Deployment and Operations Guide (Runbook)

Virtual Letter of Life (VLOL) Application

Version 1.3

Heather Barnes

Andrew Coleman

Panhavorn Hok

Michael Marcucci

Michael Shaw

Kimberly Van Allen

SWEN 670 Software Engineering Project

University of Maryland Global Campus

October 30, 2020

**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 10/10/2020 | 1.0 | Initial draft of Runbook |
| 10/16/2020 | 1.1 | Version incorporates initial guidance from Roy Gordon. |
| 10/19/2020 | 1.2 | Added details about security & production deployment |
| 10/30/2020 | 1.3 | Updated document to reflect current state of project |

**Table of Contents**

[1. Introduction 5](#_Toc54989137)

[1.1 Overview 5](#_Toc54989138)

[1.2 Scope and Purpose 5](#_Toc54989139)

[1.3 System Description 5](#_Toc54989140)

[1.4 Glossary 6](#_Toc54989141)

[2. Build 7](#_Toc54989142)

[2.1 Process 7](#_Toc54989143)

[2.2 File Structure 7](#_Toc54989144)

[3. Deployment 9](#_Toc54989145)

[3.1 Maintenance Tasks 9](#_Toc54989146)

[3.2 Production Deployment 9](#_Toc54989147)

[3.3 HIPAA Compliance 10](#_Toc54989148)

[4. Errors 11](#_Toc54989149)

[4.1 HTTP Status Errors 11](#_Toc54989150)

[4.2 Java Errors and Exceptions 12](#_Toc54989151)

[4.3 Known Issues 13](#_Toc54989152)

[5. Software Operation Instructions 13](#_Toc54989153)

[5.1 Installation 13](#_Toc54989154)

[5.2 Start, Stop, and Restart Procedures 13](#_Toc54989155)

[5.3 Security Breach Response 14](#_Toc54989156)

[6. Frequently Asked Questions 15](#_Toc54989157)

[8. License 15](#_Toc54989158)

[APPENDIX A: Work from Previous Teams 17](#_Toc54989159)

[I. Introduction 17](#_Toc54989160)

[II. Disaster Recovery/Service Level Agreement 17](#_Toc54989161)

**Table of Figures**

[Figure 1. VLOL project directory structure. 8](#_Toc54989162)

[Figure 2. Recommended architecture for hosting VLOL. 10](#_Toc54989163)

# 1. Introduction

## 1.1 Overview

This Runbook provides guidance for deploying and operating the Virtual Letter of Life (VLOL) system. This document outlines various aspects of running and maintaining the system, including build details, deployment procedures, maintenance tasks, possible error messages, known issues, disaster recovery, and general VLOL software operation.

## 1.2 Scope and Purpose

The scope of this document falls within the deployment and maintenance phases of the software development life cycle, and its overall purpose is to provide guidance to system administrators regarding building, deploying, and maintaining the VLOL software. As the VLOL system changes over time as a result of software upgrades, changes in client-sourced requirements, or other miscellaneous reasons, this document must be audited and updated as needed so that it reflects the current version of the VLOL software.

This document is intended to be read by two primary audiences: the VLOL development team, who are tasked with maintaining this document; and system administrators for servers running the VLOL software. To properly execute the tasks described in this document, it is recommended that the reader is familiar with the following languages, frameworks, systems, and software:

* Java
* JavaScript (ECMAScript 2020)
* SQL
* HTML 5
* CSS
* Spring Boot
* MySQL
* Git
* Maven

## 1.3 System Description

The Virtual Letter of Life system is an electronic, web-based adaptation of Letters of Life (also known as Files of Life). Letters of Life are most commonly implemented as paper forms, kept by members of the general public in their homes or on their persons, that provide a summary of their medical information, to be used by EMS personnel and other first responders when providing emergency care. The VLOL system aims to expand this practice by providing a secure interface for entering, updating, and accessing patients’ Letter of Life information, with the goal of making that information more up-to-date and more readily accessible by emergency personnel.

The VLOL software defines four user classes:

* **Program Participant:** Members of the general public who wish to store their Letter of Life information in the VLOL database, or medical staff such as doctors and nurse practitioners who have been granted read-only access by a specific user. These users may view, update, or delete their individual account information, including their password, Letter of Life information, and miscellaneous account information. If they have been granted read-only access to another account, they may view that data as well.
* **Medical Services Provider:** EMS personnel, emergency room staff, and other personnel who need to quickly access patients’ Letter of Life information. These users may search and view the medical information of program participants, but may not view the participants’ other account information. These users may also scan a valid QR code to obtain quick access to view a specific user’s account.
* **System Administrator:** IT personnel who have been authorized to administer and maintain the VLOL system. These users may add, search, view, update, or delete all user accounts, and have access to modify the medical information stored in users’ accounts.

## 1.4 Glossary

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

* **Azure:** also known as Microsoft Azure; a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.
* **CSS:** Cascading Style Sheets; a language used for describing the presentation of a document written in a markup language such as HTML.
* **Docker:** a software platform that uses operating system-level virtualization to deliver software in packages called containers.
* **EMS:** Emergency Medical Services.
* **Git:** a distributed version-control system for tracking changes in source code during software development.
* **H2:** a relational database management system written in Java.
* **HTML:** Hypertext Markup Language; the standard markup language for documents designed to be displayed in a web browser.
* **Java:** a general-purpose programming language; the VLOL backend is coded in this language.
* **JavaScript:** a programming language often used to code client-side user interfaces for web pages; parts of the VLOL user interface are coded in this language.
* **JDK/JRE:** Java Development Kid/Java Runtime Environment; standard tools and libraries for development of software in the Java language.
* **Maven:** also known as Apache Maven; a build automation tool use primarily for Java projects.
* **MySQL:** an open-source relational database management system
* **Parsley:** Parsley is a JavaScript client-side validation library that evaluates user input for correctness.
* **Spring Boot:** Spring Boot is an open-source, modular Java framework used to create and configure microservices. It automatically configures applications based on dependencies managed by Maven.
* **SQL:** Structured Query Language; a domain-specific language for implementing relational databases and managing data stored in them.
* **Thymeleaf:** a Java-based, server-side template engine for web and standalone environments. It dynamically generates and populates web pages based on XHTML-based templates.
* **URL:** commonly referred to as a web address; a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. (Example: https://tools.ietf.org/html/)
* **VLOL:** Virtual Letter of Life; refers to the overall VLOL system described in this document.
* **XHTML:** Extensible Hypertext Markup Language; an implementation of HTML within the wider XML markup language.

# 2. Build

The VLOL software is primarily written in Java, with a front-end web interface written in HTML 5, JavaScript, and CSS. It makes use of the following dependencies:

* JDK/JRE 11
* Spring Boot
* Thymeleaf
* Parsley

## 2.1 Process

The recommended build process involves the following steps:

1. Download and install the dependencies listed above.
2. Pull the latest VLOL version via Git.
3. Check dependencies and build the VLOL software via Maven.
4. Following a successful build, software artifacts will be generated.
5. Archive project build logs.

## 2.2 File Structure

The files and directories critical to building and operating the VLOL software are described below and depicted in Figure 1.

* **/azure:** Settings files for deploying VLOL to an Azure-based pipeline.
* **/docker:** Settings files and scripts for deploying VLOL in a Docker container.
* **/docs:** Documentation files for the VLOL project.
* **/src/main/java/com/vlol:** Java source code for the VLOL server-side module.
* **/src/main/resources/email:** XHTML templates for generating email content.
* **/src/main/resources/static:** CSS style sheets, static images, and JavaScript modules.
* **/src/main/resources/templates:** XHTML templates for the VLOL web interface.
* **/src/test/java/com/vlol:** Unit tests for server-side code.

Figure 1. VLOL project directory structure.

├───azure

├───docker

├───docs

├───src

│ ├───main

│ │ ├───java

│ │ │ └───com

│ │ │ └───vlol

│ │ │ ├───config

│ │ │ ├───controller

│ │ │ ├───data

│ │ │ ├───model

│ │ │ ├───repository

│ │ │ └───service

│ │ └───resources

│ │ ├───email

│ │ ├───static

│ │ │ ├───css

│ │ │ ├───img

│ │ │ └───js

│ │ └───templates

│ │ ├───admin

│ │ ├───layouts

│ │ ├───menu

│ │ └───user

│ └───test

│ └───java

│ └───com

│ └───vlol

│ └───model

# 3. Deployment

The VLOL software can be deployed in a Docker container for Heroku and similar services, or via an Azure pipeline system. A generalized list of deployment steps is as follows:

* Install JDK and JRE 11
* Install Maven
* Install a relational database management system, such as MySQL or H2.
* Pull the latest stable commit from the VLOL Git repository.
* Use Maven to build the application and generate executable artifacts.
* Deploy VLOL executable artifacts and resource files to the target server.

## 3.1 Maintenance Tasks

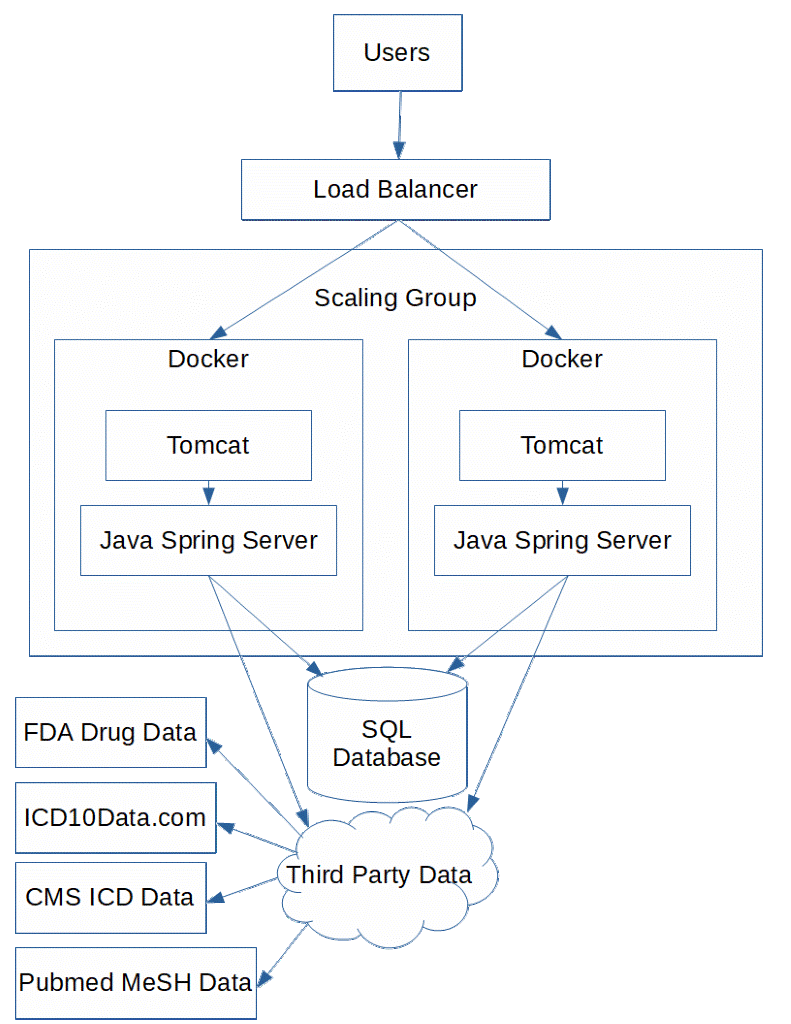
To ensure that the VLOL software remains fully functional, we recommend performing the following maintenance tasks on a regular basis:

* Daily review of log files to ensure the system is functioning as expected and to identify any problems as they occur
* Weekly audits of the database to monitor for patterns indicating malicious activity
* Installation of latest patches and fixes during scheduled maintenance windows

## 3.2 Production Deployment

For the production architecture we recommend hosting the SQL database on a separate server that the web server will connect to, rather than using an embedded database. This database server should implement encryption both at-rest an in-transit. For scalability, we also recommend placing the web servers in a scaling group, so that they can be scaled easily during high traffic events (see Figure 2.)

Figure . Recommended architecture for hosting VLOL.



## 3.3 HIPAA Compliance

To achieve HIPAA compliance for production deployments in which real data will be stored on the system, the system architecture must meet these additional requirements and documentation:

* The database should be encrypted.
* The database should be backed up.
* Application server(s) should have some sort of malware protection.
* HTTPS should be forced for user connections.
* After integration with Health Information Exchanges, assuming health data is pulled down from the exchange, the system must implement protocols to verify users prior to displaying data.
* Establish a Disaster Recovery Plan; *i.e.,* how to restore data in the event of data loss.
* Develop an action plan for what to do in the event of a security breach (see <http://www.hipaasurvivalguide.com/hipaa-regulations/part-164-d.php>).

For more information about HIPAA compliance, read the full regulation documentation, which is available at: <http://www.hipaasurvivalguide.com/hipaa-regulations/part-164.php>

# 4. Errors

## 4.1 HTTP Status Errors

The VLOL web server may report errors by returning the following HTTP status codes:

### 4.1.1 HTTP 400 Bad Request

The server cannot or will not process the request due to an apparent client error.

Action items for resolving the error:

1. Attempt the request again in a new instance of your web browser.
2. Restart the VLOL Software (see section 5.2) and attempt the request again.
3. Escalate the issue to the maintenance development team.

### 4.1.2 HTTP 401 Unauthorized

The request contained valid data and was understood by the server, but the server is refusing action because authentication has failed or has not been provided.

Action items for resolving the error:

1. Check the username and password for correctness.
2. Log in and attempt the request again.
3. If authentication fails, use the “forgot password” functionality on the VLOL site to reset the user’s password.
4. If authentication still fails, escalate the issue to the maintenance development team.

### 4.1.3 HTTP 403 Forbidden

The request contained valid data and was understood by the server, but the server is refusing action.

Action items for resolving the error:

1. Check the requested URL for correctness.
2. If not logged in to VLOL, log in and attempt the request again.
3. If the request still fails, escalate the issue to the maintenance development team.

### 4.1.4 HTTP 404 Not Found

The requested resource could not be found.

Action items for resolving the error:

1. Check the requested URL for correctness and attempt the request again.
2. If the URL is correct and VLOL still issues a 404 error in response, then escalate the issue to the maintenance development team.

### 4.1.5 HTTP 500 Internal Server Error

The server encountered an unspecified problem that prevented it from fulfilling the request.

Action items for resolving the error:

1. Wait five minutes and attempt the request again.
2. Restart the VLOL Software (see section 5.2) and attempt the request again.
3. If the error persists, escalate the issue to the maintenance development team.

### 4.1.6 HTTP 502 Bad Gateway

The server was acting as a gateway or proxy and received an invalid response from the upstream server.

Action items for resolving the error:

1. Wait five minutes and attempt the request again.
2. Restart the VLOL Software (see section 5.2) and attempt the request again.
3. If the error persists, escalate the issue to the maintenance development team.

### 4.1.7 HTTP 503 Service Unavailable

The server cannot handle the request, because it is overloaded or down for maintenance. This is generally a temporary state.

Action items for resolving the error:

1. Wait five minutes and attempt the request again.
2. Restart the VLOL Software (see section 5.2) and attempt the request again.
3. If the error persists, escalate the issue to the maintenance development team.

## 4.2 Java Errors and Exceptions

Java defines two categories of errors: Errors and Exceptions. Java Errors are typically non-recoverable, while Exceptions represent situations from which the software can and should recover. In either case these errors should always be escalated to the maintenance development team, as developer involvement is usually necessary to resolve them, and they most often indicate problems in the code itself.

Java’s OutOfMemoryError is an unusual exception to this rule. This error may be caused by other software using excessive system resources, or a problem with the server hardware itself, and so some administrative actions may alleviate the issue.

## 4.3 Known Issues

As of October 30, 2020, there are no known bugs or issues present in the VLOL software.

# 5. Software Operation Instructions

## 5.1 Installation

The VLOL software will run on any Linux or Windows-based operating system, real or virtual, provided the following requirements are met:

* x86-64 compatible processor
* 2 GB RAM for Windows; 1 GB for Linux
* JRE 11
* JDK 11
* Maven 3.6
* MySQL 8

When installing the software, ensure that the JAVA\_HOME, JDK\_HOME, JRE\_HOME, and MAVEN\_HOME environment variables contain the correct path to the JDK and Maven application directories. For example, in Windows:

* JAVA\_HOME: C:\Program Files\Java\jdk-11.x.x
* JDK\_HOME: %JAVA\_HOME%
* JRE\_HOME: %JAVA\_HOME%
* MAVEN\_HOME: C:\Program Files\apache-maven-3.6.x\bin

Afterwards, clone the VLOL git repository to local storage and build the project via Maven.

## 5.2 Start, Stop, and Restart Procedures

**Starting VLOL**

First, obtain the designated IP address and server port from the system administrator (*e.g.,* localhost:5000, etc.)

To start the application, open the application.properties file in /src/main/resources and ensure that server.port is set to the server port obtained from the administrator.

Return to the root of the VLOL Git repository and execute the following commands:

* mvn compile
* mvn spring-boot:run

Verify that the application is running by either entering the application IP address and server port in a web browser, or checking for an HTTP 200 OK response using a Client URL (cURL) command.

**Stopping VLOL**

To properly shutdown the application, issue a cURL POST command with administrator privileges to activate the shutdown endpoint:

* curl -u admin@vlol.gov:P@ssW0rd -v -X POST http://localhost:5000/actuator/shutdown

In the event of an emergency, look up the application’s process ID number (PID) and shutdown using the taskkill command:

* netstat -ano | findstr <port>
* taskkill /PID <PID>

If necessary, add the force flag (/F) to the taskkill command:

* taskkill /F PID <PID>

**Restarting VLOL after an improper shutdown**

If the application crashes or is shutdown forcefully, check the embedded Tomcat server log to determine the cause. These files are currently located in /tomcat/logs. After correcting any problems, restart the application using the steps described in the “Starting VLOL” section.

## 5.3 Security Breach Response

In the event of a security breach in which sensitive data may have been lost or otherwise compromised, we recommend several steps. All of the following steps should occur no latter than 60 days following discovery of the breach:

* All affected and potentially affected individuals should be notified of the breach. These notifications should be written in plain language.
* If data belonging to greater than 500 residents of a given state or jurisdiction was affected by the breach, prominent media outlets serving the area should be notified.
* The Secretary of Health and Human Services must be notified.

# 6. Frequently Asked Questions

1. Is an account necessary to obtain information about the VLOL system?  
   *No. Anyone who wants to read about VLOL can do so by accessing the about page in the VLOL web interface.*
2. Can users give other users access to view their account?  
   *Yes. Users should be advised to be careful when choosing who to provide access to their account, as it contains personally identifiable information.*
3. How does VLOL secure users’ information?  
   *The VLOL software is designed according to NIST security standards. Roles and access are assigned to user accounts based on least privilege and need-to-know privileges. The software uses two factor authentication and uses the HTTPS protocol.*
4. Can users elect not to include their Social Security Number (SSN) during account creation?  
   *No. VLOL uses the SSN to confirm the user’s identity.*
5. Does the VLOL software adhere to HIPAA requirements?  
   *Yes.*
6. Can EMS personnel access a person’s Letter of Life information if that person has not created an account on VLOL?  
   *No. A VLOL account is required for Letter of Life information to be available to EMS teams.*
7. Is there a way to ask for support?  
   *The current development team will provide the subsequent team with a Programmer’s Guide, which will cover operations, maintenance, and retirement. The point of contact for the current team is the Software Engineering (SWEN) department of the University of Maryland Global Campus.*

# License

This software is free to use by anyone. It comes with no warranties and is provided solely "AS-IS". It may contain significant bugs, or may not even perform the intended tasks, or fail to be fit for any purpose. University of Maryland is not responsible for any shortcomings and the user is solely responsible for the use.

The VLOL application code is currently licensed under the Massachusetts Institute of Technology (MIT) License. This is a permissive license; the code may be reused without requiring the copyright holders’ permission or the payment of royalties, but the copyright holders are not liable for any claim, damages, or other issues arising from its use. However, the developers reserve the right to change the license in the future.

The full text of the MIT License can be found at <https://opensource.org/licenses/MIT>.

# APPENDIX A: Work from Previous Teams

## I. Introduction

### I. A. Project History

The Virtual Letter of Life system began in May of 2020 as a class project designed and developed by a team of students (the Summer 2020 team) attending the graduate-level Software Engineering capstone course at the University of Maryland Global Campus. Following the conclusion of the Summer 2020 semester, the Summer 2020 team turned over control of the project and all project materials to a new team of students, the Fall 2020 team.

For the sake of maintaining a historical record of the Summer 2020 team’s work, this appendix contains any material from the Summer 2020 Runbook that has not been duplicated or adapted elsewhere in this document. This material is presented as-is, with minimal edits aside from modifications to fit the content to this document’s format or to provide context.

The VLOL Summer 2020 Team consists of seven members:

* Mohammed Allibalogun
* Charles Baisie
* Jamal Bourne
* Meron Getachew Debela
* Robert Garcia
* Augustin Mwamba
* Sefanit Urgessa

### I. B. Previous Team Revision History

The revision history from the previous team’s runbook is presented below.

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 07/03/2020 | 1.0 | Initial Project Plan Release |

## II. Disaster Recovery/Service Level Agreement

Disaster Recovery strategies encompasses business continuity factors such as recovery point objective (RPO) and recovery time objective (RTO). The developers of the VLOL utilized the free tier versions of tools and services for development, therefore, does not require any agreement of assurance or commitment from the service providers. This application is developed as a prototype-level application or free tier level application that can be adopted by any Fire Department/EMT.

Production-level infrastructure and access controls are out-of-scope for this development. Any deployed version of the application is maintained in the pre-built umgc Git repository provided by the summer 2020 DevOps team. There is no assurance for responsibility, availability, accessibility or scalability of the application beyond those offered by the respective vendors.