Open Source Engine Integration Developer's Guide

Harris SmartWorks
Open Source Engine Integration
Computer Science Dept
COSC470
Okanagan College

Aubrey Nickerson
Bao Mai
Derek Manchee
Joseph Egely
Keaton Canuel

April 29, 2021

Revision Sheet			
Revision	Date	Summary of Changes	Name
Version 1.1	04-22-2021	Baseline Draft Document	Derek Manchee
Version 1.2	04-22-2021	Added AES Encryption Standards and SQL Triggers	Aubrey Nickerson

Contents

1 Introduction

The Open-Source Integration Engine Developer's Guide is a guide book that shows the standards/conventions recommended for coding and building the software.

2 Software Management

The project team used and recommends using three tools for team coordination and management while working on the software. JIRA, Bitbucket and confluence can all be managed through an Atlassian account and are integrated together for easy access and referencing.

- JIRA – This software is used for planning and tracking the progress of the software in sprint segments. With this tool you can see what has been done, what needs to be done and add new tasks or issues that need to be solved. You can also see the time that is taken to do each task, who did the task and what tasks are currently being worked on. After a sprint a print out in graph format can be done to show the progress.
- Confluence- This software is used to show meeting notes and other text or image submittables in a centralized location for easy reference and sharing of knowledge. Can seamlessly reference sprints and sprint tasks.
- Bitbucket – A git repository management system. Allows multiple workflows for each member in what are called branches. The branches can be merged in pull requests which allow other members to check code before merge takes place. Limits access to sensitive code and allows tracking of what has been done as well as version control.

3 Document Guidelines

3.1 Accessibility

Documents pertaining to the Open-Source Integration Engine will be available at the discretion of Harris SmartWorks.

3.2 File Format

Documents are written in LaTex using Overleaf since it is a commonly used writing system and has many export formats. Having the documents on overleaf makes them easily accessible to all members of the team.

4 Target Architecture

It is a priority to write flexible code and utilize data structures that have the likeliest acceptable performance on a given architecture. For excellent performance, this can mean reducing redundancy, short functions, avoid long functions, and so on.

5 Coding Conventions

5.1 JavaScript AES Encryption Standards

To import the AES encryption library for JS, it just needs to be added as a script tag on the top of each html file in the head tag. Also add the additional class called JSAES.js. For example:

```
<head>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.2/rollups/aes.js">
 </script>
 <script type="text/javascript" src="js/JsAES.js"></script>
</head>
```

The 2 passkeys that encrypts the data is declared at the top of each JS file and it looks like this:

```
let key = "ddfbccae-b4c4-11";
let iv = "ddfbccae-b4c4-11";
```

To encrypt data in JavaScript it looks like this:

```
aes_encrypt(variableName, key, iv);
```

To decrypt data, it looks like this:

```
aes_decrypt(variableName, key, iv);
```

5.2 Python AES Encryption Standards

To import the AES encryption, just add 'import' and the PyAES.py file at the top of the app.py file like shown below:

```
from PyAES import AesCrypto
```

And declare the aes object like:

```
aes = AesCrypto('ddfbccae-b4c4-11')
```

The PyAES.py file handles the encryption.

To encrypt the data in Python on the app.py, it looks like this:

```
aes.encrypt(variableName)
```

To decrypt the data in Python, it looks like this:

```
aes.decrypt(varibaleName)
```

When decrypting data coming from JavaScript, to Python. There are extra spaces that are added at the end of the data on the Python side for some odd reason. To get around this, the team had to add a function that gets rid of the extra space as shown below:

```
''.join(e for e in aes.decrypt(variableName) if e.isalnum())
```

5.3 Python Coding Standards

5.3.1 Naming Conventions

Variable names and functions use camel case (myVariable, myFunction(x, y))

5.4 Line Lengths and Breaks

Line should be no longer than 40 characters long before going into a new line.

5.5 Indentation

It is important to add indentation in Python otherwise errors will occur. 4 space indentations are required after if statements, while loops and functions.

5.6 Brackets and Braces

Brackets and braces are not common in python because it enforces indentation. Python only uses square brackets when declaring arrays or objects.

5.7 SQL Naming Conventions

SQL names use underscore separation (my_variable).

5.8 SQL triggers

There are only two triggers in the database. PATIENTVISITS_TRIGGER and PATIENTVISITS_DATE_TRIGGER

```
BEGIN
  DECLARE vUserID INT;
  SELECT USER_ID INTO vUserID
  FROM ROLES
  WHERE USER_ID = NEW.PATIENT_ID;
  IF vUserID IS NOT NULL THEN
    UPDATE ROLES
    SET IS_ACTIVE = 1;
  END IF;
END
```

The trigger above is the PATIENTVISITS_TRIGGER. When a new row is inserted into the PATIENTVISITS table, It checks the user id in the ROLES table to see if the newly inserted patient exists in the ROLES table. If the patient exists then the IS_ACTIVE column in the ROLES table is set to 1, meaning the patient is active.

```
BEGIN
  IF NEW.DISCHARGEDATETIME IS NOT NULL THEN
    UPDATE ROLES
    SET IS_ACTIVE = 0
    WHERE USER_ID = NEW.PATIENT_ID;
  END IF;
END
```

The trigger above is the PATIENTVISITS_DATE_TRIGGER. When a new row is inserted into the PATIENTVISITS table, it checks if the new DISCHARGEDATETIME is not empty. If it isn't, then the patient associated with the visit in the ROLES table is set to non active.

5.9 Comments

All comments within the code refers to the section under the comment.

6 Shared and External Libraries

6.1 JavaScript Libraries

- **j**Query
- **C**rypto-JS (AES Encryption library)
- **A**nyChart (Interactive Charts library)
- **m**oment (Date library)

6.2 Python Libraries

- **h**l7 (Library for parsing HL7 messages)
- **m**ysql-connector-python (MySQL Client library)
- **d**atetime (Date and Time library)
- **f**lask (Python Web Framework)
- **f**lask_cors (Cross Origin Resource Sharing library)
- **f**lask_socketio (bi-directioanal communications library)
- **b**inascii (Convert between binary and various ASCII- encoded binary representations)
- **c**ryptography (AES Encryption library)
- **p**ycrypto (AES Encryption library)
- **f**lask_alchemy (For Unit and Regression testing)
- **u**nittest (Unit Testing library)