# VLOL Application Development Turnover

Sefanit Urgessa

Mohammed Allibalogun

Robert Garcia

Meron Getachew Debela

Charles Baisie

Augustin Mwamba

Jamal Bourne

SWEN 670 Software Engineering Project

University of Maryland Global Campus

Date: 08/04/2020

**Table of Contents**

[1. Introduction 3](#_Toc47467233)

[2. Background 3](#_Toc47467234)

[4. Current Situation 5](#_Toc47467235)

[4.1. Key Locations: 5](#_Toc47467236)

[4.2. Requirements Overview 5](#_Toc47467237)

[4.3. Requirements Breakdown 6](#_Toc47467238)

[5. Outstanding Issues 10](#_Toc47467239)

[6. Summary 11](#_Toc47467240)

**Table Of Figures**

[Figure 1. VLOL Application Landing Page 3](#_Toc47467320)

[Figure 2. Requirement Fulfilment Breakdown 5](#_Toc47467321)

[Figure 3. SonarQube Results 11](#_Toc47467322)

## Introduction

The purpose of this document is to provide subsequent teams with a starting point to continue development of the Virtual Letter of Life (VLOL) application. The scope of this document covers the actions taken and their results during the period between May 20, 2020 to August 11, 2020. New teams are encouraged to publish a similar document following their development cycle and include this paper with their documentation.



Figure 1. VLOL Application Landing Page

## Background

On May 20, 2020, the Salisbury Fire Department published a Request for Proposal (RFP) through the City of Salisbury's GovQuote web portal. Team 3, SWEN 670 (Summer), at the University of Maryland, Global Campus (UMGC), responded to the proposal the next day.

The Salisbury Fire Department's Business Problem and Desired Outcomes are:

*"We are searching for a system that will allow emergency medical services (EMS) to retrieve the medical information contained in a patient's "Letter of Life" at any location.*

*"In an emergency, the Letter of Life provides responders with a list of pertinent medical conditions and illnesses when the patient cannot provide this information on their own. We recommend all residents, especially the elderly and those suffering from chronic medical conditions, to complete a "Letter of Life" and place it in a location easily accessible by EMS personnel, such as on a refrigerator or by the entry door to the residence. However, when an emergency occurs outside the home, EMS personnel cannot access this list, which is why we are searching for an easily accessible, cloud-based version of the letter.*

*"Our desired outcomes are:*

* *Provide EMS personnel access to a patient's Letter of Life information, from a laptop and mobile device, through a secure, but easily navigable interface.*
* *Allow EMS personnel to scan a barcode or quick response (QR) code, found on a non-electronic token, such as a card, a bracelet, or necklace, and retrieve Letter of Life information for non-responsive patients.*
* *Allow patients and authorized users to create, read, update, and delete records, from a desktop or laptop computer, through a secure, but easily navigable administration interface.*
* *Allow patients and authorized users to print Letters of Life or save them in portable data format (PDF).*

The first phase of the project, requirements elicitation, began on May 27, 2020, with an initial meeting with the subject matter expert (SME). Shortly afterwards, the SME and the team defined an initial set of user stories, and development began. However, in mid-June, based on input from the DevOps team, the team switched solution stacks from Windows/IIS/SQL Server/C#, using the .NET Core framework, to a Linux/Tomcat/MySQL/Java, using Spring Boot.

Over the next six weeks, the team set out to meet each user story, starting with a basic Spring Create, Read, Update, and Delete (CRUD) framework and data model, and implementing security, styling, and other features, such as retrieving participant information using QR scanning. This turnover snapshot provides a status report of the requirement fulfillment to date, as well as known issues.

1. **Timeline Highlights**

**Note** - Refer to the project’s plan in the project documentation for a more detailed overview of the schedule and cost of developing the application.

* May 27, 2020 – Initial Meeting
* May 27, 2020 – Initial user stories defined
* June 9, 2020 – Project Plan and Software Requirements Specification (SRS) submitted and approved
* June 30, 2020 – Technical Design Document (TDD) and Test Plan submitted and approved
* July 21, 2020 – Programmer’s Guide and Application Runbook submitted and approved
* August 4, 2020 – User Guide and Test Report submitted and approved
* August 7, 2020 – Final Presentation to the Client and Program Manager

## Current Situation

### Key Locations:

* Repository Location: [https://github.com/garciart/SWEN670](https://nam03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fgarciart%2FSWEN670&data=02%7C01%7Crgarcia92%40student.umgc.edu%7C0553835205784c3e90d608d838b57f85%7C704ce3d6a4bf4e098516d52840c9f7a9%7C0%7C0%7C637321699488313962&sdata=uLy7NbLs6nPwILDAApfnhB34SkQYTi8DuPoNUu%2B2ryA%3D&reserved=0). The code may also be found at [https://github.com/umgc/fire.department](https://nam03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fumgc%2Ffire.department&data=02%7C01%7Crgarcia92%40student.umgc.edu%7C0553835205784c3e90d608d838b57f85%7C704ce3d6a4bf4e098516d52840c9f7a9%7C0%7C0%7C637321699488313962&sdata=js93NVSjDCsy1rYH1Tdh%2F%2FSZWZ6m%2F2%2B7O0k0jjcIVnU%3D&reserved=0).
* Documentation Location: <https://onedrive.live.com/?authkey=%21AOw2vxm7lnY4Rmk&id=D159C6583F3538AF%21275&cid=D159C6583F3538AF>
* Development Site Location: [https://warm-hamlet-22515.herokuapp.com/](https://nam03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwarm-hamlet-22515.herokuapp.com%2F&data=02%7C01%7Crgarcia92%40student.umgc.edu%7C0553835205784c3e90d608d838b57f85%7C704ce3d6a4bf4e098516d52840c9f7a9%7C0%7C0%7C637321699488323955&sdata=FvqfAf1b8ON69JwzASsKuZ%2BceeCwzM%2F2Innf61RvjLU%3D&reserved=0) (see the HEROKU markdown file for instructions on how to deploy the application to the live server)

### Requirements Overview

* Per the SRS, there were 57 total requirements. The team completed 37; partially completed 8; and did not start 12 during the 12-week course.

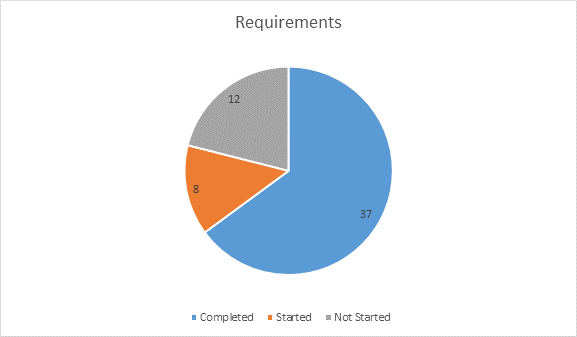


Figure 2. Requirement Fulfilment Breakdown

**Note –** Before continuing, access the application and execute the directions contained in the WALKTHROUGH markdown file or the User Guide. This action will provide context for the following requirements.

### Requirements Breakdown

Partially Completed Not yet Implemented

* **Unauthenticated and Unauthorized User (3/4 requirements completed, 1 partially complete)**

1. As an unauthenticated and unauthorized user, I want access to an informative Landing page, so I can learn about the Letter of Life (LOL) program.
2. As an unauthenticated and unauthorized user, I want access to an informative About page, so I can learn about the LOL program.
3. As an unauthenticated and unauthorized user, I want access to a Contact page, so I can solicit information about the LOL program or contact the System Administrator about an issue – ***The contact page only contains a mail:to hyperlink. The intent is to add a form, fillable by a visitor, which would be forwarded to a system administrator account via a Simple Mail Transfer Protocol (SMTP) service.***
4. As an unauthenticated and unauthorized user, I want access to a secure Registration page, so I can enroll in the Letter of Life program.

* **Program Participant (4/7 requirements completed, 3 not started)**

1. As a registered, authenticated, and authorized Program Participant, I want to access my LOL via a Login Page, so unauthenticated and unauthorized users cannot access my medical history.
2. As a registered, authenticated, and authorized Program Participant, I want to add my LOL information to the database via an online form, so medical personnel can have access to my medical history in case of an emergency.
3. As a registered, authenticated, and authorized Program Participant, I want to view my LOL information via an online form, so I can validate my information or print my LOL.
4. As a registered, authenticated, and authorized Program Participant, I want to edit my LOL information via an online form, so I can ensure my medical history is up to date.
5. As a registered, authenticated, and authorized Program Participant, I want to delete my LOL information from the database via an online form, so I may stop participating in the program – ***Not yet implemented.***
6. As a registered, authenticated, and authorized Program Participant, I want to print a barcode or quick response (QR) code with a link to my LOL (which I may affix to a bracelet, card, or necklace), so medical personnel can quickly access my medical history in case of an emergency – ***Not yet implemented.***
7. As a registered, authenticated, and authorized Program Participant, I want access to a secure Contact page, so I can contact my Patient Advocate (if designated) and/or the System Administrator about an issue – ***Not yet implemented.***

* **Medical Services Provider (5/7 requirements completed, 2 not started)**

1. As a registered, authenticated, and authorized Medical Services Provider, I want to access the LOL system via a Login Page, so unauthenticated and unauthorized users cannot access a patient's medical history.
2. As a registered, authenticated, and authorized Medical Services Provider, I want to edit my profile information via an online form, so I can ensure my information is up to date.
3. As a registered, authenticated, and authorized Medical Services Provider, I want to search for patients on a laptop or mobile device via an online form, so I can retrieve their medical history and provide correct medical care.
4. As a registered, authenticated, and authorized Medical Services Provider, I want to search for a patient by scanning a barcode or quick response (QR) code, using a mobile device or authorized accessory, so I can retrieve their medical history and provide correct medical care.
5. As a registered, authenticated, and authorized Medical Services Provider, I want to view a patient's medical history on a laptop or mobile device via an online form, so I can provide correct medical care.
6. As a registered, authenticated, and authorized Medical Services Provider, I want to print a MIEMSS-approved Short Form, populated with my patient's information, so I can provide non-system medical facilities and staff with my patient's medical history – ***Not yet implemented.***
7. As a registered, authenticated, and authorized Medical Services Provider, I want access to a secure Contact page, so I can contact the System Administrator about an issue – ***Not yet implemented.***

* **Patient Agent (5/8 requirements completed, 1 partially complete, 2 not started)**

1. As a registered, authenticated, and authorized Patient Agent, I want to access the LOL system via a Login Page, so unauthenticated and unauthorized users cannot access the medical history of my patients.
2. As a registered, authenticated, and authorized Patient Agent, I want to edit my profile information via an online form, so I can ensure my information is up to date.
3. As a registered, authenticated, and authorized Patient Agent, I want to search for patients on a laptop or mobile device via an online form, so I can view, edit, and print patient information, as necessary – ***Patient agents can currently see every user in the system. The plan is to limit agent access to those records which contain the agent’s ID number in the user’s agent ID field.***
4. As a registered, authenticated, and authorized Patient Agent, I want to search for a patient by scanning a barcode or quick response (QR) code, using a mobile device or authorized accessory, so I can view, edit, and print patient information, as necessary.
5. As a registered, authenticated, and authorized Patient Agent, I want to view the LOL information of my patients via an online form, so I can validate their information or print their LOL.
6. As a registered, authenticated, and authorized Patient Agent, I want to edit the LOL information of my patients via an online form, so I can ensure their medical history is up to date.
7. As a registered, authenticated, and authorized Patient Agent, I want to print a barcode or quick response (QR) code with a link to my patient's LOL (which I may affix to a bracelet, card, or necklace), so medical personnel can quickly access their medical history in case of an emergency – ***Not yet implemented.***
8. As a registered, authenticated, and authorized Patient Agent, I want access to a secure Contact page, so I can contact the System Administrator about an issue – ***Not yet implemented.***

* **System Administrator (13/14 requirements completed, 1 partially complete)**

1. As a registered, authenticated, and authorized System Administrator, I want to access the LOL system via a Login Page, so unauthenticated and unauthorized users cannot access patient medical history or user account information.
2. As a registered, authenticated, and authorized System Administrator, I want to search for user accounts on a laptop or mobile device via an online form, so I can add, view, edit, and delete user accounts, as necessary.
3. As a registered, authenticated, and authorized System Administrator, I want to add accounts for Medical Services Providers and Patient Advocates, so I can limit non-patient access to patient information.
4. As a registered, authenticated, and authorized System Administrator, I want to view accounts for Medical Services Providers and Patient Advocates, so I can validate their information.
5. As a registered, authenticated, and authorized System Administrator, I want to edit accounts for Medical Services Providers and Patient Advocates, so I can ensure their information is up to date.
6. As a registered, authenticated, and authorized System Administrator, I want to delete accounts for Medical Services Providers and Patient Advocates, so I can limit non-patient access to patient information.
7. As a registered, authenticated, and authorized System Administrator, I want to search for patients on a laptop or mobile device via an online form, so I can add, view, edit, and delete patient information, as necessary.
8. As a registered, authenticated, and authorized System Administrator, I want to add accounts for patients, so I can enroll them in the program.
9. As a registered, authenticated, and authorized System Administrator, I want to view accounts for patients, so I can validate their information.
10. As a registered, authenticated, and authorized System Administrator, I want to edit accounts for patients, so I can ensure their information is up to date.
11. As a registered, authenticated, and authorized System Administrator, I want to delete accounts for patients, so I can disenroll them from the program.
12. As a registered, authenticated, and authorized System Administrator, I want to view a system error log, so I can inform the development team of problems or issues with the application – ***The system logs activity and errors in one file. The intent is to have two separate logs, one for activity and one for errors.***
13. As a registered, authenticated, and authorized System Administrator, I want to view a system activity log, so I can audit access and use of the system.
14. As a registered, authenticated, and authorized System Administrator, I want to a user guide, so I provide users with instructions on how to use the system.

* **Non-Functional Requirements: (8/18 requirements completed, 5 partially complete, 5 not started)**

1. As a system, I want to limit Create-Read-Update-Delete (CRUD) access to authenticated and authorized users only, so I may preserve the confidentiality and integrity of Personally Identifiable Information (PII) – ***While Thymeleaf security prevents users from accessing functionality on client pages based on role, any authenticated user can access any page at this time. The intent is to limit access to pages using routing in the WebMvcSecurity class.***
2. As a system, I want new registrants to verify their accounts via a link sent via email (2FA), so I may mitigate the creation of fake accounts – Not yet implemented.
3. As a system, I want all non-verified accounts to be deleted within 24 hours, so I may prevent database bloating – ***Not yet implemented.***
4. As a system, I want all user input validated, so I may prevent injection and scripting attacks.
5. As a system, I want user sessions to timeout within 30 minutes, so I may prevent inadvertent disclosure of PII – Not yet implemented.
6. As a system, I want to limit multi-account access to authorized devices only (MAC Whitelisting), so I may preserve the confidentiality and integrity of PII, as well as the availability of the system – ***Not yet implemented.***
7. As a system, I want the following NIST SP 800-53 controls enforced throughout the application, so I may preserve the confidentiality and integrity of PII, as well as the availability of the system:
   1. AC-2: ACCOUNT MANAGEMENT
   2. AC-3: ACCESS ENFORCEMENT – ***As stated earlier, authenticated user can access any page at this time. The intent is to limit access to pages using routing in the WebMvcSecurity class.***
   3. AC-7: UNSUCCESSFUL LOGON ATTEMPTS – Not yet implemented.
   4. AC-8: SYSTEM USE NOTIFICATION
   5. AC-11: SESSION LOCK
   6. AC-12: SESSION TERMINATION
   7. AU-2: AUDIT EVENTS – ***As stated earlier, the system logs activity and errors in one file. The intent is to have two separate logs, one for activity and one for errors.***
   8. AU-8: TIME STAMPS
   9. IA-2: IDENTIFICATION AND AUTHENTICATION (ORGANIZATIONAL USERS)
   10. SC-13: CRYPTOGRAPHIC PROTECTION
   11. SC-23: SESSION AUTHENTICITY
   12. SI-10: INFORMATION INPUT VALIDATION
   13. SI-11: ERROR HANDLING – ***Currently, errors are collected, and users are redirected to a custom error page (the stack trace of the error is displayed if the user is an administrator). However, as stated earlier, the system logs activity and errors in one file. The intent is to have two separate logs, one for activity and one for errors. In addition, errors are not forwarded to a system administrator account via a Simple Mail Transfer Protocol (SMTP) service.***
8. As a system, I want all SQL queries to use prepared statements, so I can prevent SQL injection attacks.
9. As a system, I want all tables to be normalized to 3rd Normal Form, so I can improve data integrity and reduce data redundancy.
10. As a system, I want to unit test all functions and methods, so I can ensure boundary conditions are not violated – ***The team prioritized creating unit tests for all model methods and their edge conditions, to prevent SQL injections or storage of bad data. However, all methods in the configuration, controller, repository, and service folders need a unit test.***
11. As a system, I want to incorporate static code analysis, using both automated and manual methods, so I can ensure there are no weaknesses due to code syntax.
12. As a system, I want to incorporate dynamic code analysis and penetration testing, so I can ensure there are no vulnerabilities due to code semantics.
13. As a system, I want to style and comment all code per the appropriate style guide, so I can properly maintain and turn-over the application, as necessary – ***Class header comments per the Google Style Guide are complete, but most methods do not have JavaDoc comments and a JavaDoc was not generated.***
14. As a system, I want to incorporate version control, so I may audit and track code generation.
15. As a system, I want all CRITICAL, HIGH, MEDIUM, and STANDARD issues and risks, identified during static and dynamic analysis, corrected before uploading to a repository, so I can ensure the repository's code meets the bug bar – **See the list of outstanding vulnerabilities below.**
16. As a system, if a CRITICAL, HIGH, MEDIUM, and STANDARD issue and risk cannot be corrected immediately, I want it recorded in a bug log, so I can ensure the issue or risk is corrected before deployment.
17. As a system, I want all presentation code to be HTML5 and W3C compliant, so I can ensure the application provides the same functionality across different platforms – **Not yet implemented.**

### Outstanding Issues

The following is an overview of the current issues identified by SonarQube scanning, adjusted by the team to reflect ground truth (see the SONARQUBE markdown file in the repository for instructions on how to use the scanner with Maven). A full description of each issue, broken down by category, is included in the test report.

* Lines of Code Analyzed: 2,200
* Total Issues: 151
* Current Technical Debt Owed to Quality Improvement: 28 hours
* Bugs: 0
* Potential Vulnerabilities: 29
* Potential Security Hotspots: 30
* Potential Code Smells: 122
* Unit Tests: 189
* Unit Test Coverage: SonarQube reports 0% code coverage, but the 189 tests protect the five Model classes, which account for 216 of the 245 total methods, for 88.1% method coverage. These counts include untestable code that do not return values or throw exceptions.
* Duplications: 4.5%

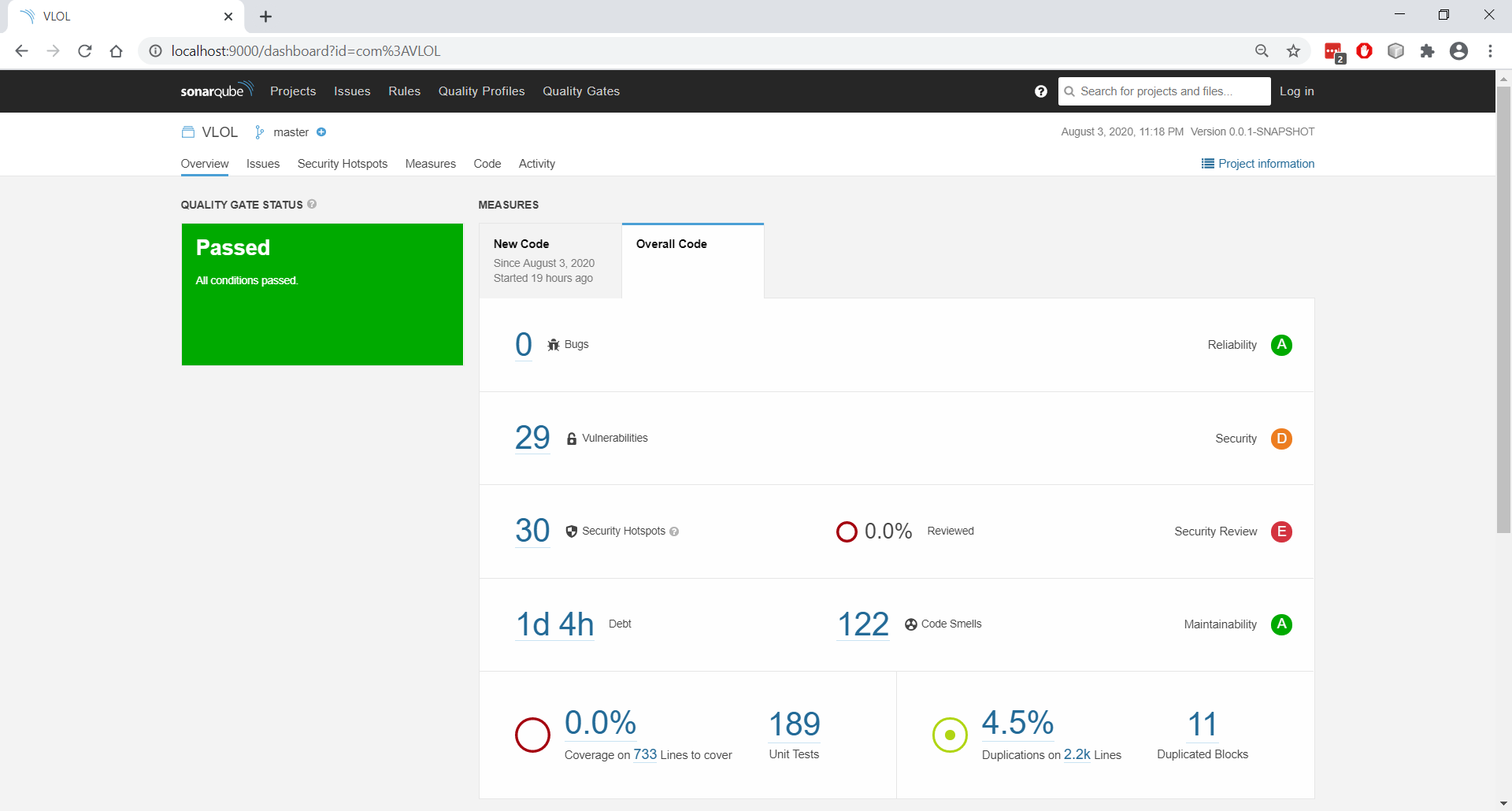


Figure 3. SonarQube Results

### Summary

While the team made considerable progress and the application is functional on both desktop and mobile browser, the amount of time allotted to the project was not enough to fulfil each of the requirements. This document, in conjunction with the other documents produced for this project, is meant to provide a subsequent team with key information, which will allow them to quickly continue development.