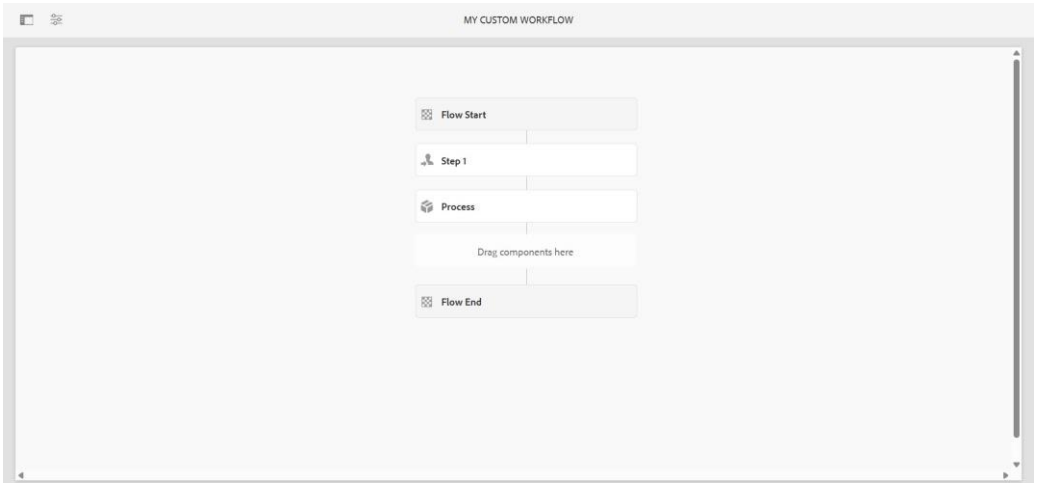
AEM TASK 06

1.Create Custom Workflow (my custom workflow)



2.Create custom workflow process and print the page title in logs and run this workflow in page so that it can give some metadata in logs.

03.04.2025 16:30:48.482 *INFO* [JobHandler: /var/workflow/instances/server0/2025-0403/my-custom-workflow_1:/content/myTraining/us/en/about-us] com.adobe.granite.workflow.core.job.JobHandler Start of Workflow Execution: Title=My Custom Workflow, Process=com.myproject.workflows.MyCustomWorkflowProcess, Workfront-objecttype=null
03.04.2025 16:30:48.482 *INFO* [[0:0:0:0:0:0:0:1] [1743678048477] GET /editor.html/content/myTraining/us/en/about-us.html HTTP/1.1] com.day.cq.wcm.core.impl.designer.SearchPathLimiter Search path limiter configured with searchPathLimiterFeatureToggleOn: true and searchPathThreshold: true.
03.04.2025 16:30:48.482 *INFO* [JobHandler: /var/workflow/instances/server0/2025-0403/my-custom-workflow_1:/content/myTraining/us/en/about-us] com.myproject.workflows.MyCustomWorkflowProcess Executing Workflow Process for Page: /content/myTraining/us/en/about-us
03.04.2025 16:30:48.482 *INFO* [JobHandler: /var/workflow/instances/server0/2025-04-03/my-custom-workflow_1:/content/myTraining/us/en/about-us] com.adobe.granite.workflow.core.job.JobHandler End of Workflow Execution - Title: My Custom Workflow, Process: com.myproject.workflows.MyCustomWorkflowProcess

3.Create Event handler in aem and print the resource path in logs.

package com.myTraining.core.listeners; import

```
org.apache.sling.api.resource.Resource; import
org.apache.sling.api.resource.ResourceResolver; import
org.apache.sling.api.resource.observation.ResourceChange; import
org.apache.sling.api.resource.observation.ResourceChangeListener; import
org.osgi.service.component.annotations.Component; import
org.slf4j.Logger; import org.slf4j.LoggerFactory; import java.util.List;
@Component(
   service =
ResourceChangeListener.class,
immediate = true,     property = {
       ResourceChangeListener.PATHS + "=/content/myTraining",
       ResourceChangeListener.CHANGES + "=ADDED",
       ResourceChangeListener.CHANGES + "=REMOVED",
       ResourceChangeListener.CHANGES + "=CHANGED"
   }
)
public class MyEventHandler implements ResourceChangeListener {
   private static final Logger LOG = LoggerFactory.getLogger(MyEventHandler.class);
   @Override
   public void onChange(List<ResourceChange> changes) {
for (ResourceChange change : changes) {
         LOG.info("Resource Changed: Path = {}, Type = {}", change.getPath(), change.getType());
      }
   }
}
```

03.04.2025 17:53:41.770 *INFO* [sling-oak-observation-13] com.myTraining.core.listeners.MyEventHandler Resource Changed: Path = /content/myTraining/TestingNode, Type = ADDED

4. Create sling job to print hello world message in logs package

```
com.myTraining.core.jobs;
import org.apache.sling.event.jobs.consumer.JobConsumer;
import org.osgi.service.component.annotations.Component;
import org.slf4j.Logger; import org.slf4j.LoggerFactory;
@Component(
```

```
   service = JobConsumer.class,
immediate = true,
   property = {"job.topics=com/myTraining/helloworld"}
)
public class HelloWorldJob implements JobConsumer {
   private static final Logger LOG = LoggerFactory.getLogger(HelloWorldJob.class);
   @Override
   public JobResult process(org.apache.sling.event.jobs.Job job)
{       LOG.info("Hello World from Sling Job!");        return
JobResult.OK;
   }
}
```

03.04.2025 18:53:58.490 *INFO* [sling-threadpool-81d78f1c-2baf-49ae-b486-cbbcaaeb91d5(apache-sling-job-thread-pool)-24-<main queue>(com/myTraining/helloworld)] com.myTraining.core.jobs.HelloWorldJob Hello World from Sling Job!

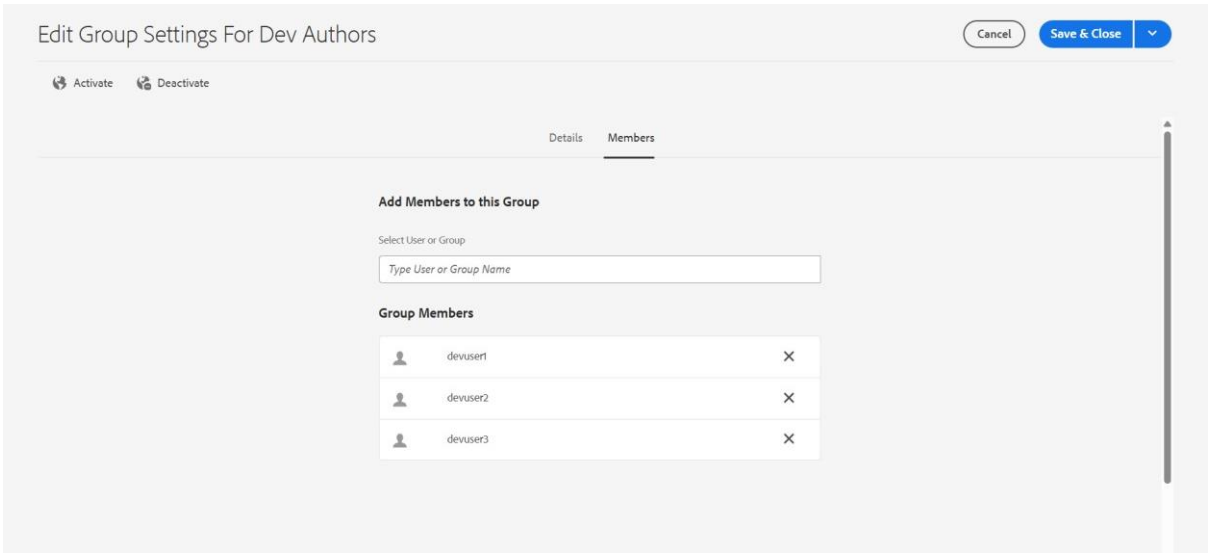
5. Create one scheduler to print the yellow world in logs in every 5 mins through custom configuration .

```
package com.myTraining.core.schedulers;
import org.apache.sling.commons.scheduler.ScheduleOptions; import
org.apache.sling.commons.scheduler.Scheduler;
import org.osgi.service.component.annotations.Activate; import
org.osgi.service.component.annotations.Component; import
org.osgi.service.component.annotations.Deactivate; import
org.osgi.service.component.annotations.Modified; import
org.osgi.service.component.annotations.Reference; import
org.osgi.service.metatype.annotations.AttributeDefinition; import
org.osgi.service.metatype.annotations.Designate; import
org.osgi.service.metatype.annotations.ObjectClassDefinition; import
org.slf4j.Logger; import org.slf4j.LoggerFactory;
@Component(service = Runnable.class, immediate = true)
@Designate(ocd = HelloWorldScheduler.Config.class)
public class HelloWorldScheduler implements Runnable {
```

```
    private static final Logger LOG = LoggerFactory.getLogger(HelloWorldScheduler.class);
    @Reference
    private Scheduler scheduler;
    private String schedulerName = "HelloWorldScheduler";
@ObjectClassDefinition(name = "Hello World Scheduler Config")
public @interface Config {
        @AttributeDefinition(name = "Cron Expression", description = "Schedule job every 5 minutes")
        String scheduler_expression() default "0 */5 * * * ?";
    }
    @Activate
@Modified
    protected void activate(Config config) {
        ScheduleOptions options =
scheduler.EXPR(config.scheduler_expression());
options.name(schedulerName);        options.canRunConcurrently(false);
scheduler.schedule(this, options);
        LOG.info("Hello World Scheduler Activated.");
    }

    @Deactivate
    protected void deactivate() {
scheduler.unschedule(schedulerName);
        LOG.info("Hello World Scheduler Deactivated.");
    }
    @Override
public void run() {
        LOG.info("Hello World from Scheduler!");
    }
}
```

6. Create 3 users and add them in a group(Dev author create this new group) and give permission to read only for /content and /dam folder only and they should have replication access as well.


Edit Group Settings For Dev Authors
Cancel
Save & Close
Activate
Deactivate
Details
Members
Add Members to this Group
Select User or Group
Type User or Group Name
Group Members
devuser1
devuser2
devuser3