ΕΜΕΊΣ

CMXM

CodusMaximus

HyperPerform
User Manual

Client: Schalk Lotz, Magna BC

Developers:
Rohan Chhipa *14188377*
Claudio Da Silva *14205892*
Jason Gordon *14405025*
Avinash Singh *14043778*

Updated October 12, 2016

Contents

1 System Overview **3**

2 System Configuration **3**

2.1 Docker installation . . . 3

2.2 Manual Installation . . . 3

2.3 Event Gathering . . . 4

2.4 Miscellaneous . . . 5

3 Installation **5**

3.1 Docker Installation . . . 5

3.2 Manual Installation . . . 5

3.2.1 WildFly . . . 5

3.2.2 PostgreSQL . . . 6

3.2.3 ActiveMQ . . . 9

3.3 Notifications . . . 10

3.3.1 Deploying to WildFly . . . 12

3.3.2 Front-end Dashboard . . . 12

4 Getting Started/Using the System **14**

1 System Overview

Many different tools are available for measuring the quality of products made, but very few tools exist which assess the quality of the people making said products. People play a huge role in a project, and trying to monitor each and every one becomes a tedious task which diverts man power away from other more critical tasks. Whether it be for an end of year evaluation, or attempting to assess the current status of a project, generating a report on a staff member can help keep up productivity, as well as get them any help they need in order to resume quality performance. By ensuring that there is constant quality performance from each individual on a project, one can increase project quality as well as reduce project risks such as loss of an important team member during a critical stage of a project's life-cycle.

The HyperPerform system has the ability to automatically gather information from multiple integrations such as GitHub and Travis. However due to the highly pluggable nature of the system one could easily add more integrations to pull information from. The HyperPerform system simplifies management by allowing a manager to view all relevant employee data in simple and easy to understand reports, where reports can be a summarised report or a more detailed report on employee activity.

2 System Configuration

This guide has been made for users using a Linux based operating system. To install the HyperPerform system on the machine you will be required to have an active connection to the internet. Please note that high amounts of data might be consumed.

2.1 Docker installation

To install system with ease and avoid all configurations you can download Docker. Docker can be found at `www.docker.com` where guides are made available for installing docker on a particular operating system. If you intend to use Docker to install the HyperPerform system then please ensure you install Docker on your machine.

2.2 Manual Installation

The manual installation requires you to download the source code from the GitHub repository. The newest release is highly recommended. To carry out a manual installation please ensure you have Maven and the WildFly application server on your machine.

Maven can be downloaded from: `maven.apache.org`

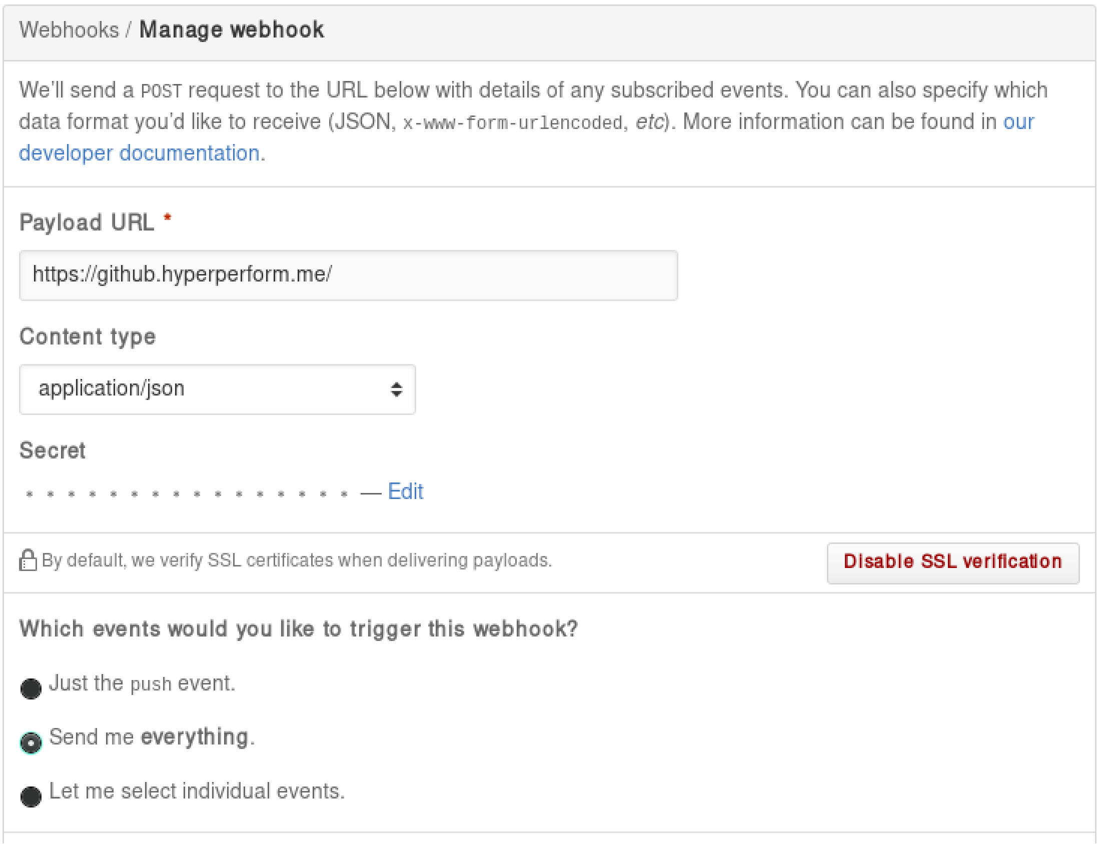
WildFly can be downloaded from: `wildfly.org`

Please ensure you download WildFly 10. The HyperPerform system was fully tested on this version of WildFly. Any other version might produce unexpected behaviour.

For the front-end Dashboard please ensure you have Nodejs (version 6.4.0 or higher) installed on your machine. Nodejs can be found at `https://nodejs.org/en/`.

2.3 Event Gathering

The system gathers information through webhook technology. Thus to be able to receive any events the computer on which the system will be installed **must** be connected to the internet. When configuring the integrations you will need to provide the URL for that integration to send events to. The following figure shows the GitHub webhook.



Webhooks / **Manage webhook**

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://github.hyperperform.me/

Content type

application/json

Secret

* * * * * * * * * * * * * * * * * — Edit

By default, we verify SSL certificates when delivering payloads.

Which events would you like to trigger this webhook?

Just the push event.

[✓ Send me **everything**.]

Let me select individual events.

Figure 1: Adding GitHub webhook

An optional feature would be to bind a domain name to the global IP address of the server. However this is merely for readability purposes and work affect system performance in any way.

2.4 Miscellaneous

Certain components of the system require user names which are consistent. The Git event processor is one such component. When using GitHub on your local machine please ensure that your local configurable Git name corresponds to your actual GitHub username.

To check your local Git name open terminal or command prompt and run the following command:

```
git config user.name
```

If the name corresponds to your account name then nothing further needs to be done and you can continue to the installation. However if the two names do not match then run the following command:

```
git config --global user.name "<username>"
```

In the command above the <username> is your actual account name.

3 Installation

3.1 Docker Installation

Assuming you have docker installed on your machine, simply run the following command in terminal:

```
docker run hyperperform/HyperPerform
```

This will download the HyperPerform Docker image from DockerHub and run it on you machine.

The front end component does not have a Docker image at this point in time. To install the front end component please refer to section 3.2.5 for the manual installation.

3.2 Manual Installation

This installation guide assumes a Linux Server running Ubuntu 14 or higher:

3.2.1 WildFly

Once you have downloaded the WildFly application server please carry out the WildFly installations and add a user. Once this is done proceed to installing PostgreSQL.

3.2.2 PostgreSQL

The install PostgreSQL on your machine:

Install via terminal:

```
sudo apt-get update
sudo apt-get install postgresql postgresql-contrib
```

To configure PostgreSQL to connect remotely:

```
sudo nano /etc/postgresql/9.3/main/postgresql.conf
```

Edit the following lines:

```
listen_addresses = "*"
```

Create database hyperperform and the tables

Run the following commands in terminal:

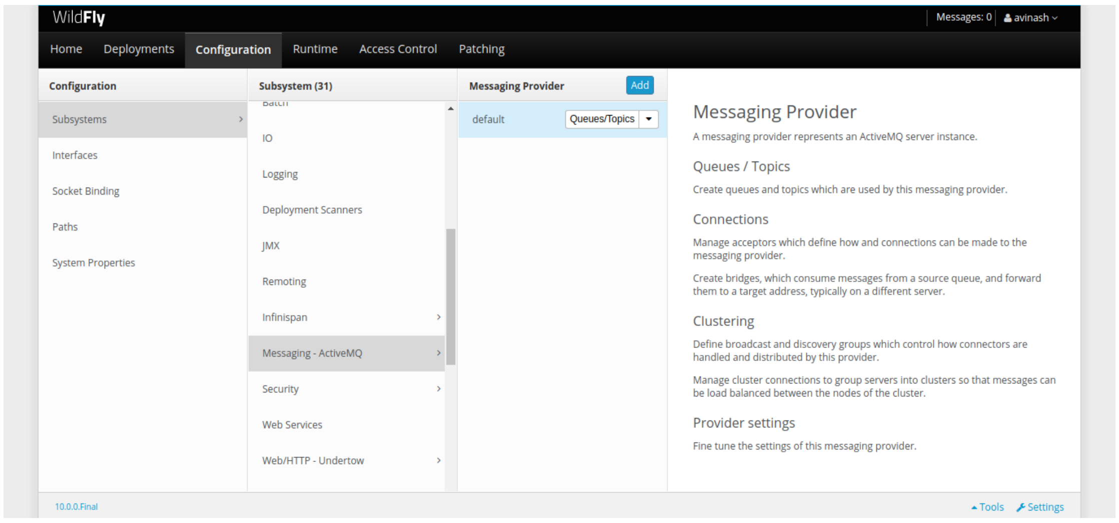
```
psql -c 'CREATE DATABASE hyperperform;' -U postgres

psql -d hyperperform -c 'CREATE TABLE public."GitPush" ( id
   integer NOT NULL, repository character varying(255), "
   timestamp" timestamp without time zone, username character
   varying(255), commitsize integer, url character varying(255)
   , message character varying(255), CONSTRAINT "GitPush_pkey"
   PRIMARY KEY (id) ); CREATE SEQUENCE public.
   hibernate_sequence INCREMENT 1 MINVALUE 1 MAXVALUE
   9223372036854775807 START 1 CACHE 1;' -U postgres

psql -d hyperperform -c 'CREATE TABLE public."TravisEvent" ( id
    integer NOT NULL, branch character varying(255), commiter
   character varying(255), repo character varying(255), status
   character varying(255), "timestamp" timestamp without time
   zone, CONSTRAINT "TravisEvent_pkey" PRIMARY KEY (id));' -U
   postgres

psql -d hyperperform -c 'CREATE TABLE public."CalendarProject"
   ( projectid integer NOT NULL, calendarid character varying
   (255), collaborators bytea, creator character varying(255),
   duedate timestamp without time zone, eventid character
   varying(255), reponame character varying(255), "timestamp"
   timestamp without time zone, CONSTRAINT "
   CalendarProject_pkey" PRIMARY KEY (projectid));' -U postgres

psql -d hyperperform -c 'CREATE TABLE public."CalendarMeeting"
   ( meetingid integer NOT NULL, calendarid character varying
   (255), creator character varying(255), duedate timestamp
   without time zone, eventid character varying(255), location
   character varying(255), "timestamp" timestamp without time
   zone, CONSTRAINT "CalendarMeeting_pkey" PRIMARY KEY (
   meetingid));' -U postgres
```

```
psql −d hyperperform −c 'CREATE TABLE public."
    CalendarMeeting_attendees" ( "CalendarMeeting_meetingID"
    integer NOT NULL, attendees integer , attendees_key character
     varying(255) NOT NULL, CONSTRAINT "
    CalendarMeeting_attendees_pkey" PRIMARY KEY ("
    CalendarMeeting_meetingID", attendees_key), CONSTRAINT
    fkn4q1pmj9vx3tfsaw9irp9voax FOREIGN KEY ("
    CalendarMeeting_meetingID") REFERENCES public."
    CalendarMeeting" (meetingid) MATCH SIMPLE ON UPDATE NO
    ACTION ON DELETE NO ACTION);' −U postgres

psql −d hyperperform −c 'CREATE TABLE public."GitIssue"(id
    integer NOT NULL, action character varying(255), assignee
    character varying(255), createdby character varying(255),
    issueid bigint , repository character varying(255), "
    timestamp" timestamp without time zone, title character
    varying(255), url character varying(255), CONSTRAINT "
    GitIssue_pkey" PRIMARY KEY (id));' −U postgres

psql −d hyperperform −c 'CREATE TABLE public."User" (email
    character varying(255) NOT NULL, gitusername character
    varying(255), name character varying(255), "position"
    integer , profilepicture bytea,  role integer , surname
    character varying(255), username character varying(255),
    password character varying(255), CONSTRAINT "User_pkey"
    PRIMARY KEY (email));' −U postgres

psql −d hyperperform −c 'CREATE TABLE "AccessEvent"(id integer
    NOT NULL, email character varying(255), day bigint , deviceid
     character varying(255), employeeid character varying(255),
    name character varying(255), surname character varying(255),
     "timestamp" timestamp without time zone, CONSTRAINT "
    AccessEvent_pkey" PRIMARY KEY (id) );' −U postgres

psql −d hyperperform −c 'CREATE TABLE public."ForecastData"(
    data character varying(10485760) NOT NULL, CONSTRAINT "
    ForecastData_pkey" PRIMARY KEY (data));' −U postgres
```

3.2.3 ActiveMQ

To setup ActiveMQ on your server:

- Start up your WildFly application server
- Navigate to WildFly management console on localhost:9990
- Navigate to configurations tab and click on sub-systems
- Scroll down and search for Messaging-ActiveMQ and click on it
- Click on default, select queues/topics



Messaging Provider

A messaging provider represents an ActiveMQ server instance.

Queues / Topics

Create queues and topics which are used by this messaging provider.

Connections

Manage acceptors which define how and connections can be made to the messaging provider.

Create bridges, which consume messages from a source queue, and forward them to a target address, typically on a different server.

Clustering

Define broadcast and discovery groups which control how connectors are handled and distributed by this provider.

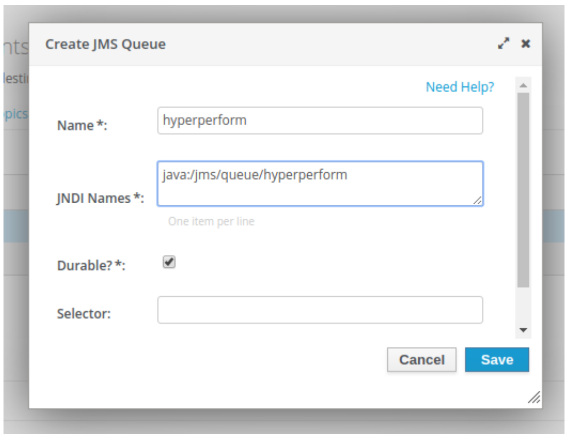
Manage cluster connections to group servers into clusters so that messages can be load balanced between the nodes of the cluster.

Provider settings

Fine tune the settings of this messaging provider.

Figure 2: Subsystems Configuration

- Click add and input the following information:
  - Name*: hyperperform
  - JNDI Names*: java:/jms/queue/hyperperform


Create JMS Queue

Need Help?

Name *: hyperperform

JNDI Names *: java:/jms/queue/hyperperform

One item per line

Durable? *: ✓

Selector:

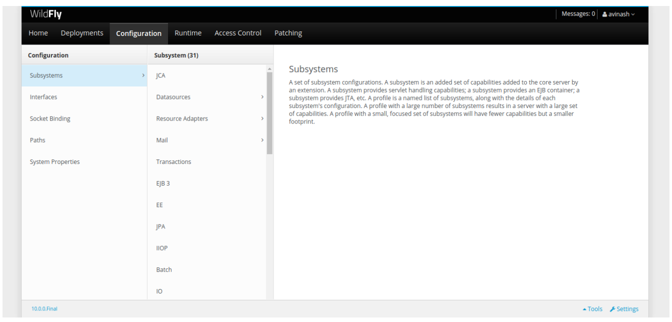
Figure 3: Queue Configuration

- Click save

3.3 Notifications

Please Note: These settings are for a Gmail configuration if you are running your own stmp server you need to adjust the information appropriately. To setup Notifications via Email on the server:

- Start up your WildFly application server
- Navigate to WildFly management console on localhost:9990
- Navigate to configurations tab and click on sub-systems
- Scroll down and search for Email and click on it



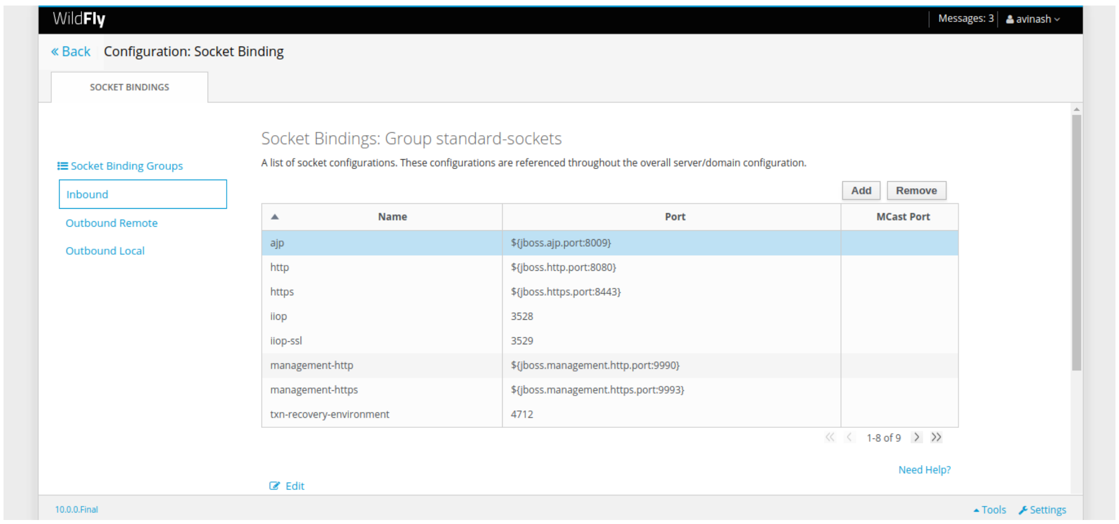
Subsystems

A set of subsystem configurations. A subsystem is an added set of capabilities added to the core server by an extension. A subsystem provides servlet handling capabilities; a subsystem provides an EJB container; a subsystem provides JTA, etc. A profile is a named list of subsystems, along with the details of each subsystem's configuration. A profile with a large number of subsystems results in a server with a large set of capabilities. A profile with a small, focused set of subsystems will have fewer capabilities but a smaller footprint.

Figure 4: Subsystem Email

- Click add and input the following information:
  - Name*: Gmail
  - JNDI: java:/jboss/mail/gmail
- Click save

- There after View the new configuration and Add a new type
  - Socket-binding: mail-smtp
  - Type: smtp
  - Username: "Your@gmail account"
  - Password: "Gmail password"
  - SSL: enable
- There after reload the server
- After reloading navigate to configurations tab and the Socket Bindings



Configuration: Socket Binding

Socket Bindings: Group standard-sockets

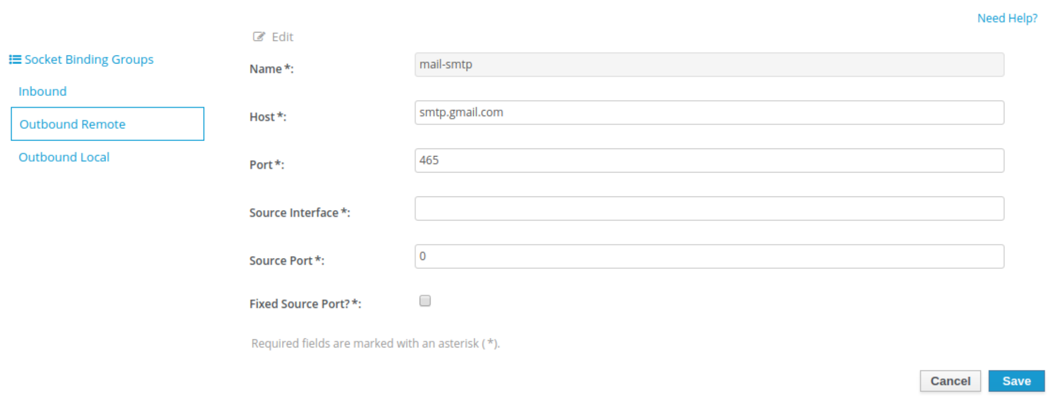
A list of socket configurations. These configurations are referenced throughout the overall server/domain configuration.

Name	Port	MCast Port
ajp	${jboss.ajp.port:8009}	
http	${jboss.http.port:8080}	
https	${jboss.https.port:8443}	
iiop	3528	
iiop-ssl	3529	
management-http	${jboss.management.http.port:9990}	
management-https	${jboss.management.https.port:9993}	
txn-recovery-environment	4712	

Figure 5: Socket Bindings

- Click on View and there after under standard-sockets click view
- Navigate to Outbound Remote and edit the mail-smtp socket bindings
  - Host: smpt.gmail.com

  - Port: 465



Name *:	mail-smtp
Host *:	smtp.gmail.com
Port *:	465
Source Interface *:	
Source Port *:	0
Fixed Source Port? *:	☐

Required fields are marked with an asterisk (*).

Figure 6: Outbound Socket Configuration

- Reload server

3.3.1 Deploying to WildFly

To deploy the HyperPerform system to the application you will need to build the system using from the source code.

- Ensure the WildFly server is running.
- Navigate to `https://github.com/HyperPerform/hyper-perform-server/releases` and download the newest release source code.
- Extract the source code
- Navigate to the root directory of the source code. A file named pom.xml should be clearly visible.
- Run the following command: mvn clean wildfly:deploy
- Maven will then ask you to provide your user name and password for the Wildfly Server.
- Thereafter Maven will automatically deploy the compiled code (war) to WildFly

3.3.2 Front-end Dashboard

Please notee that there is no release yet for the dashboard and there might be a few bugs, or limitations to the software.

To start up the front end please ensure you have Node 6.4.0 or higher installed on your machine. Node can be found at `https://nodejs.org/en/`.

Please make sure that these commands execute successfully before attempting to run the system:

```
npm install -g gulp
npm install -g bower
npm install -g sass
```

Once that has completed with no errors do the following.

- Download the Dashboard source code from `https://github.com/HyperPerform/hyper-perform-web-application`
- Navigate to the root directory of the source code

Run the following commands in terminal:

```
npm install
gulp build
gulp serve
```

The front-end system will auto launch in your default browser in order to view the data in the front-end system the Wildfly application server must be running.

4 Getting Started/Using the System

Once the front-end Dashboard is served your default browser should automatically open. In the event that it didn't, simply open the browser of your choice and navigate to the following URL: `localhost:3000`.

Once the Dashboard loads you will be presented with the following screen:

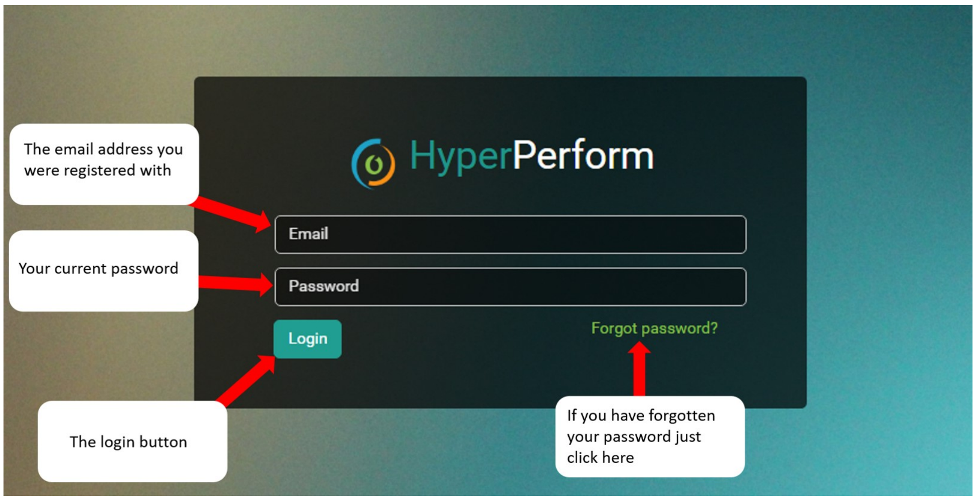

HyperPerform
The email address you were registered with
Email
Your current password
Password
Login
Forgot password?
The login button
If you have forgotten your password just click here

Figure 7: Login screen

The default username and password is Admin. This is a default login for when the system is installed for the first time. Once managers exist within the database this feature will be disabled for security purposes.

Once logged in the user will be presented with the following screen:

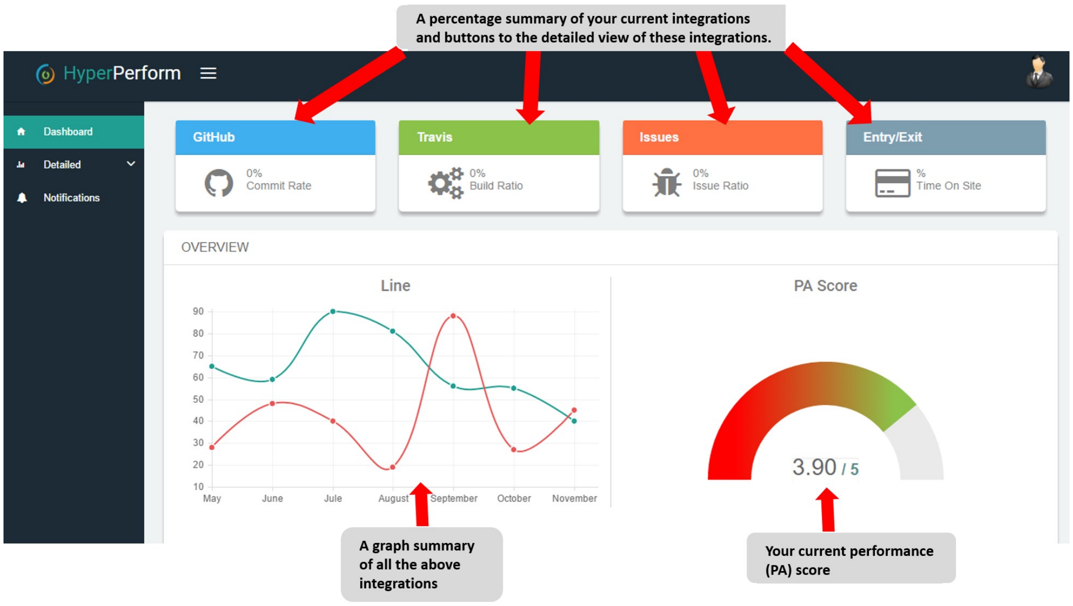


Figure 8: Dashboard

On this screen you are given a summarised view of all the integrations parts of the HyperPerform system. Note the four colour-coded panels on top, each of these panels represents an integration. These panels are click-able and will direct you to a details screen which will be discussed in the next section 5.

If you wish to logout then you merely click on the profile icon in the top right corner. Once clicked you will be presented with a small menu.

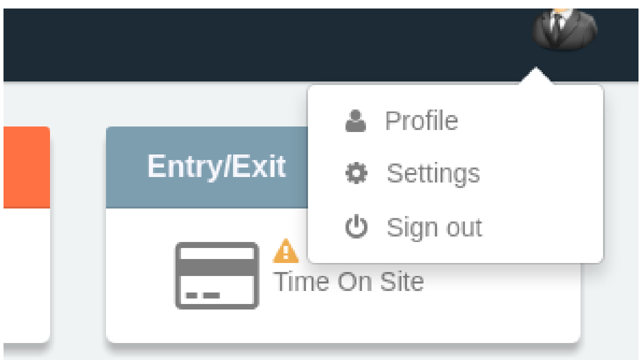


Figure 9: Dashboard

In this menu you have a few options to choose from. The Profile option will direct you to a profile page where you will be able to view and edit your current details.

The second option is a simple settings page where you can customize the dashboard.

And finally the Sign Out option can be used to log off the system. Once logged off you will be returned to the login screen in Figure 1.