Programmer’s Guide

Virtual Letter of Life (VLOL) Application

Version 1.0

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: 07/21/2020

Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 07/09/2020 | 1.0 | Initial Programmer’s Guide Release by Augustin Mwamba.  Reviewed by EMSPlus internal team. |
| 07/21/2020 | 1.1 | Reviewed and approved by EMSPlus internal team |
|  |  |  |
|  |  |  |

Table of Contents

[1. Introduction 1](#_Toc45933878)

[1.1. Overview 1](#_Toc45933879)

[1.2. Scope 1](#_Toc45933880)

[1.3. References 1](#_Toc45933881)

[1.4. Acronyms 1](#_Toc45933882)

[2. Context 2](#_Toc45933883)

[2.1. Architecture Guiding Principles 2](#_Toc45933884)

[2.2. Architecture Constraints and Assumptions 4](#_Toc45933885)

[2.2.1. Constraints 4](#_Toc45933886)

[2.2.2. Assumptions 4](#_Toc45933887)

[2.2.3. Alignment with Priority Areas 5](#_Toc45933888)

[3. Data 5](#_Toc45933889)

[3.1. Storage 5](#_Toc45933890)

[3.2. Data Integrity 5](#_Toc45933891)

[3.3. Data Archiving 5](#_Toc45933892)

[4. Software 5](#_Toc45933893)

[4.1. Business Logic 7](#_Toc45933894)

[4.2. User Interface 9](#_Toc45933895)

[5. Technical Positions 9](#_Toc45933896)

[6. Coding Standards 10](#_Toc45933897)

# Introduction

## Overview

This document describes the Programmer’s guide for the Virtual Letter of Life (VLOL) system. It outlines key principles and guidelines for developing and upgrading the VLOL software, and describes important features that are needed in the design and implementation of the VLOL application.

## Scope

The purpose of this programmer’s manual is to help streamline software development for the VLOL system and to make future software modification easier. By using this programmer’s reference manual, software developers working on the VLOL system will understand how the application is constructed, and how to make modifications or improvements to the existing code when necessary. Specifically, the manual provides information on the following:

* An overview of the architecture of the VLOL system and the rationale behind it.
* Programming guidelines and desired program features for the VLOL software.
* Software development utilities, and libraries available for developing the VLOL system.

## References

1. Bass, L., Clements, P., & Kazman, R. (2012). Software architecture in practice. Upper Saddle River, NJ: Addison-Wesley Professional.
2. Cervantes, H., & Kazman, R. (2016). *Designing Software Architectures: A Practical Approach*. Boston, MA: Addison-Wesley.
3. EMSPlus team (2020). *Virtual Letter of Life (VLOL) Software Design Document*.
4. EMSPlus team (2020). *Virtual Letter of Life (VLOL) Software Requirements Specification Document*.
5. EMSPLus team (2020). *Virtual Letter of Life (VLOL) User Interface Design Document*.
6. IEEE Std 14764-2006 (2006). Software Engineering, Software Life Cycle Processes, Maintenance. Retrieved from <https://standards.ieee.org/standard/14764-2006.html>
7. Microsoft Patterns & Practices team (2009). *Microsoft Application Architecture Guide*. Microsoft Press.
8. Pressman, R. S. (2014). Software engineering: A Practitioner's Approach. New York, NY: McGraw-Hill Education.
9. Sommerville, I. (2016). Software engineering. Essex, England: Pearson Education Limited.
10. The Section 508 Standards. Retrieved from [The Section 508 Standards](file:///C:\Users\augustin\AppData\Roaming\Microsoft\Word\The%20Section%20508%20Standards)

## Acronyms

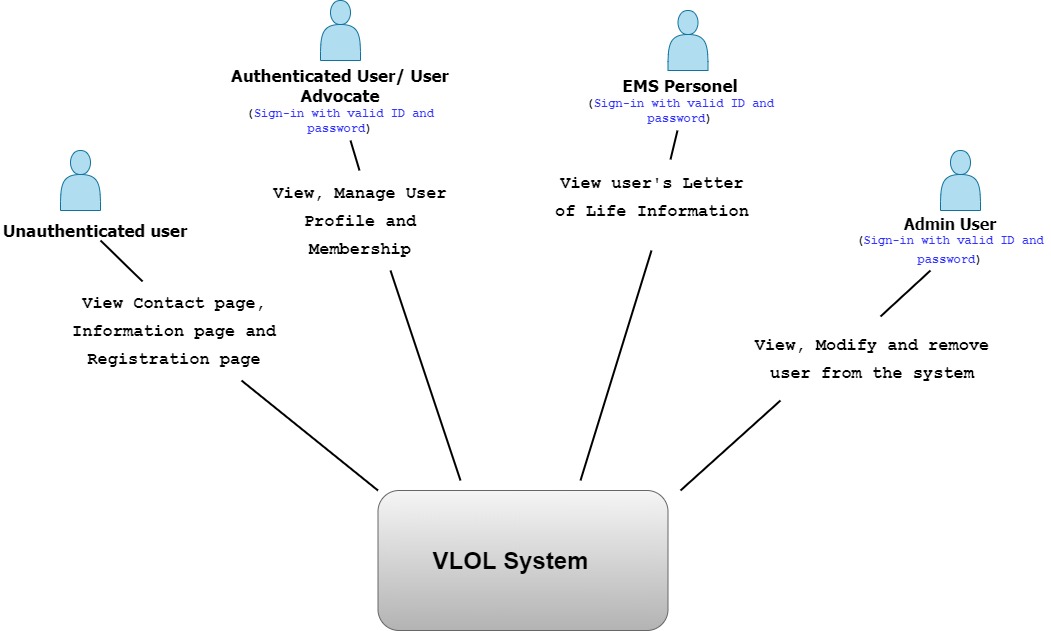
The following is a glossary of terms used in this document:

* **DB** – Database
* **HIPPA** - Health Insurance Portability and Accountability Act
* **JDBC** - Java Database Connectivity
* **MVC** - Model View Controller
* **ORM** - Object Relational Model
* **SDD** - Software Requirements Document
* **SRS** - Software Requirements Specification
* **VLOL** - Virtual Letter of Life

# Context

The VLOL system main purpose is to provide users and first responders with a web-based version of the letter of life. At its core, the application allows authenticated users to retrieve User’s Letter of Life information from a data source (an external database) using a desktop browser or a mobile device, and to create, update and delete records based on roles and privileges they are granted within the system. Unauthenticated users (users without an account or credentials) can access the application’s contact page, information page and registration page.

The context diagram shown in **Figure 1** illustrates the basic functionality of the VLOL software.



**Figure 1**. VLOL system basic functionality

The ideal solution that satisfies the system’s functional requirements, from a perspective of the four groups of users (Figure 1), is a software architecture that not only adheres to proven principles, but also minimizes costs and maintenance requirements, and promotes overall system usability and extendibility.

The remainder of this section describes the guiding principles and rationale behind the architectural choice made for the VLOL software, known constraints and assumptions for this architecture, and the architecture alignment with business priorities.

## Architecture Guiding Principles

The VLOL system is based on a light weight MVC (Model View Controller) design pattern consisting of a data model, a view (user interface), and controllers (processes that handle input).

The development has adopted the following design decisions in order to satisfy the system’s functional requirements and quality attributes:

* Logically structure the VLOL system as a web and mobile application, and use the Separation of Concerns principle to organize the system into software layers based on their functionality (presentation layer, business logic, and data access layer).

The rationale behindthis design decision is that, the VLOL system is primarily intended to be accessed from web browsers and deployed on mobile devices such as smart phones, tablets, etc., therefore using a web application or mobile application architecture would meet the system’s functional requirements and quality attributes.

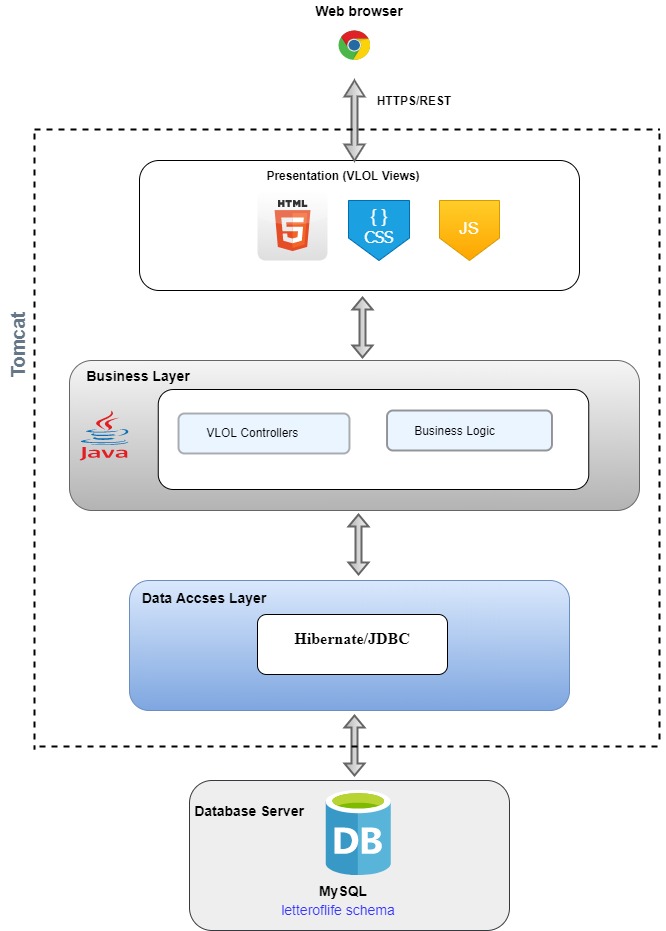
***Pros***: Web applications can be accessed from a variety of platforms using a standard web browser, they have fast page loading time and are simple to deploy. Mobile applications, for their part, support a rich user interaction.

***Cons***: Web applications do not support rich user interaction, and mobile applications are less portable and have screen limitations.

***Discarded Alternative***: Rich Internet Application (RIA)

The RIA option was discarded because, although this type of application supports a rich user interface and can be easily deployed or upgraded, the development team believed that the plugins (runtime environment) for executing RIA were less available than the Java Virtual Machine (JVM).

**Figure 2** shows a logical or layered architectural view of the VLOL system.



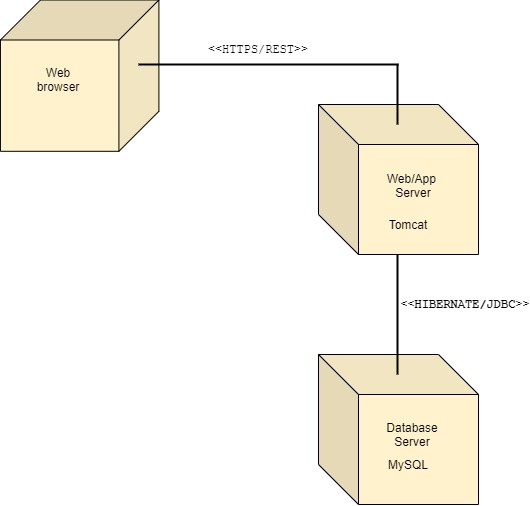
**Figure 2**. VLOL system layered view

* Physically structure the application using a three-tier deployment pattern with Client, Business, and Data tiers.

***Rationale***: The system must be accessed from a web browser and a mobile device, and an existing database (Relational database) must be used.

The benefit of this approach is that there is less performance cost compared to a 2-tier deployment scheme, for example.

The following diagram shows the deployment pattern of the VLOL system



**Figure 3**. VLOL system Deployment diagram

* Select the Spring Security framework to manage authentication and authorization: The needs for this application also include managing user authentication and authorization. All these features are available in the Spring Security framework which can easily integrate with other frameworks such as Java.

Other alternatives, such as ad hoc code, were also considered, but they are challenging, error-prone and take significant time to develop.

In summary, the development team believes that the functional decomposition of the system and service components, and their distributed deployment will provide improved scalability, availability, manageability, and resource utilization for the newly created VLOL system.

## Architecture Constraints and Assumptions

Development of the VLOL software will follow the design roadmap used for greenfield systems for mature domains, since it’s a new system that is being developed from scratch and there is an already established infrastructure of tools and technologies and associated knowledge base for this type of systems.

The following constraints and assumptions are identified to have a potential to properly constrain and bound the development of the VLOL system:

### Constraints

**Application Constraints**: The VLOL application and database designs must be flexible, and extensible to satisfy future need for better performance, accessibility, scalability and availability. Therefore, existing data must never be negatively impacted by new changes or implementations.

**Development Constraints**: The VLOL system development must comply with industry standards including existing software patterns, coding standards and best practices. The development process should include a formal validation process to assure that the enhancements are reliable, functional, and scalable and meet business requirements.

**Technology Constraints**: The system shall be composed primarily of open source technologies (for cost reasons). For those components where the value/cost of using proprietary technology is much higher, proprietary technology may be used.

**Architectural Constraints**: The system must be accessed through a Web browser in different platforms (Windows, OSX, Linux, etc.) and via Mobile devices.

### Assumptions

**System Scalability**: It must be possible to scale the deployment of the VLOL system to a large number of users geographically dispersed across the US without a significant increase in effort or cost.

**System Usability**: The application user interface must be intuitive to use, must provide a good overall user experience, must be localized and globalized, and must be accessible to disabled users in accordance with the **508** standards.

**System Testability**: The VLOL system must be designed in a modular fashion to support testing. It must also provide instrumentation or implement probes for testing, enable mechanisms to debug output, and ways to specify inputs easily. Also, the system must design components that have high cohesion and low coupling to allow testability of components in isolation from the rest of the system. Proper testing tools should be used for unit testing and the execution of unit tests should be automated.

**System Availability**: The system must run 24 x 7 x 365, with overall availability of 99.9% and a down time of 0.1% (With the exception of planned outages or scheduled maintenance).

**System Security**: All communications must be authenticated and encrypted using strong and secured certificates.

**Reusability**: Modularization of components and functionality shall be used throughout the development of this software.

### Alignment with Priority Areas

The goal initial purpose of this software architecture is to help achieve development goals. Therefore, the development team believe that having a design will help drive development work, guide system construction, satisfy system requirements and ultimately prepare for an eventual software release.

# Data

This section of the Programmer’s Guide describes application data storage, security and integrity checking schemes. Additional information pertaining to the VLOL application data architecture can be found in the application Software Design Document (SDD).

## Storage

The VLOL system will store its data in a MySQL database server to minimize costs and save time. Access to the database server must be accomplished via the Hibernate Framework. Database connections must be standard MySQL connections over TCP/IP. HIPPA security and standards data access security measures must be observed at all times. The VLOL application data will be backed up, when necessary, using appropriate backup devices.

## Data Integrity

The VLOL system must ensure that application data is protected against attacks that try to steal or corrupt data, and protect the mechanisms used to gain access to the data source. The VLOL system will include sanitizing error and exception handling mechanisms so that data source information is not revealed, and using least privilege accounts to restrict privileges to only those needed to perform the operations required by the application.

Database access should be done through parameterized queries to prevent SQL injection attacks from succeeding. The system will prevent the use of string concatenation to build dynamic queries from user input data.

Database integrity checking and recovery software integrity checking should run as a

background task checking for signs of database corruption.

Provide definitions for all terms, acronyms, and abbreviations required to properly interpret how they are used in this document. Provide a reference to the project’s glossary document if necessary.

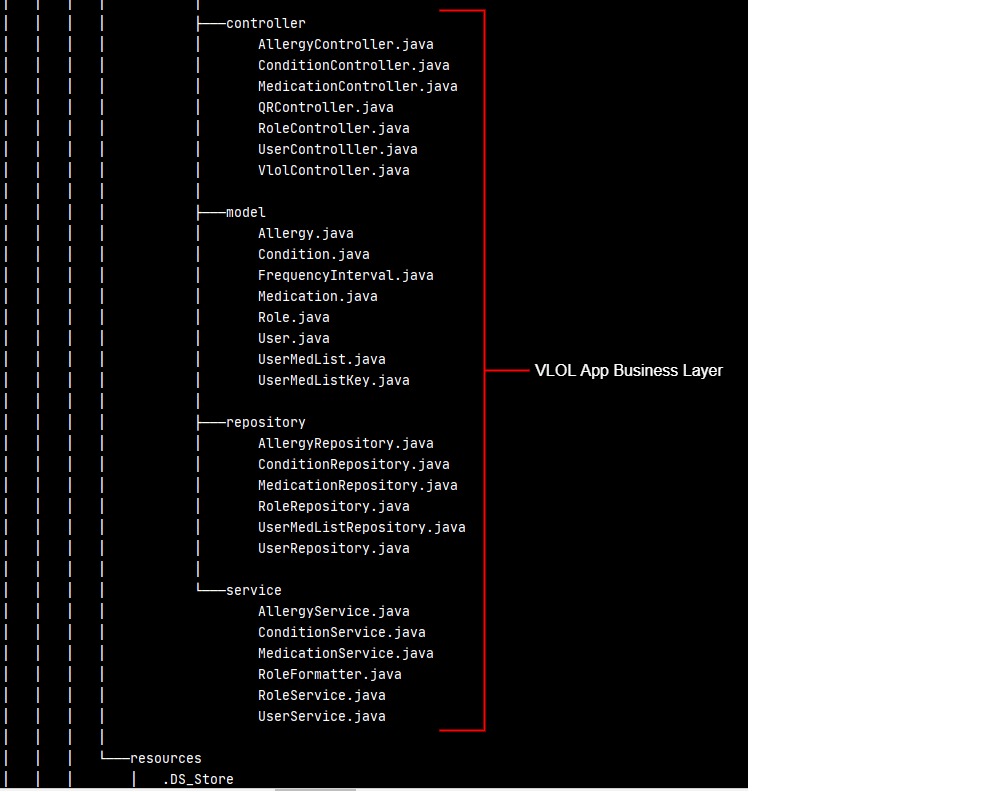
## Data Archiving

When necessary, application data will be archived in accordance with the best practices regarding data arching, and archived data will be stored to tape or disk.

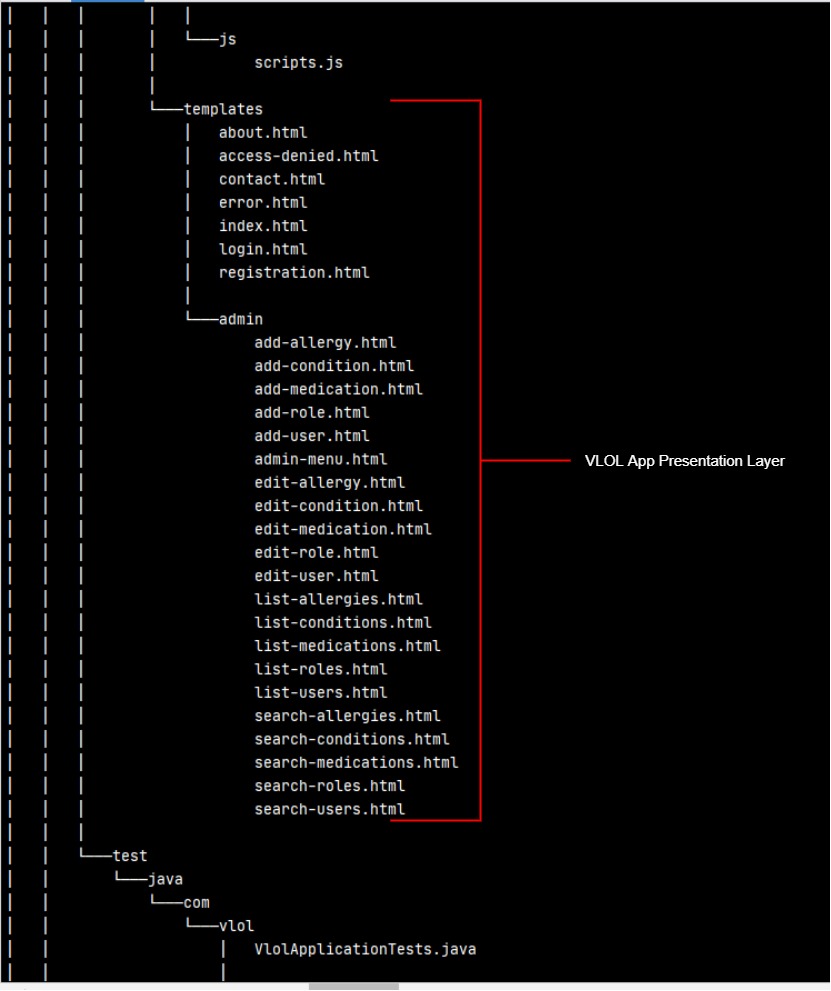
# Software

The VLOL software modules (source code, data files, executables, documentation etc.) are organized in terms of packaging, layering, and configuration management (ownership, release strategy etc.).

Figures 4 and 5 show the application layers, packages and package hierarchy. The package hierarchy starts with standard Java namespace compliant structure and then is divided into sub-packages for the different layers including *vlol.src.main.java.com.vlol* for the Business layer (controller, model, repository and service) and *vlol.src.main.resources.templates*, *vlol.src.main.reources.templates.admin* for the Presentation layer.



**Figure 4**: VLOL system implementation view (Business Layer).



**Figure 5**: VLOL system implementation view (Presentation Layer).

## Business Logic

Business logic is defined as any application logic that is concerned with the retrieval, processing, transformation, and management of application data; application of business rules and policies; and ensuring data consistency and validity. The business logic components shown in Figure 6 in the application Business Layer, contain behavior and application logic that is specific to the use cases (or user stories) described in the VLOL Software Requirements Specification Document (SRS). Each of the following component (class) contains a set of methods that represent business rules for the VLOL system:

**UserService.java:**

public User findUserByEmail(String email)

public User saveUser(User user)

public List<User> getAllUsers()

public User getUser(long userID)

public void deleteUser(long userID)

public List<User> findUserByKeyword(String keyword)

**RoleService.java:**

public List<Role> getAllRoles()

public void saveRole(Role role)

public Role getRole(long roleID)

public void deleteRole(long roleID)

public List<Role> findRoleByKeyword(String keyword)

**RoleFormatter.java:**

public String print(Role object, Locale locale)

public Role parse(String text, Locale locale) throws ParseException

**AllergyService.java:**

public List<Allergy> getAllAllergies()

public void saveAllergy(Allergy allergy)

public Allergy getAllergy(long allergyID)

public void deleteAllergy(long allergyID)

public List<Allergy> findAllergyByKeyword(String keyword)

**ConditionService.java:**

public List<Condition> getAllConditions()

public void saveCondition(Condition condition)

public Condition getCondition(long conditionID)

public void deleteCondition(long conditionID)

public List<Condition> findConditionByKeyword(String keyword)

**MedicationService.java:**

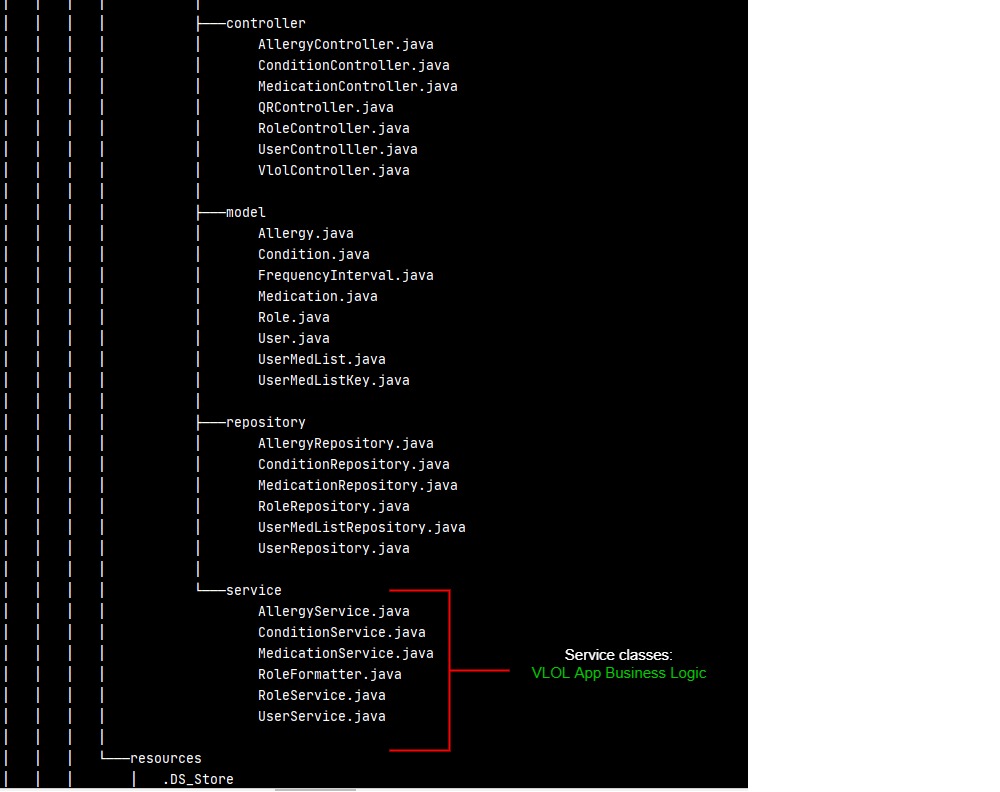
public List<Medication> getAllMedications()

public void saveMedication(Medication medication)

public Medication getMedication(long medicationID)

public void deleteMedication(long medicationID)

public List<Medication> findMedicationByKeyword(String keyword)



**Figure 6**: VLOL Business Logic Components.

## User Interface

The VLOL system user interface shall conform to usability guidelines for user interface design, particularly the Jakob Nielsen's 10 general principles for interaction design. It must be intuitive to use, must provide a good overall user experience, must be localized and globalized, and must be accessible to disabled users in accordance with the 508 standards.

Additionally, in order to provide rich feedback to the user, especially for errors and exceptions, and the system unresponsiveness, the VLOL system must also implement technologies and techniques that provide maximum user interactivity, such as Asynchronous JavaScript and XML (AJAX) in web pages and client-side input validation.

The reader is encouraged to consult the VLOL User Interface Design Document for additional information pertaining to the system user interface.

# Technical Positions

The VLOL system uses the following framework technologies to build its application and to interact with the underlying MySQL database:

* **Java SE 11.0.7**. Java is used as the core development technology**.** The system will be programmed in Java and Java-related technologies to leverage the expertise of the development team.
* **Spring Boot Framework**: Spring Boot will be used to provide a complete and modular framework for developing the VLOL application. It is a widely used framework in support of enterprise application development.

An alternative that was considered for developing the VLOL system is **JEE**. Spring was eventually selected because it is considered more “lightweight” and the development team is already familiar with it, which can lead to greater and earlier productivity.

* **Hibernate Framework**: Hibernate is an object to relational mapping (ORM) framework that integrates well with Spring. It offers to the VLOL application an ORM Framework specialized in data retrieval and persistence.

Other ORM frameworks were not considered, as the development team was already familiar with, and happy with the performance of Hibernate.

# Coding Standards

The development of the VLOL system will adopt the following industry naming and coding standards:

|  |  |
| --- | --- |
| **Standard** | **URL** |
| CERT Secure Coding Standards – Java Guidelines | <https://www.oracle.com/java/technologies/javase/seccodeguide.html> |
| Code Conventions for the Java TM Programming Language | <https://www.oracle.com/java/technologies/javase/codeconventions-contents.html> |
| Guidelines, Patterns, and code for end-to-end Java applications | <https://www.oracle.com/technetwork/java/namingconventions-139351.html> |
| Secure Coding Guidelines for Java SE | <https://www.oracle.com/java/technologies/javase/seccodeguide.html> |