Scheduler

Contents

[1. Background 2](#_Toc44311725)

[2. Scope 2](#_Toc44311726)

[3. Assumptions 2](#_Toc44311727)

[4. High Level Architecture 3](#_Toc44311728)

[5. Scheduler Features 3](#_Toc44311729)

[1.1 Login 3](#_Toc44311730)

[1.2 Manage Job Configuration 5](#_Toc44311731)

[1.3 View Job History 7](#_Toc44311732)

[1.4 Manage Schedules 7](#_Toc44311733)

[1.4.1 Ad-HOC Schedule: 8](#_Toc44311734)

[1.4.2 Cron Schedule: 8](#_Toc44311735)

[1.4.3 Delete Schedule: 9](#_Toc44311736)

[1.4.4 Stop Job: 9](#_Toc44311737)

[6. Technologies 10](#_Toc44311738)

[7. Deployment 10](#_Toc44311739)

[7.1 Deployment Steps 10](#_Toc44311740)

[7.2 Steps to change the properties 10](#_Toc44311741)

[8. Open/Closed Issues 12](#_Toc44311742)

# Background

The Scheduler application is designed to enable scheduling the jobs at a configured time or configured regular and recurring intervals.

Example Use cases:

**Adhoc Job:**

* To run a process on next Monday 10 am, schedule the job with date and time so the respective job will run at the specified time.

**Cron Job:**

* To schedule the job on every Monday 10 am, provide a cron expression to schedule the Job to run on every Monday at 10 am.

**One Time Job:**

* It is same as Adhoc job, but once job triggered it will Run 24/7, until we stop the Job.

**The ways of scheduling jobs:**

* Locally: To be scheduled the job should be available in the class path
* Over Internet: Schedule jobs by calling a REST service

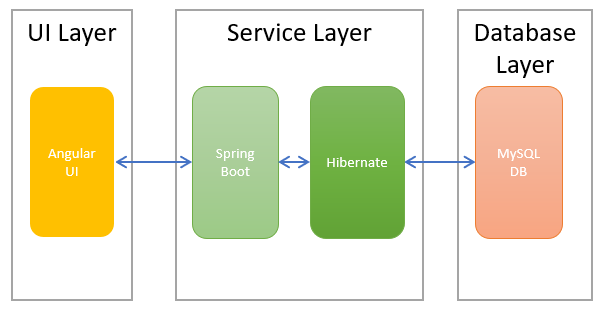
# Scope

The scheduler application is exposed with a UI. The user can create, configure, start and stop the jobs using the buttons provided. Scheduler calls the configured jobs on the specified time by using start method, stops the job by calling stop method. The start and stop methods should be implemented by the job owners. Scheduler calls the service start method with one String param only, it is service owner’s responsibility to parse the String param into required format.

# Assumptions

* Target Application should be developed in java and implement 2 public methods named start (optionally may take delimited string as parameter), stop. Make sure jar file in the classpath.
* Target Application should expose Rest API for start and stop the jobs.
* It is the responsibility of the target application how they handle the job stopping.
* Authenticated users have admin role only. These details need to enter from backend to **fhir\_scheduler.user** table.

# High Level Architecture



Fhir Scheduler Developed as 3 Layered Architecture.

* UI Layer
* Service Layer
* Database Layer

**UI Layer:** UI layer provides User interface for the end user to configure jobs and view existing status of the jobs. This UI layer developed using Angular9. It will make a REST calls to Service Layer for all user events and render latest data in the UI.

**Service Layer:** Service Layer expose REST services for UI layer. These REST services developed using Spring boot. Quartz scheduler (open source library) for scheduling tasks. For All Required Data for each Rest Service will be fetched from Database Layer via Hibernate layer.

**Database Layer:** Quartz Scheduler related table are residing in the fire\_scheduler schema in MySQL database. All CURD operations on these tables are achieved through ORM calls from Hibernate in Service Layer.

# Scheduler Features

Scheduler Provides below features to the end User.

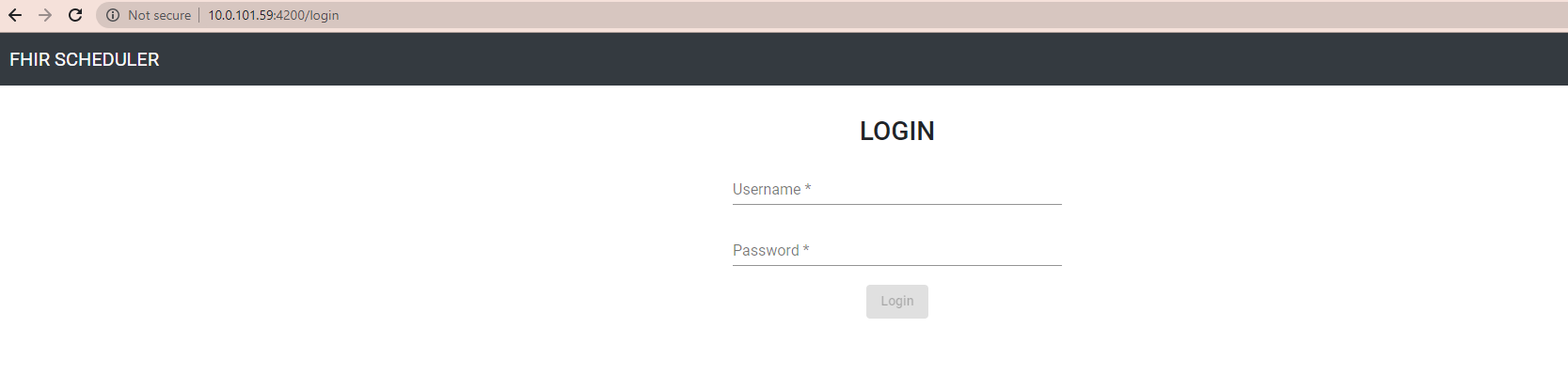
1. Login
2. Manage Job Configuration
3. View Job History
4. Manage Schedules

## Login

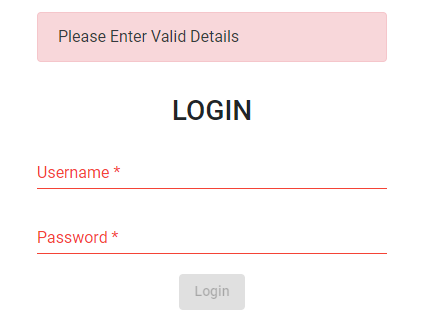
Only Authenticated user can open scheduler application. Username and password should be insert into **fhir\_scheduler.user**  table by db admin**.**

**Login to Application:**

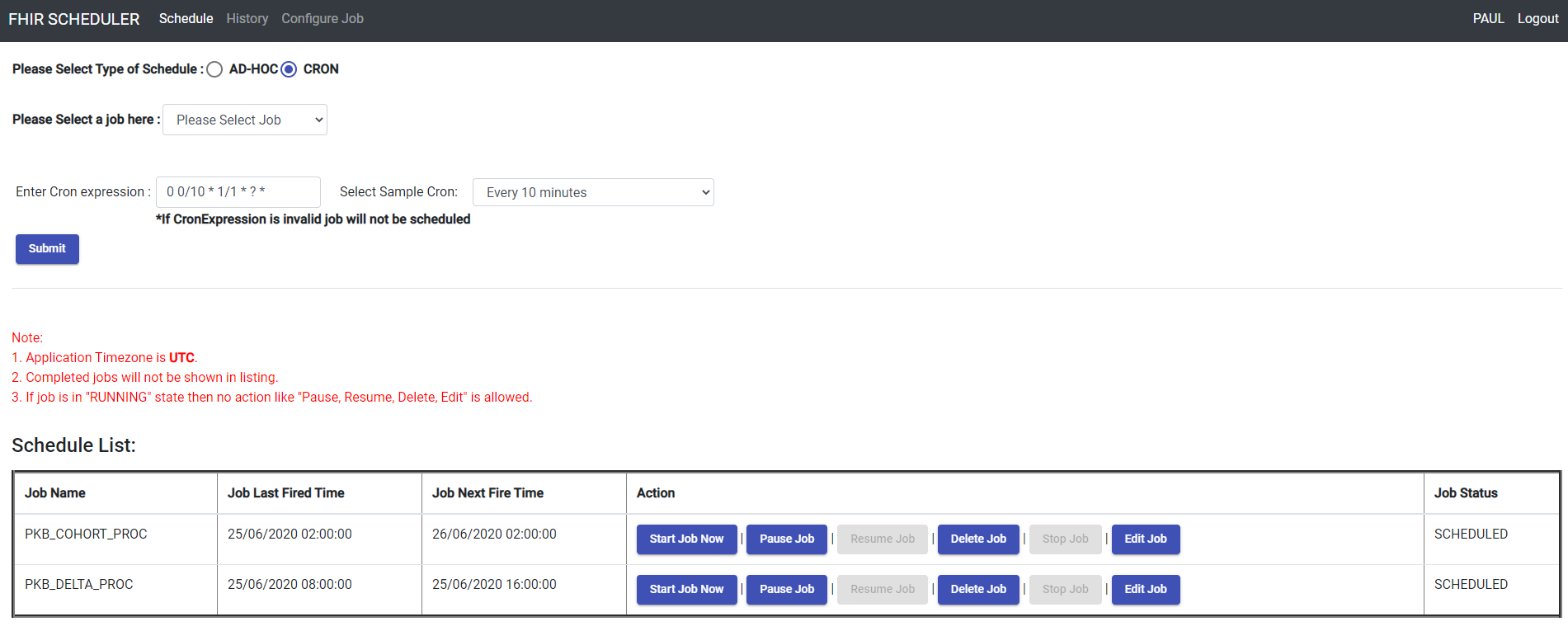
Open the application in any browser and enter the username , password in the login page of the application and click on login button.



If user credentials are incorrect, login screen shows the error message until he gives correct credentials.

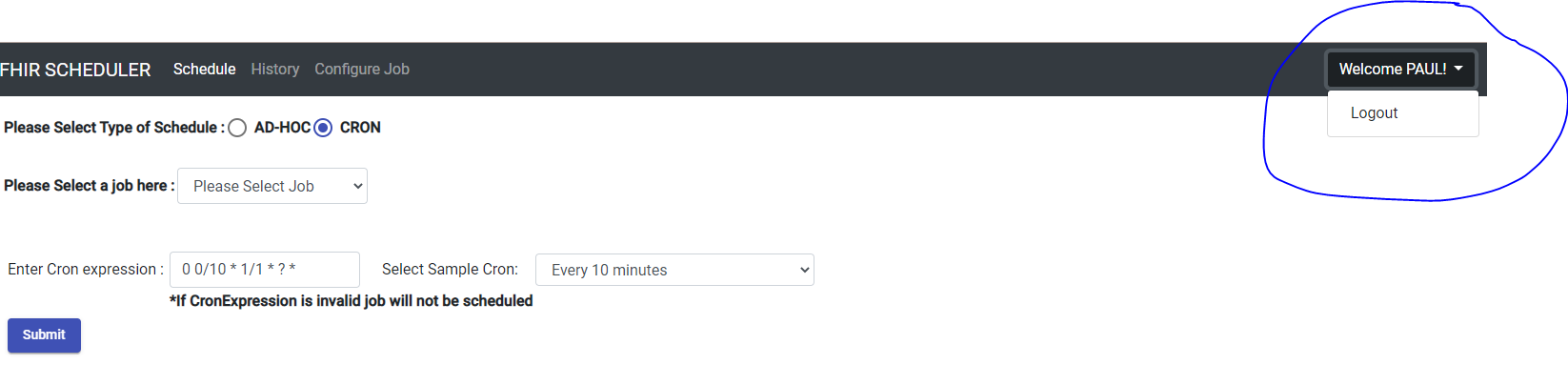


If user credentials are correct, he will navigate to the home page of the application



**Logout from Application:**

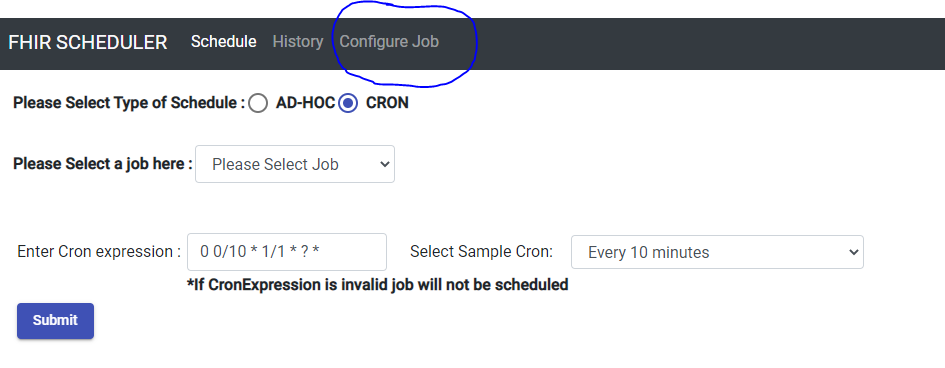
Form the home page, user clicks on logout link on the top right corner then application will be logged n off and redirected it to login page.



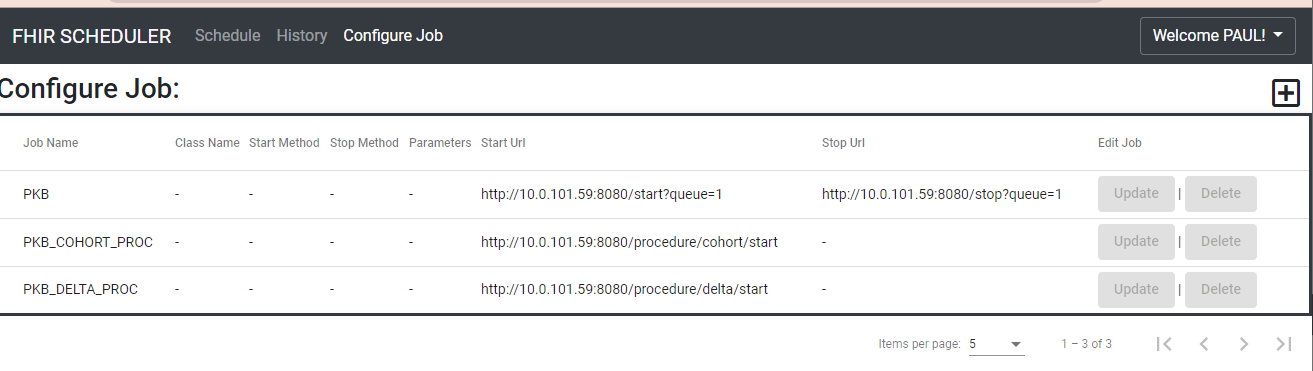
## Manage Job Configuration

Using scheduler, we can schedule a trigger to invoke public method in java class or make a call to synchronous REST service. User can configure new Job or update or delete existing configurations.

Authenticate User from Home page click on tab named Configure Job

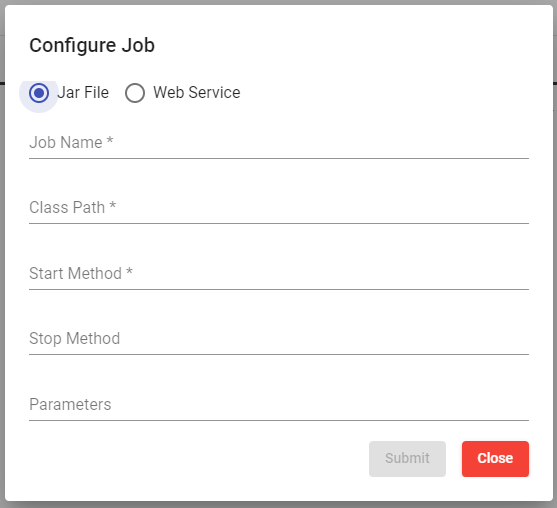


User will land into Configure Job Tab as shown below. This page contains paginated table that shows existing jobs. If JOB is currently scheduled (in use) then Update, Delete buttons will be disabled.



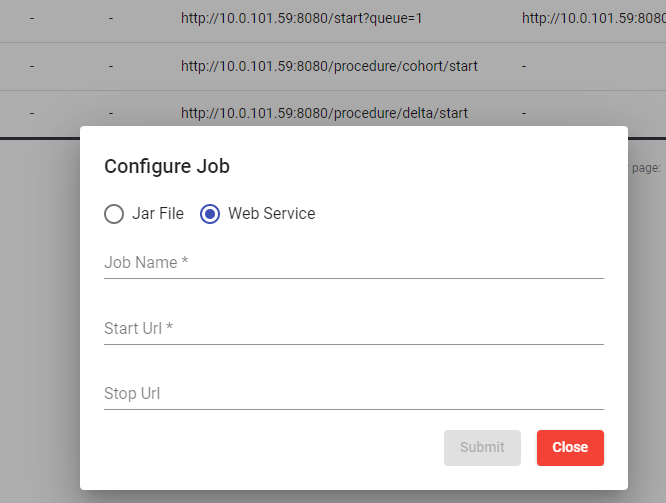
To add new Job , click on + icon on top right corner of the table. It will open a popup which is useful to create new JOB configuration for Jar file or Webservice end point.

**Configure new job Using Jar file Details:**



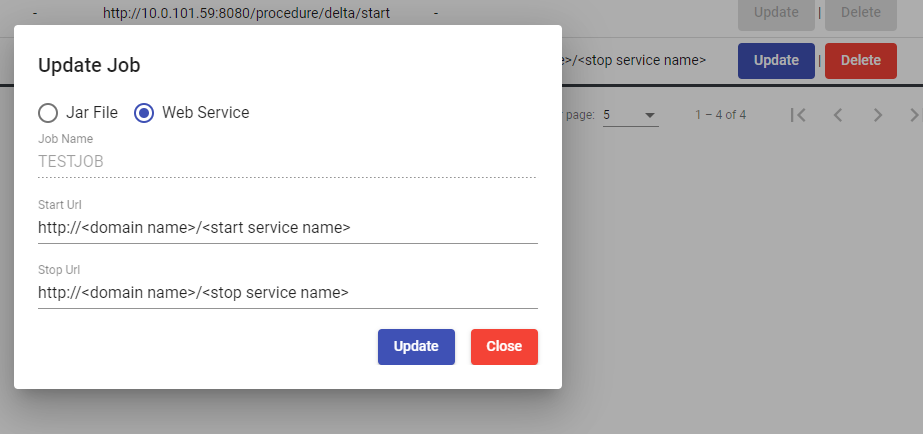
Here User should enter new Job Name, Class Path (Package name), start method name, start method parameter as delimited String. When User clicked on submit button new Job will be created and this job is available in dropdown in home page (Schedule Tab).

**Configure new job Using Rest Service End points:**



Here User should enter new Job Name, Start Rest service end point, Stop Rest service end point. When User clicked on submit button new Job will be created and this job is available in dropdown in home page (Schedule Tab).

**Update Job Configuration:**



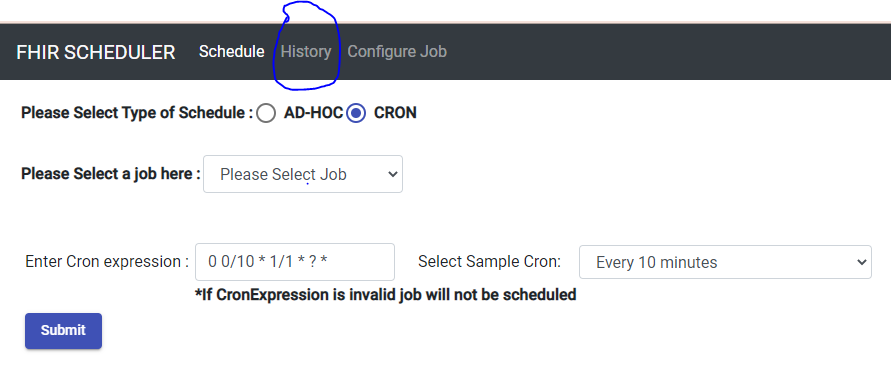
To Update Existing configuration, which is not yet schedule, User can click on Update button in the corresponding row. I the update popup edit the details and click on Update button. The changes will be updated.

**Delete Job Configuration:**

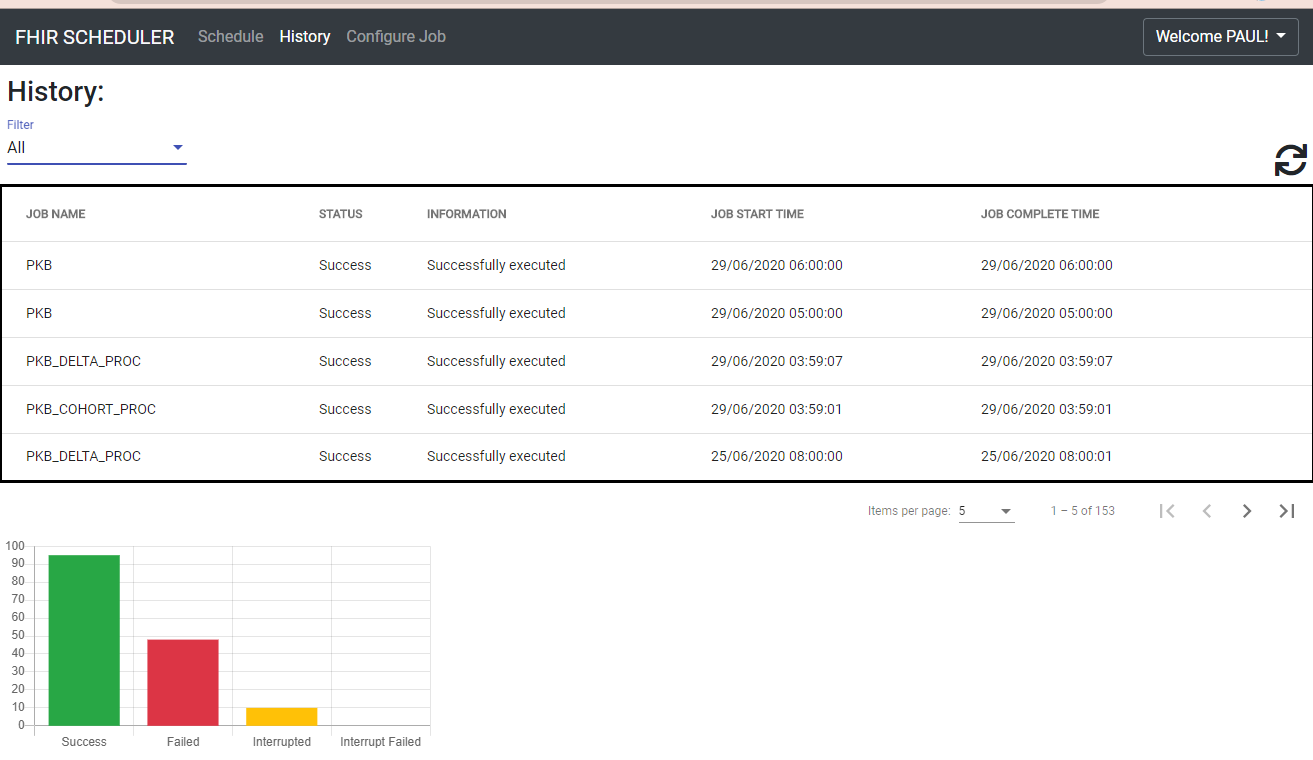
To Remove Existing configuration, which is not yet schedule, User can click on Delete button in the corresponding row. The Job will be deleted.

## View Job History

Using scheduler, we can see the history of jobs and its statuses. To View History of Jobs User can click on History tab in Home page.



User will be landed into History Tab shown below.

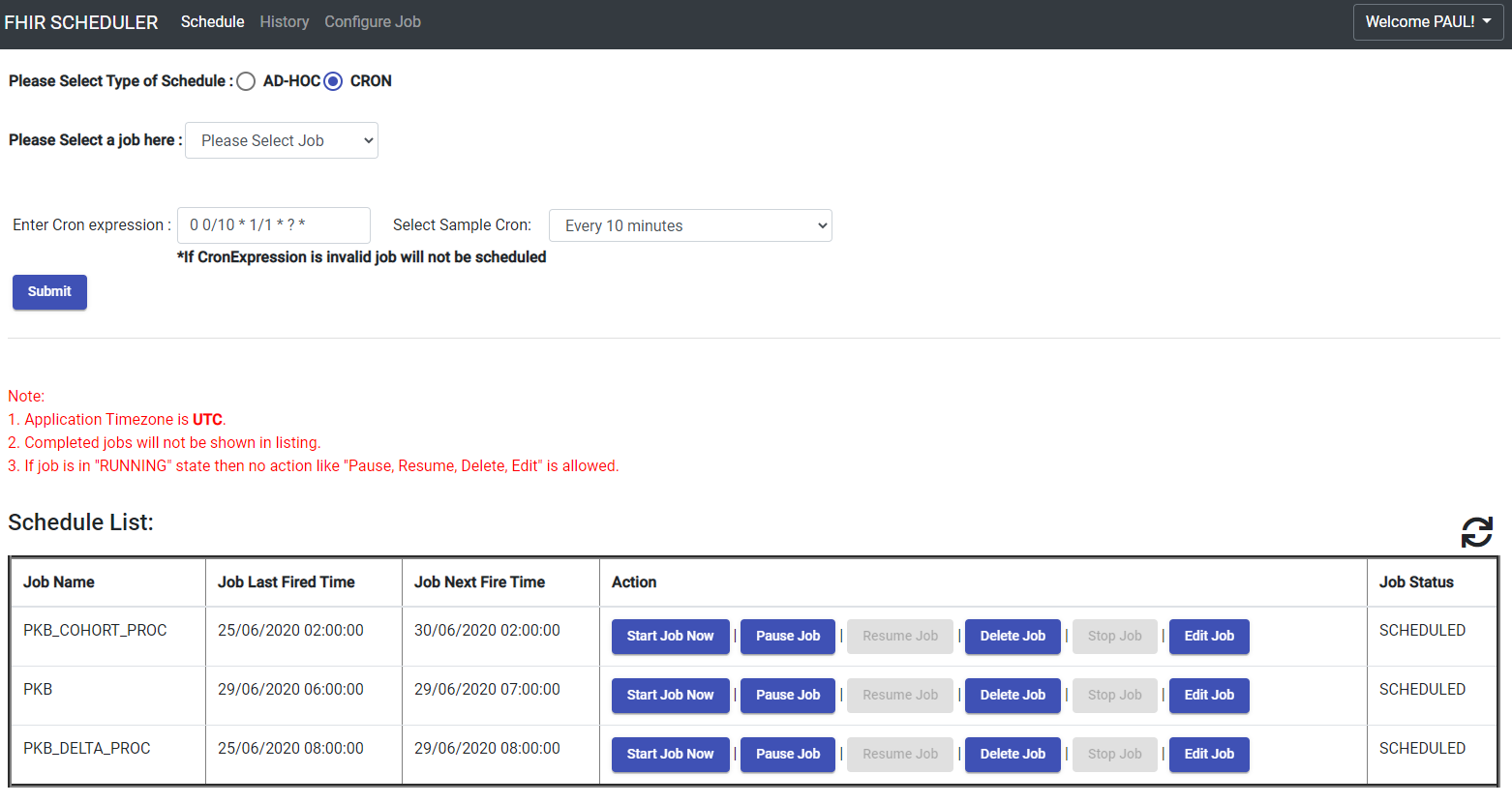


This page contains History paginated table and bar graph will represents the each status count. User can filer the records based on the status by using the filter options.

User should click on refresh icon to see the latest History.

## Manage Schedules

Authenticated user can schedule trigger time for each available job. From the home page of the application user can manage schedules

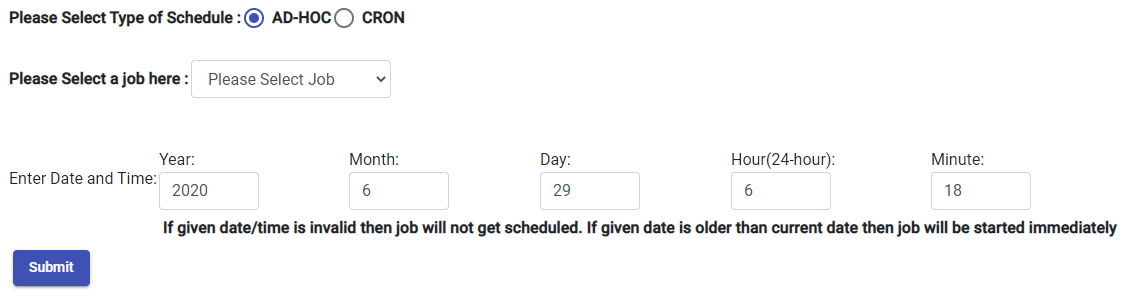


This Page split into two parts vertically. User can schedule one time schedule (adhoc schedule) or Cron schedule. Lower Part shows the current schedules list as paginated table. User need to click on refresh Icon to get the latest list.

All date-times in application shows Time in UTC.

### Ad-HOC Schedule:

User should select the top radio as ADHOC. Then panel will be refreshed to take date time for ADHOC schedule as shown below.

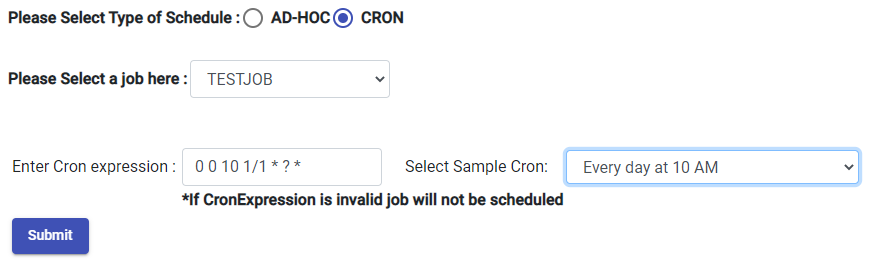


When user selects job and clicks on submit then new schedule will be created for the selected UTC date-time.

### Cron Schedule:

User should select the top radio as CRON. Then panel will be refreshed to take date time for CRON schedule as shown below.

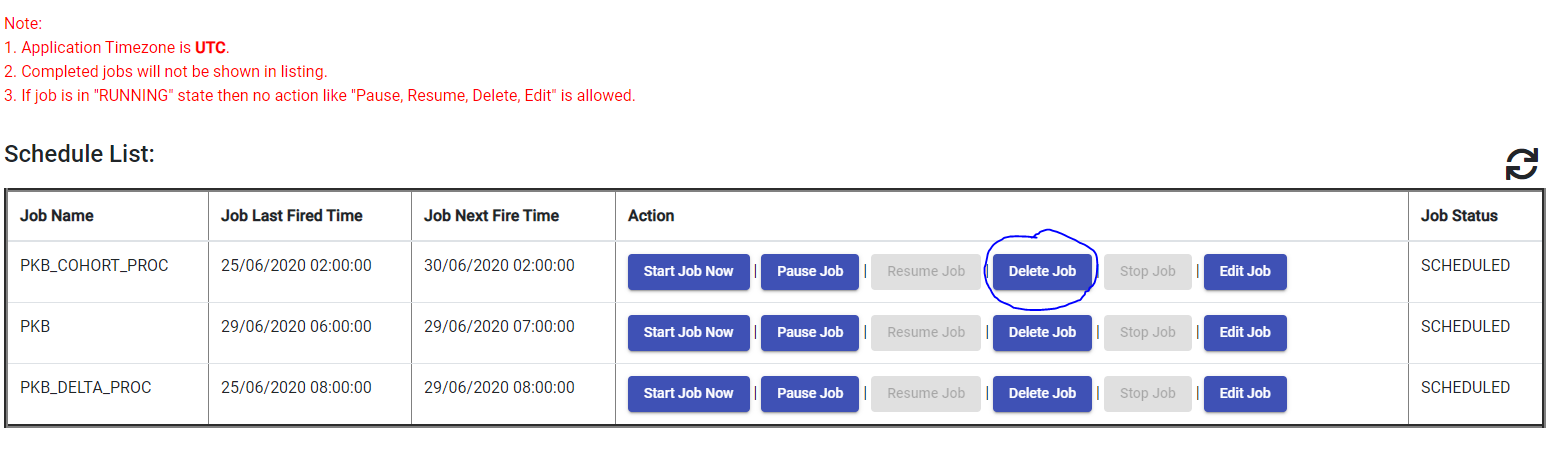
Useful link to generate cron expression: https://www.freeformatter.com/cron-expression-generator-quartz.html



When user selects job and clicks on submit then new schedule will be created for the selected CRON Expression.

### Delete Schedule:

To delete schedule, User can click on Delete Job Button in the corresponding row in Schedule List table.

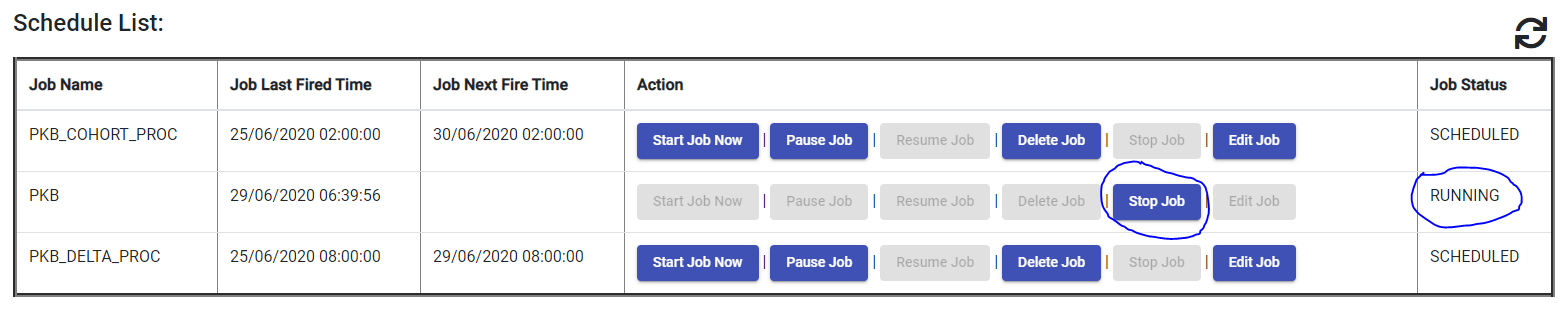


User will be notified as JOB deleted successfully. User need to refresh the table to see thy latest list of schedules.

Note: schedule cannot be deleted while running, however it can be stopped and deleted.

### Stop Job:

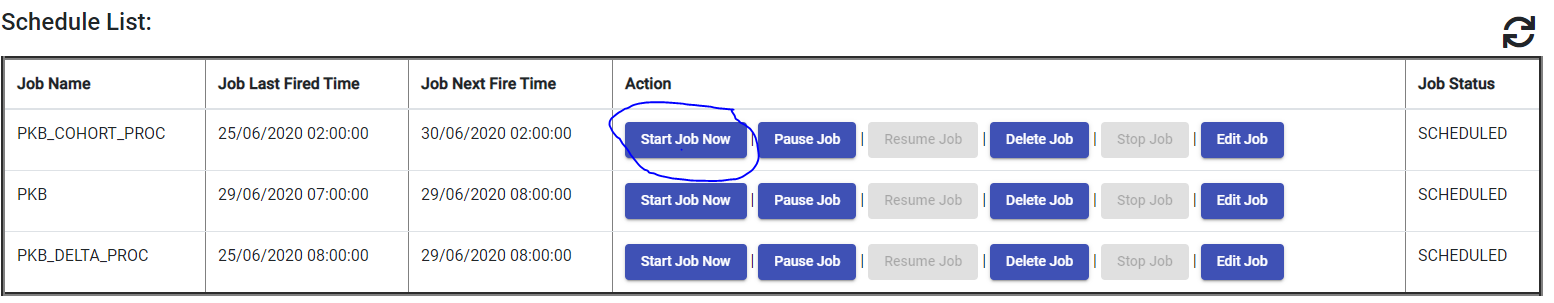
To stop job while it is running, User can click on Stop Job Button in the corresponding row in Schedule List table.



User will be notified as JOB Stopped successfully. User need to refresh the table to see thy latest list of schedules.

### Start Job Now:

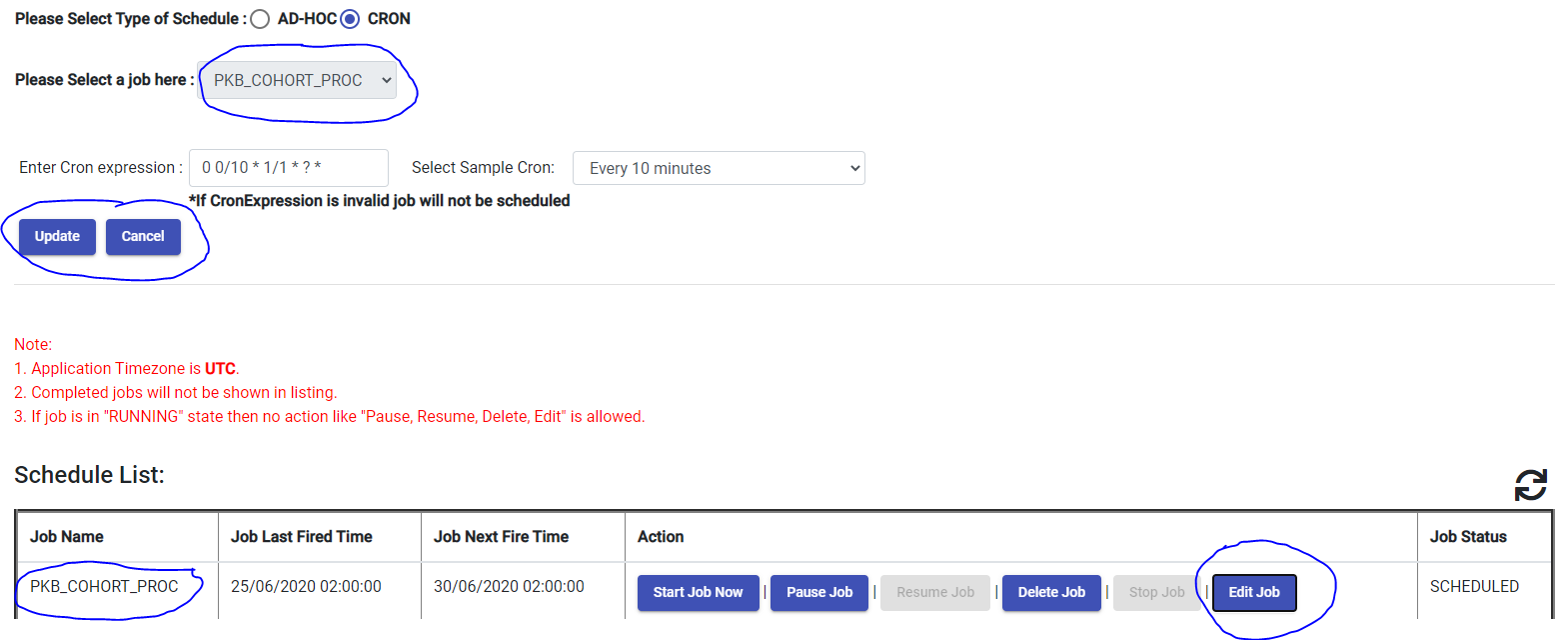
To stop job now which is scheduled later, user can click on Start Job Now button, in the corresponding row in Schedule List table.



User will be notified as JOB Started successfully. User need to refresh the table to see thy latest list of schedules.

### Edit Scheduled Job:

If user want to edit the schedule time of the previously scheduled jobs by clicking on Edit, in the corresponding row in Schedule List table. As a result, top panel changed to edit mode for the selected scheduled Job.



After selecting new time and click on Update button will update the schedule job. User need to refresh the table to see thy latest list of schedules.

# Technologies

Scheduler application is developed using the following software:

1. Spring boot (A server-side framework)
2. Angular9 (For front end)
3. Quartz scheduler to schedule jobs (A open source library to schedule jobs)
4. MySQL database (or any other DB) to store the scheduled jobs and the job details

# Deployment

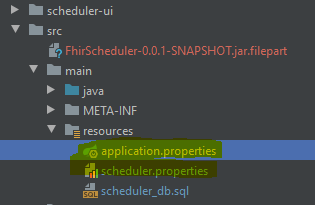
## Steps to change the properties

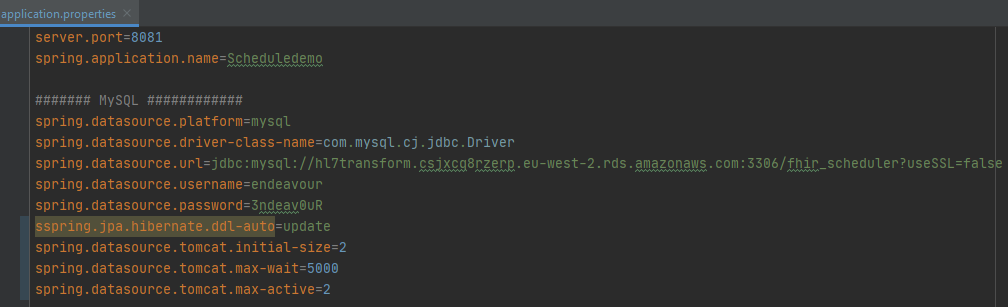
Incase there is a need to change the properties follow the below steps:

1. Open the application.properties file
2. In the application.properties file, specify the following properties:

application.properties. spring.datasource.platform=*Database name*

spring.datasource.driver-class-name= *Database Driver*  
spring.datasource.url=*Database\_url/ schema name*  
spring.datasource.username=*Username*  
spring.datasource.password=*Password*





1. In the scheduler.properties file configure following properties:

org.quartz.scheduler.instanceName=Name of the scheduler

org.quartz.scheduler.instanceId=AUTO

org.quartz.threadPool.threadCount=Default 50 (can be configured based on the capacity)

#Choose the persistence store either DB or RAM as mentioned below

org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX # to store the jobs in DB org.quartz.jobStore.class=org.quartz.impl.ramjobstore.JobStoreTX # to store the jobs in RAM

#Select the Type of JDBC Store

org.quartz.jobStore.driverDelegateClass =org.quartz.impl.jdbcjobstore.StdJDBCDelegate

org.quartz.jobStore.useProperties=true

#Wait time after which a trigger is considered as misfired

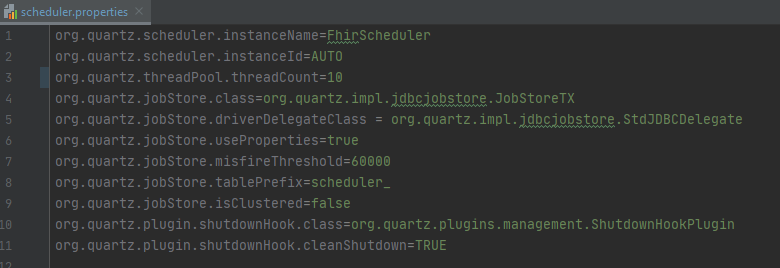
org.quartz.jobStore.misfireThreshold=60000

#Specify the prefix for the tables in database

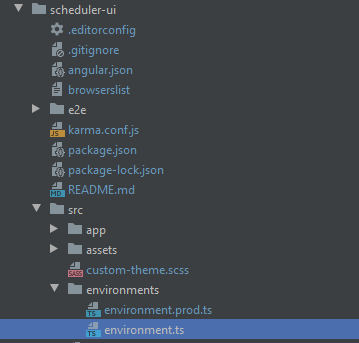
org.quartz.jobStore.tablePrefix=scheduler\_

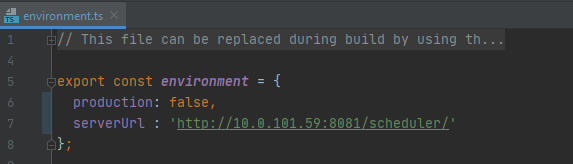
#Clustered environment flag

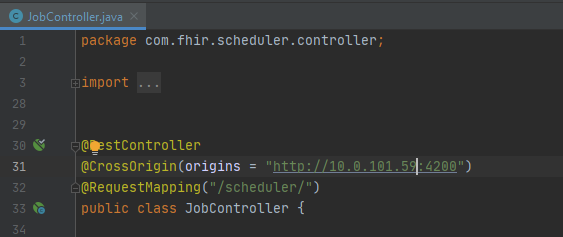
org.quartz.jobStore.isClustered=false



1. Change the IP address in environment.ts and JobController.java







## Deployment Steps

The setup instructions are as mentioned below:

1. Download Scheduler from below Github path

<https://github.com/endeavourhealth-discovery/PatientKnowsBest/tree/master/scheduler>

1. Configure the database to enable scheduler to persist job details in the database.

Execute Scheduler.sql script in mysql datable

1. Go to scheduler/taget directory and type blow command to run scheduler service layer

# java -jar FhirScheduler-0.0.2-SNAPSHOT.jar > schedulerlog.txt &

1. Install NodeJS
2. Open command prompt/terminal, run the command given below:

npm install -g @angular/cli

1. Goto scheduler/scheduler-ui directory and excute below commands to make it up UI application.

* npm install
* node\_modules/@angular/cli/bin/ng serve --host <ipaddress> >schduler\_ui\_log.txt &

1. To open UI application ,

https://<ipaddress>:4200

# Open/Closed Issues

Any issues discussed and open/closed.