**Harley Test Automation Management System**

Table of Contents

[Introduction 3](#_Toc486686642)

[Section 1 - Module Description 4](#_Toc486686643)

[Database 4](#_Toc486686644)

[Initializing the database 5](#_Toc486686645)

[Updating the database 5](#_Toc486686646)

[Configurations 6](#_Toc486686647)

[Editing settings configuration 6](#_Toc486686648)

[Generating Report 6](#_Toc486686649)

[Input parameters 6](#_Toc486686650)

[Output 6](#_Toc486686651)

[Run 7](#_Toc486686652)

[Section 2 - Recipes 8](#_Toc486686653)

[Edit Command 8](#_Toc486686654)

[Recipe 1: To edit Harley development machine name 8](#_Toc486686655)

[Recipe 2: To toggle database update 8](#_Toc486686656)

[Recipe 3: To toggle pagination for HTML report 8](#_Toc486686657)

[Recipe 4: To edit Harley user name 8](#_Toc486686658)

[Recipe 5: To edit Harley development machine 8](#_Toc486686659)

[Recipe 6: To edit output file name 8](#_Toc486686660)

[Recipe 7: To edit output file format 8](#_Toc486686661)

[Generate Command 8](#_Toc486686662)

[Recipe 8: To generate protocol status for a release and for a test type 8](#_Toc486686663)

[Recipe 9: To generate protocol status for a release and for all test type 8](#_Toc486686664)

[Run Command 8](#_Toc486686665)

[Recipe 10: To run the job schedule 8](#_Toc486686666)

[Recipe 12: To input the job schedule for the runner 8](#_Toc486686667)

Table of Figures

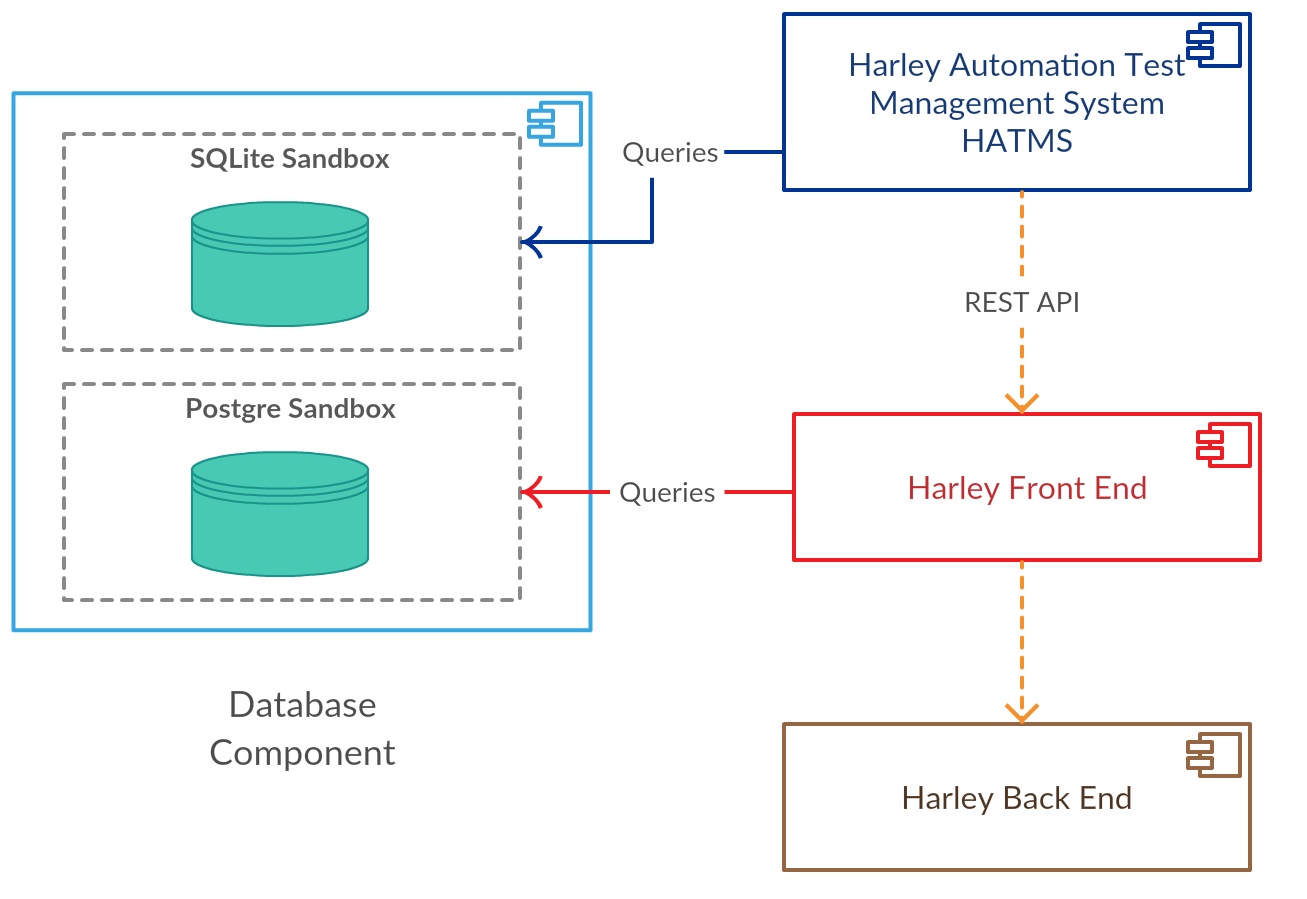
[Figure 1 Overview of HATMS 4](#_Toc486686670)

[Figure 2 Class Diagram 5](#_Toc486686671)

[Figure 3 Database schema 6](#_Toc486686672)

# Introduction

Harley Test Automation Management System (henceforth be called as HATMS) is a framework used to generate reports and run batch jobs using the Harley front end. The overview of the Automation Management System with Harley is shown in the figure below.



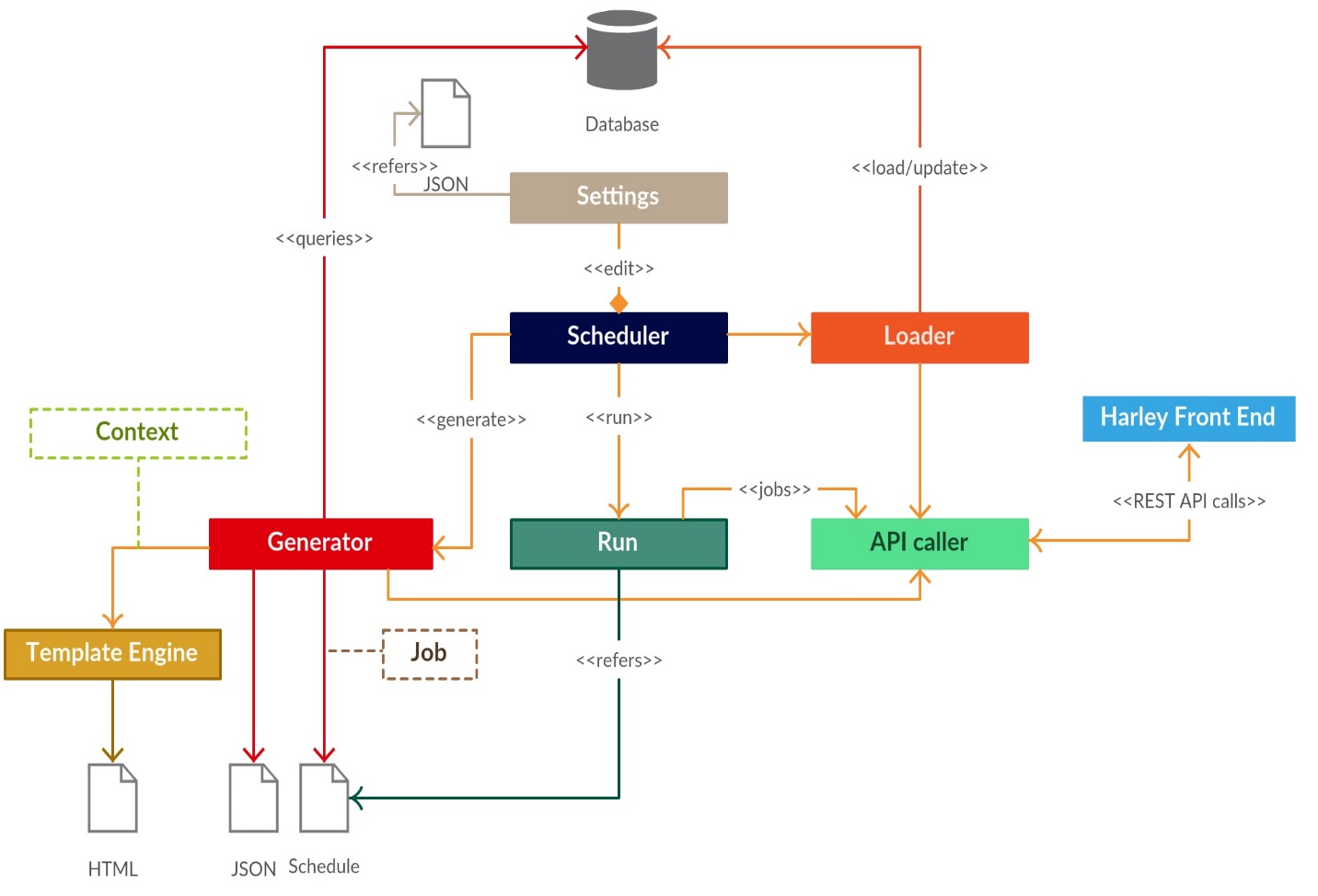
**Figure 1 Overview of HATMS**

The goal of HATMS is to ensure that each production release of the ghost test framework is working for all the supported protocols. By using HATMS developers will be able to idle specific cases where the release failed and their performance compared to their previous release.

The document has two sections. First section covers HATMS features and the functionality of each module. The second section contains recipe’s showing you on how to use HATMS through the command line interface.

# Section 1 - Module Description

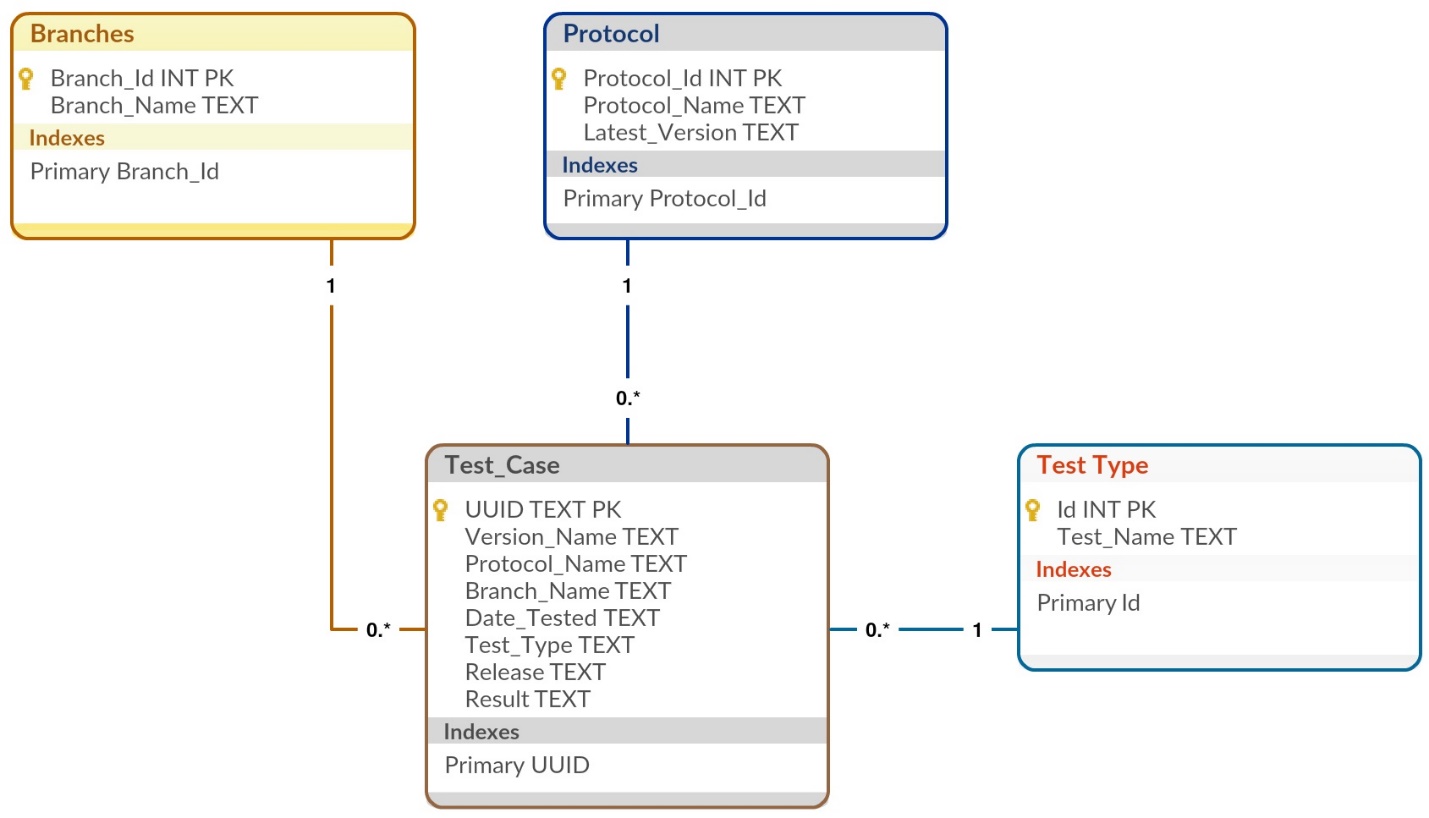
The class diagram of HATMS is shown below. HATMS has four key modules which performs the required functionalities.



**Figure 2 Class Diagram**

## **Database**

HATMS gets its data from Harley front end which exposes REST API calls to fetch the data corresponding to ghost test framework. When HATMS starts, it checks for the availability of the database. If it does not find one, it starts to initialize the database with contents from the ghost test framework. If it finds a database, it checks for any updates that needs to be synchronized and performs the update.



**Figure 3 Database schema**

### **Initializing the database**

During this stage the database loader module uses SQL script file to create the database and the schema. The database schema is shown above. It starts populating the tables by making REST API calls to Harley front end.

**Note:**

There are no restrictions in the way the SQL script are written if they are valid. The database loader uses regular expression (regex) to parse the SQL statement. It is robust and guarantees creation of database schema

### **Updating the database**

When HATMS sees existing database, it starts to update it with values from Harley front end. It deletes all the records from tables – ***Branches, Protocol and Test Type*** and populates them again. The number of records for these tables are few and they get populated almost instantly.

Records for the ***Test Case*** table are updated based on the recent timestamp between the local and remote database. The database loader starts iterating through the data from the remote database and loads them in the local copy until it finds the matching time stamp between the two.

Both these operations happen before the scheduler wakes up to perform other operation which ensures that correct results are always generated.

## **Configurations**

HATMS depends on configuration files to perform it tasks. These configuration files are present in the ‘config’ folder. The configuration files are in Java Script Object Notation (JSON) format. No other formats are supported. The following are the configuration files that will be used by HATMS

1. *Settings* – Used by the generator
2. *New job template* – Used by runner
3. *Schedule job* – Used by runner
4. *Log* – Used for configuring the logger

The description and usage of the configuration files will be discussed in their appropriate module description. The figures below show you the parameters present in each of the configuration file

Figures of 1, 2, 3 go here

## **Editing settings configuration**

The settings configuration will be used by the generator and the database loader modules. The information about the Harley machine and database will go in this file. The key, value pair of the settings file is self-explanatory. This configuration can be edited using the command line interface or the file directly. This is the only file which can be edited using the command line interface. You can find the recipes for editing the settings in the next section

## **Generating Report**

HATMS supports generating two types of reports

1. Protocol Status for all the protocols for a release and a test type
2. Protocol Status for all the protocols for a release and for all test types

### **Input parameters**

The following are the input parameters to the generator.

***Branch*:** Name of the branch. Currently the reports can be generated for a specific branch. If the user wants to generate reports for all the branches. He / She can generate report for each branch individually

***Protocol*:** Name of a protocol or ‘all’ the protocols supported by a branch.

***Release:*** The ghost test framework release

***Test type:*** Individual test type or for all the test types.

***Version:*** Currently specifying a protocol version is not supported. The type of report generated will have the latest version of the protocol.

### **Output**

The generator creates a report and a job schedule.

***Report:*** It supports two formats for the generator – HTML and JSON. User has the option to choose between the two using the command line interface.

***Job schedule:*** The generator will track the protocols which failed or not tested for the current release. It creates a configuration file containing the protocols which failed or not tested. This configuration file serves as an input to the runner. Currently the configuration file is created for a specific test type.

## **Run**

The runner module is responsible to generating batch jobs. It uses REST API calls to interact with the Harley front end to start the jobs. The input for the runner is from the job schedule configuration file.

The job schedule configuration file holds all the required parameters to populate new jobs. The parameters in the job schedule are self-explanatory. However, the following two parameters need some additional information.

|  |  |
| --- | --- |
| ***Run:*** | The run parameter can handle three type of values – ‘all’, list of protocol names and dictionary of protocol name and version. These three options are mutually exclusive. You have any only one at a time |
| ***Exclude:*** | The exclude parameter supports only one type of format – list of protocol names. Even if you want to exclude only one protocol. The protocol listed in the exclude parameter will be skipped during the batch run. |

There is an optional argument where the user can specify job schedule other than the default one. If the user has supplied the job schedule using the optional argument, the default job schedule will be skipped.

# **Section 2 - Recipes**

## **Edit Command**

### ***Recipe 1: To edit Harley development machine name***

python hatms.py edit --harley harley-dev1

### ***Recipe 2: To toggle database update***

* ***set*** - python hatms.py edit --db\_update True
* ***reset*** - python hatms.py edit --db\_update False

### ***Recipe 3: To toggle pagination for HTML report***

* ***set*** - python hatms.py edit --pagination True
* ***reset*** - python hatms.py edit --pagination False

### ***Recipe 4: To edit Harley user name***

python hatms.py edit --user admin

### ***Recipe 5: To edit output file name***

* python hatms.py edit -o status\_release (*or)*
* python hatms.py edit --output status\_release

### ***Recipe 6: To edit output file format***

|  |  |
| --- | --- |
| ***html*** - | * python hatms.py edit -f html (*or)* * python hatms.py edit --format html |
| ***json*** - | * python hatms.py edit -f json *(or)* * python hatms.py edit --format json |

## **Generate Command**

### ***Recipe 8: To generate protocol status for a release and for a test type***

* python hatms.py generate -b master -p all -t smoke -r 1.4.0 *(or)*
* python hatms.py generate --branch master --protocol all --type smoke --release 1.4.0

### ***Recipe 9: To generate protocol status for a release and for all test type***

* python hatms.py generate -b master -p all -r 1.4.0 *(or)*
* python hatms.py generate --branch master --protocol all --release 1.4.0

## **Run Command**

### ***Recipe 10: To run the job schedule***

python hatms.py run

### ***Recipe 12: To input the job schedule for the runner***

* python hatms.py run -i /path/to/new\_schedule.json *(or)*
* python hatms.py run --infile /path/to/new\_schedule.json