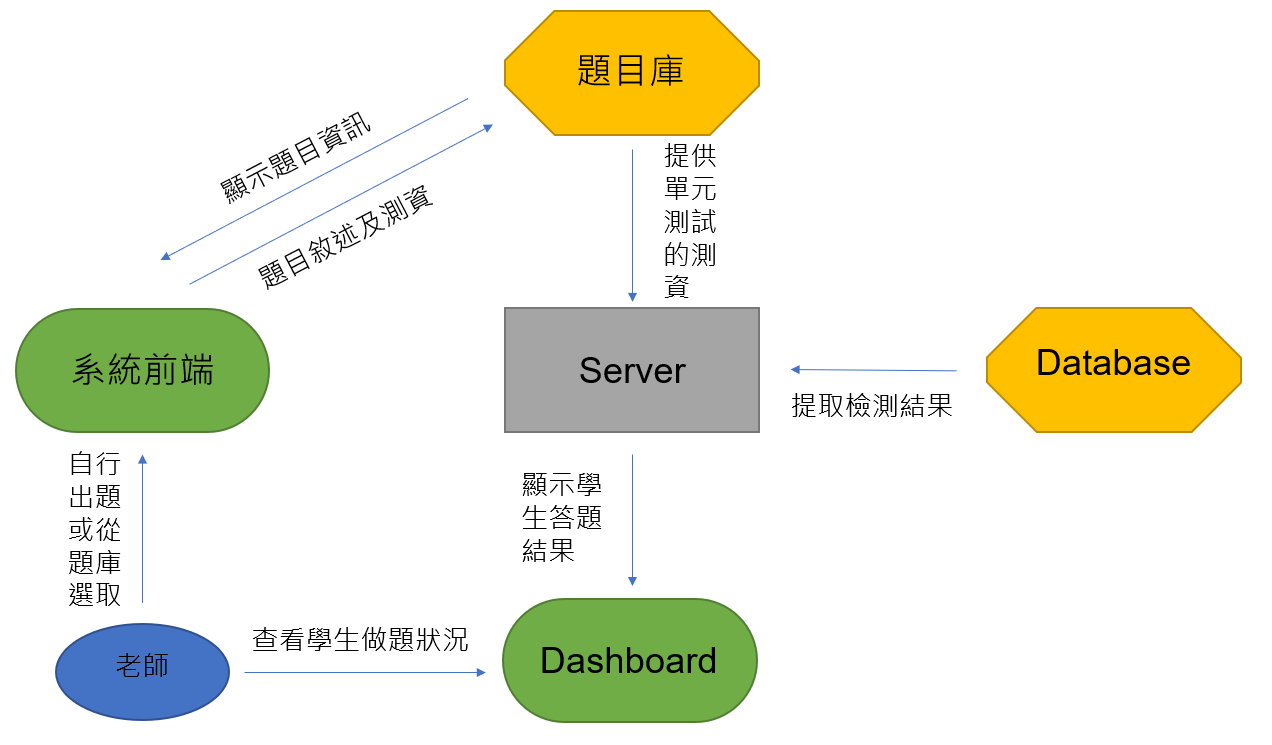
(四)研究方法及步驟

本計畫預計使用GitLab、Jenkins作為繳交作業的工具，來達到上面所提到的DevOps開發模式的概念，以提升學生對於DevOps的熟悉度，關於單元測試，則透過Maven的test階段來進行，若程式通過單元測試檢測，才透過SonarQube進行程式品質檢測。系統會由SonarQube歷次檢測報告中的Bugs、Vulnerability(漏洞)及code smell(異味)的數目來繪製分析曲線圖，分別對應到前面程式品質規範中的安全性，可靠性及可維護性，此外亦利用SonarQube報告對於程式碼的警告，在該段程式碼處給予使用者建議程式碼，以達到提升使用者程式碼品質的目的。



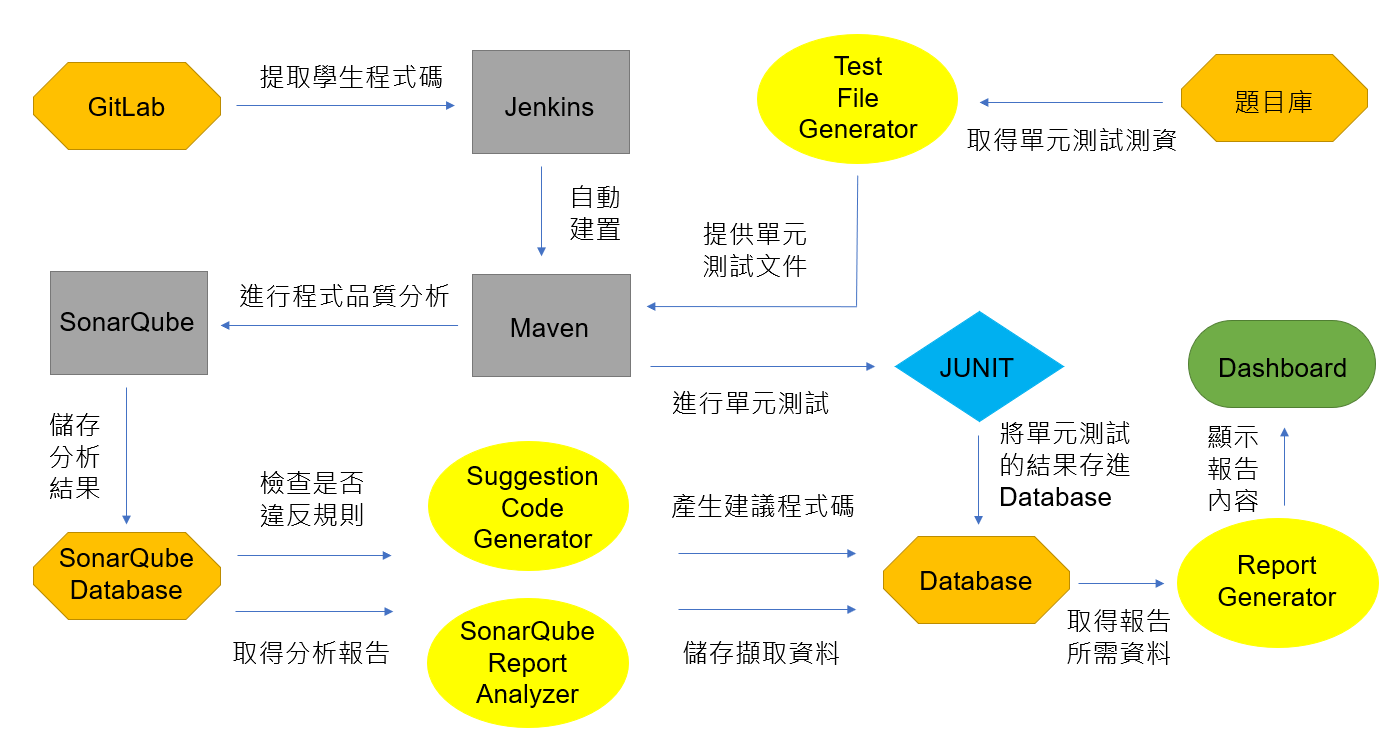
圖二 學生操作流程

如圖二所示，學生從系統前端查看題目，完成作業後統一將撰寫的程式碼以Git提交到GitLab上，而當有檔案被上傳到GitLab時，系統會自動建置並進行單元測試，通過單元測試後使用SonarQube檢測程式碼的品質，並將結果彙整成一份報告呈現在儀表板。



圖三 老師操作流程

老師的流程部分，如圖三所示。老師可以在透過系統前端介面進行出題，且可在儀表板觀看學生的答題狀況。在出題方面，預計採用SCORM的概念，老師可以從網路上找題庫進行匯入或是自己出題，只要符合系統格式，就可以將題目匯入題目庫中。



圖四 系統資料流

系統建置規劃上，預計將各個系統元件分別放入多個Docker Container中，各元件所含的Docker Image將寫入docker-compose.yml中，利用Docker-Compose組合多個 container 成為一個完整服務的工具，使用 docker-compose up 指令，即可將所有的 Docker Container 執行，如此便可以快速且方便的啟動多個 container。

如圖四所示，預計的系統資料流程如下，首先，當學生將作業提交到GitLab上後，Jenkins偵測到版本變更，Maven會做自動建置，系統會利用Maven的test階段的Maven- surefire -plugin插件來調用Junit做單元測試，通過單元測試後，SonarQube會進行程式的品質分析並產生報告。系統會分析SonarQube產生的報告，並將需要的資料存入系統的資料庫中，Report Generator則會利用系統資料庫中的資料生成報告並將內容顯示在Dashboard上。

本計畫的研究步驟分為三個階段，分別為：

階段一：基本環境建構(GitLab, Jenkins, SonarQube, Maven, Database, 題目庫)及Test File Generator 的製作

階段二：Suggestion Code Generator, SonarQube Report Analyzer的製作

階段三：加入前端，及Report Generator的製作

在階段一時，會先建構基礎的環境，包含GitLab, Jenkins, SonarQube和Database的架設。其中GitLab會作為每個學生繳交作業的工具，為了防止學生互相偷看程式碼、抄襲作業，目前預計每一位學生每一題都建立一份專案，設定成只有該專案的成員才能進入該專案。

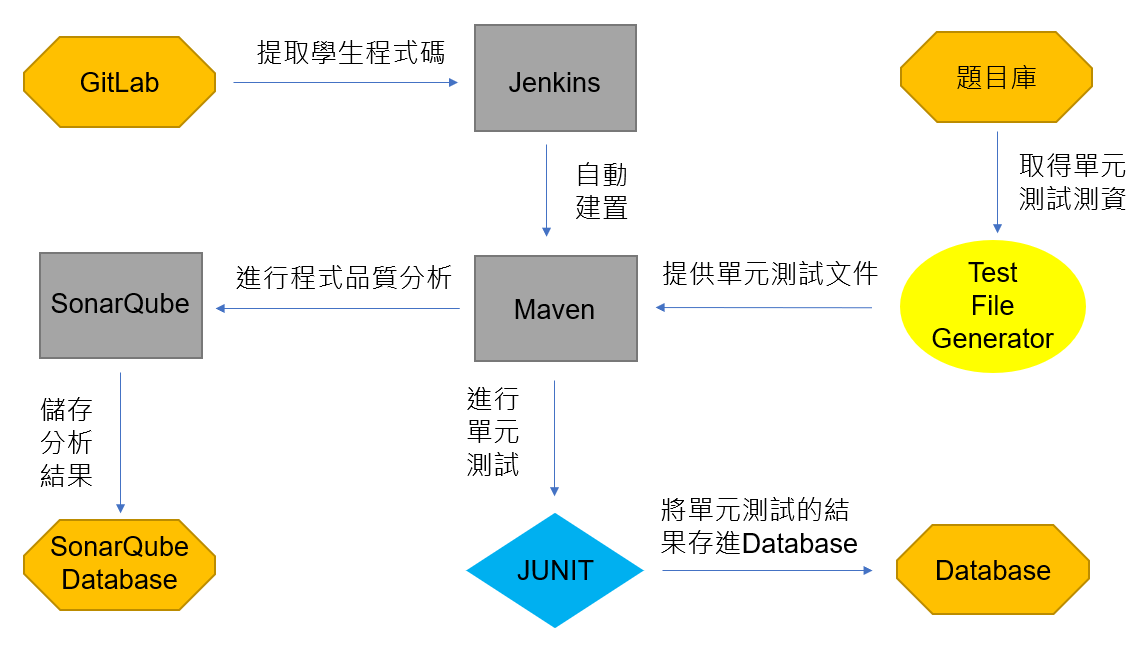
Jenkins的部分，我們會在Jenkins安裝GitLab的plugin，然後在GitLab拿到SSH的金鑰，再回到Jenkins配置token，最後設定當有程式碼提交時，就會觸發Jenkins的部署操作。如此每當GitLab有新的檔案上傳時，Jenkins便會自動建置專案(Maven)並對程式碼進行單元測試及品質測試。

接著會將SonarQube加入系統中，架設SonarQube後與Jenkins連接，也會架設屬於SonarQube的資料庫，SonarQube會將程式碼掃描後的結果儲存到SonarQube的資料庫。SonarQube產生的報告內容將安全性、可靠性及可維護性各自分為A,B,C,D,E這五個等級，可維護性藉由評估Technical Debt Ratio來給予等級，其中Technical Debt定義為將所有code smell修正所需花費的時間，可靠性則是藉由程式碼產生的Bugs的嚴重程度來給予等級(Blocker Bugs, Minor Bugs, Minor Bug,s Minor Bugs)，安全性藉由評估程式碼的Vulnerability issues嚴重程度來給予等級。 另外還會提供覆蓋率，重複率，以及程式碼的Top Common Issues, Top Issues By Severity, Top Security HotSpots to Review。

題目庫的部分預計以JSON的格式做儲存，為了符合SCORM規格一致化且能配合其他符合格式的平台使用，我們會為題目制定一個格式，整個題目會分為題目敘述、測資和答案三個部分，只要符合這個格式的題目都能存入題目庫。除了匯入現有的題目庫，老師也能自行出題，但必須符合格式才能被採用。

單元測試則是透過Maven default生命週期的test階段綁定了Maven-surefire -plugin插件，該插件可以調用Junit完成單元測試，我們會製作一個Test File Generator，將題目庫中老師選取的題目的測資和答案，做成單元測試的程式。目前計畫系統會先產生單元測試程式的基礎樣本，透過這個樣本我們將資料庫中的測資和答案寫到樣本中，最後將程式移到Maven專案中。

而在專案自動建置的過程會需要pom.xml來設定專案的版本、定義要用到的插件還有專案自動建置時運作的順序(script)，這裡我們提供兩種方法來產生pom.xml，第一種是一開始就產生一個基本的pom.xml，具備基本運行規則，出題老師若須要而外的功能則須修改pom.xml來達到需求。第二種方法是在出題之前勾選系統列出的功能中所需要的部分，由系統自動生成pom.xml。



圖五 基本環境架構

第二階段會製作SonarQube Report Analyzer，能解析SonarQube的報告內容，並把系統會需要使用到的資訊儲存到系統資料庫，以及製作能生成建議程式碼的Suggestion Code Generator，以下敘述各個元件的實作方式。

(1)Suggestion Code Generator：

目前計畫參考[4]的實作方式，在其研究中提供了SonarQube中最常被手動更正的11個靜態代碼分析規則，依據這些規則，我們會製作每條規則的代碼轉換模板(code-transformation templates)，利用這些模板套用到使用者的程式碼中，即可產生建議程式碼。而在系統確實產生建議程式碼前會先針對SonarQube檢測產生的結果進行檢查，若確實違反了其中某項規則，才會套用模板。

以下為Suggestion Code Generator運作方法：

a.針對SonarQube的報告內容偵測是否違反預先決定的11條規則。

b.若有違反規則，找出使用者程式碼中違反規則的程式碼片段。

c.套用代碼轉換模板，產生建議程式碼。

以下為論文[4]提供的C8規則的代碼轉換模板(code-transformation templates)：

refactoredExp = parse(#Expression, "<beforeFunc>.isEmpty()");

此代碼轉換模板能套用在違反規則C8的程式碼片段上，能將Collection.size()==0替換為Collection.isEmpty(),如此便可消除SonarQube品質檢測所產生的issues。

(2)SonarQube Report Analyzer：

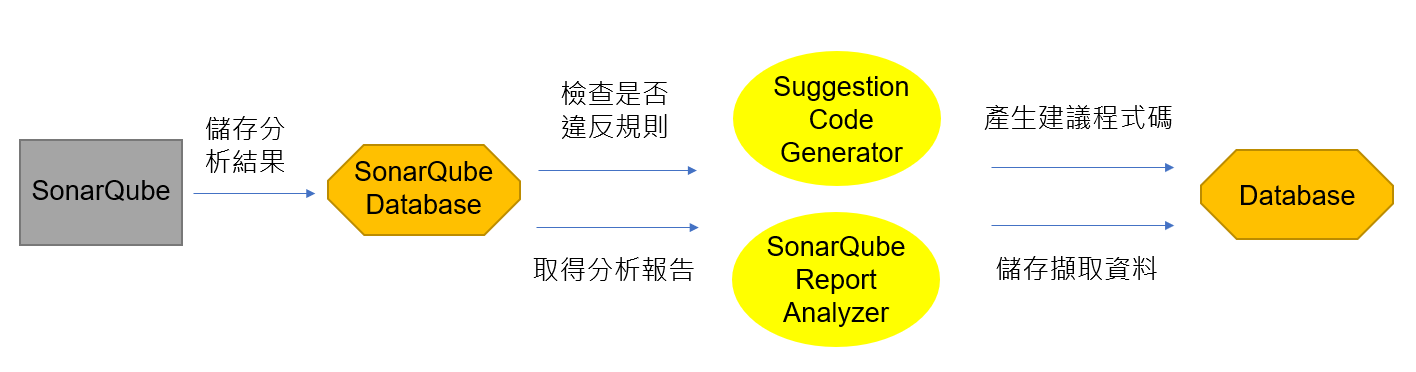
目前計畫SonarQube Report Analyzer會擷取包含三部分：

a.三項指標(安全性，可靠性，可維護性)的評估等級

b.報告中較嚴重的issue(Major issues, Critical issues, Blocker issues)

c.程式碼產生的Bugs數目，Vulnerability數目，及code smell的數目

目前預計利用SonarQube提供的的web API中的Get(內容位於api/measures/ component及api/issues/search中)將上述三項(a, b, c)的內容提出，並且利用Restful API中的Post，將需要的資料存入系統的資料庫中。



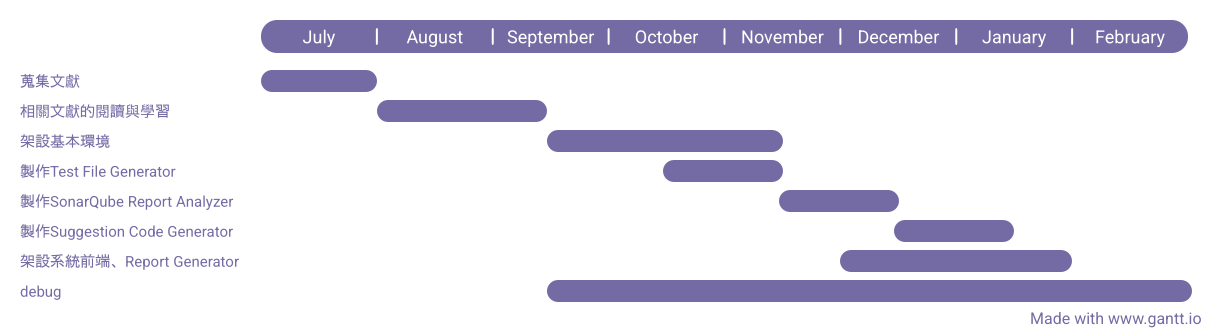
圖六 階段二內部資料流

第三階段，我們會開始著手處理前端，及Report Generator的製作。

前端的部分會將重點放在儀表板上，儀表板會顯示該堂課老師出了哪些作業、各個作業所有學生的繳交狀況，點進該作業自己的名字能看到自己每次作業建置的狀況和報告。

Report Generator 會從系統的資料庫中，抓取每次繳交產生的Bugs數目，vulnerability數目及code smell的數目，分別對應到安全性，可靠性及可維護性，利用資料庫中各項目歷次的數目來繪製三張曲線圖，並且Report Generator 會將本次建置的分數，三張曲線圖，及程式碼的修改建議彙整成一份報告傳送至儀表板，讓學生了解作業的作答情況，老師也能查看學生作業繳交狀況。

以下是預計的計畫甘特圖:



圖七 預計進度-甘特圖

(五)預期結果

預期功能:

(1) 提供單元測試以及品質檢測。

(2) 根據學生程式碼錯誤提供對應的建議。

(3) 老師在出題時可以自由選擇自己出題或是從題庫出題。

(4) 提供品質曲線圖以便學生了解進步狀況。

(5) 報告結果整合到儀表板上。

(六)參考文獻

[1] Hsi-Min Chen, Wei-Han Chen, Nien-Lin Hsueh, Chi-Chen Lee and Chia-Hsiu Li, "ProgEdu - an automatic assessment platform for programming courses, " 2017 International Conference on Applied System Innovation (ICASI), 2017, pp. 173-176, doi:10.1109/ICASI.2017.7988376

[2] Pulasthi Perera, Roshali Silva, and Indika Perera, "Improve Software Quality through Practicing DevOps" 2017 International Conference on Advances in ICT for Emerging Regions (ICTer): 013 - 018

[3] Automated quality characteristic measures. Accessed:2022-01-28. url:[https://it-cisq.org/standards/automated-quality-characteristic-measures/.](https://it-cisq.org/standards/)

[4] Diego Marcilio,Carlo Alberto Furia,Rodrigo Bonifacio,Gustavo Pinto "Automatically Generating Fix Suggestions in Response to Static Code Analysis Warnings" 2019 DOI:10.1109/SCAM.2019.00013

[5] SCORM,url:[SCORM.com HomePage: What is SCORM and How it Works](https://scorm.com/)

[6] Docker docs url:<https://docs.docker.com/get-started/overview/>

[7] GitLab官方網站 url:<https://about.gitlab.com/stages-devops-lifecycle/>

[8] SonarQube url:[Code Quality and Code Security | SonarQube](https://www.sonarqube.org/)

(七)需要指導教授指導內容

本研究計畫預期需要指導教授指導的內容如下：

(1)研究計畫相關方向，內容執導。

(2)引導使用研究計畫相關之文獻。

(3)成果報告撰寫。

(4)進度規劃與審查