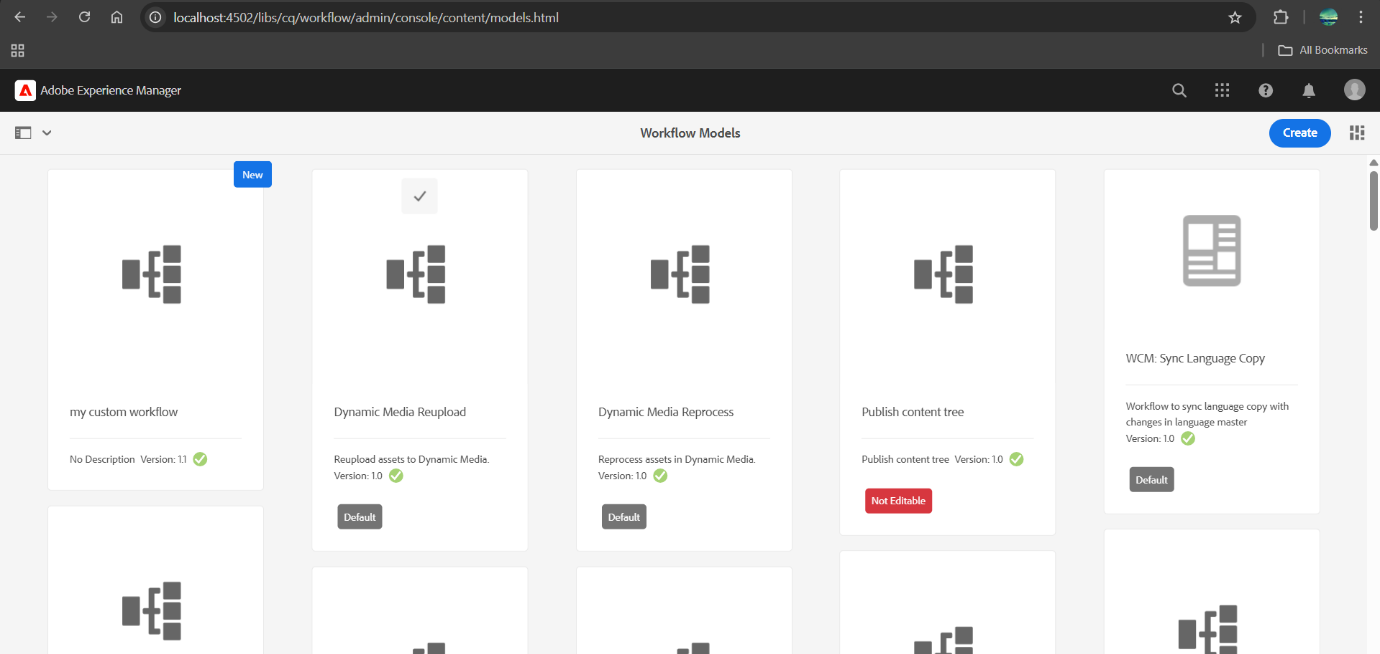
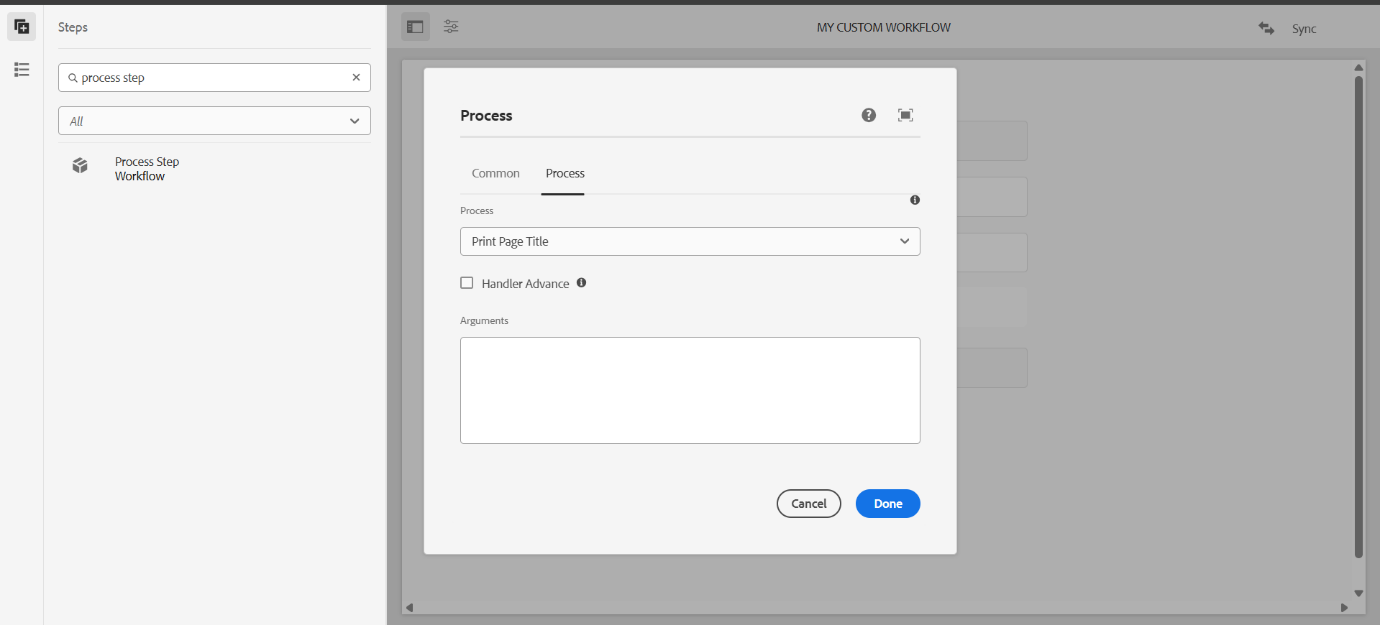
**1.Create Custom Workflow (my custom workflow)**

****

**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**

****

**⚙️ Workflow triggered for path: /content/us/en/news/Article-1**

**📄 Page Title: Article 1**

**🔍 Metadata - executedBy: admin**

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

**Event Handler Class**

package com.trainingProj.core.listeners;

import org.osgi.service.component.annotations.Component;

import org.osgi.service.component.propertytypes.ServiceDescription;

import org.osgi.framework.Constants;

import org.apache.sling.api.resource.ResourceResolverFactory;

import org.apache.sling.api.resource.ResourceResolver;

import org.apache.sling.jcr.api.SlingRepository;

import javax.jcr.RepositoryException;

import javax.jcr.observation.Event;

import javax.jcr.observation.EventListener;

import javax.jcr.observation.ObservationManager;

import javax.jcr.Session;

import javax.jcr.Node;

import org.osgi.service.component.annotations.Reference;

import javax.annotation.Activated;

@Component(

service = EventListener.class,

immediate = true,

property = {

Constants.SERVICE\_DESCRIPTION + "=Custom Page Event Listener"

}

)

public class PageEventHandler implements EventListener {

@Reference

private SlingRepository repository;

private Session session;

@Activated

public void activate() {

try {

session = repository.loginService("datawrite", null);

ObservationManager observationManager = session.getWorkspace().getObservationManager();

observationManager.addEventListener(

this, // EventListener

Event.NODE\_ADDED | Event.NODE\_REMOVED | Event.PROPERTY\_CHANGED,

"/content", // Path to listen to

true, // isDeep?

null, // UUIDs filter

null, // NodeTypes filter

false // NoLocal

);

} catch (RepositoryException e) {

e.printStackTrace();

}

}

@Override

public void onEvent(javax.jcr.observation.EventIterator events) {

while (events.hasNext()) {

Event event = events.nextEvent();

try {

String path = event.getPath();

System.out.println("📌 AEM Event Triggered at Path: " + path);

} catch (RepositoryException e) {

e.printStackTrace();

}}}}

**In logs:**

**AEM Event Triggered at Path: /content/us/en/news/Article-1/jcr:content**

**4.create sling job to print hello world message in logs**

**Sling job consumer class**

package com.example.core.jobs;

import org.apache.sling.event.jobs.Job;

import org.apache.sling.event.jobs.consumer.JobConsumer;

import org.osgi.service.component.annotations.Component;

@Component(

service = JobConsumer.class,

immediate = true,

property = {

JobConsumer.PROPERTY\_TOPICS + "=com/example/helloworld"

})

public class HelloWorldJobConsumer implements JobConsumer {

@Override

public JobResult process(Job job) {

// Print to AEM logs

System.out.println("✅ Sling Job Triggered - Hello World!");

return JobResult.OK;

}}

**5. Create one schedular to print the yellow world in logs in every 5 mins through custom configuration using cron expression.**

**Scheduler class**

package com.trainingProj.core.schedulers;

import org.osgi.service.component.annotations.Component;

import org.osgi.service.metatype.annotations.Designate;

import org.osgi.service.component.annotations.Activate;

import org.osgi.service.component.annotations.Modified;

import org.osgi.framework.Constants;

import org.osgi.service.metatype.annotations.ObjectClassDefinition;

import org.osgi.service.metatype.annotations.AttributeDefinition;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.concurrent.atomic.AtomicBoolean;

import org.apache.sling.commons.scheduler.ScheduleOptions;

import org.apache.sling.commons.scheduler.Scheduler;

@Component(

service = Runnable.class,

immediate = true,

configurationPid = "com.example.core.schedulers.YellowWorldScheduler"

)

@Designate(ocd = YellowWorldScheduler.Config.class)

public class YellowWorldScheduler implements Runnable {

private static final Logger LOG = LoggerFactory.getLogger(YellowWorldScheduler.class);

@ObjectClassDefinition(name = "💛 Yellow World Scheduler Config")

public @interface Config {

@AttributeDefinition(name = "Cron Expression", description = "Cron expression for scheduling the task")

String scheduler\_expression() default "0 0/5 \* \* \* ?";

}

@org.osgi.service.component.annotations.Reference

private Scheduler scheduler;

private final AtomicBoolean isScheduled = new AtomicBoolean(false);

@Activate

@Modified

protected void activate(Config config) {

String expression = config.scheduler\_expression();

ScheduleOptions scheduleOptions = scheduler.EXPR(expression);

scheduleOptions.name("YellowWorldSchedulerJob");

scheduleOptions.canRunConcurrently(false);

if (isScheduled.compareAndSet(false, true)) {

scheduler.schedule(this, scheduleOptions);

LOG.info("✅ YellowWorld Scheduler Registered with cron: {}", expression);

} }

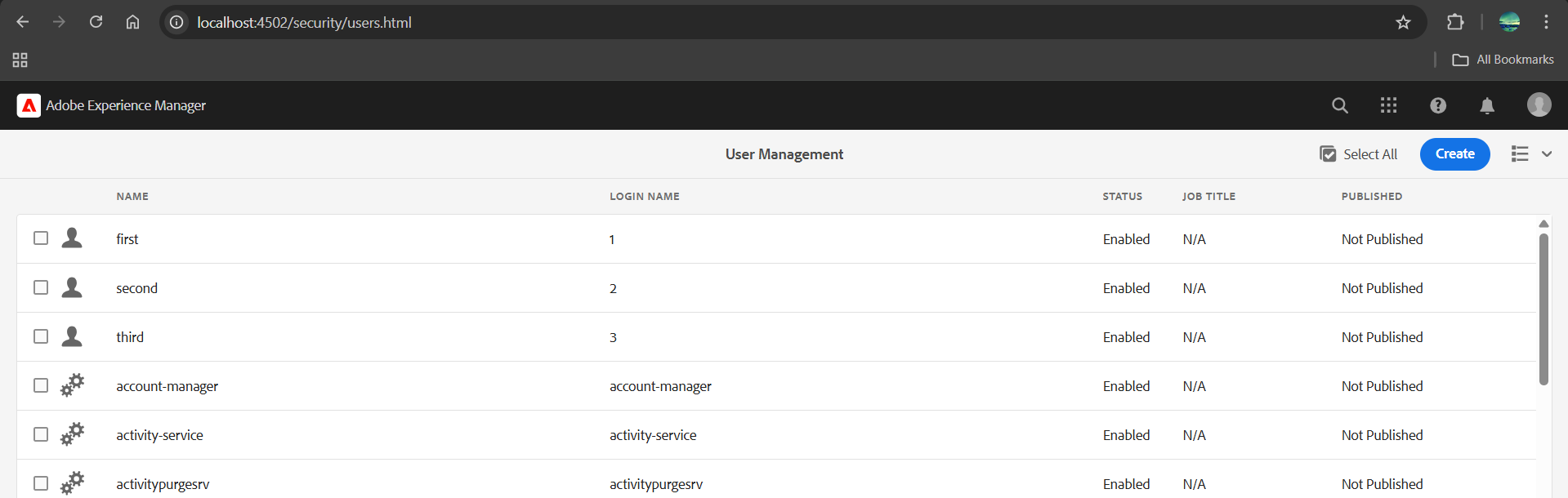
@Override

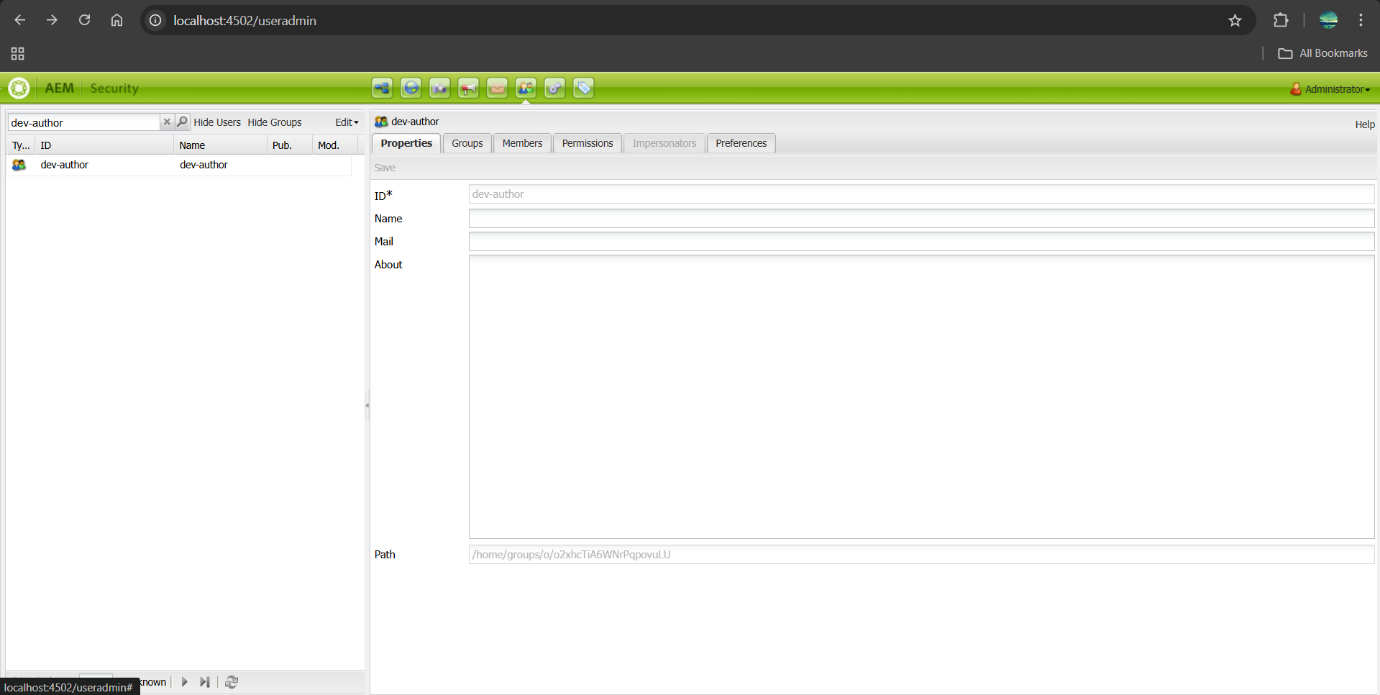
public void run() {

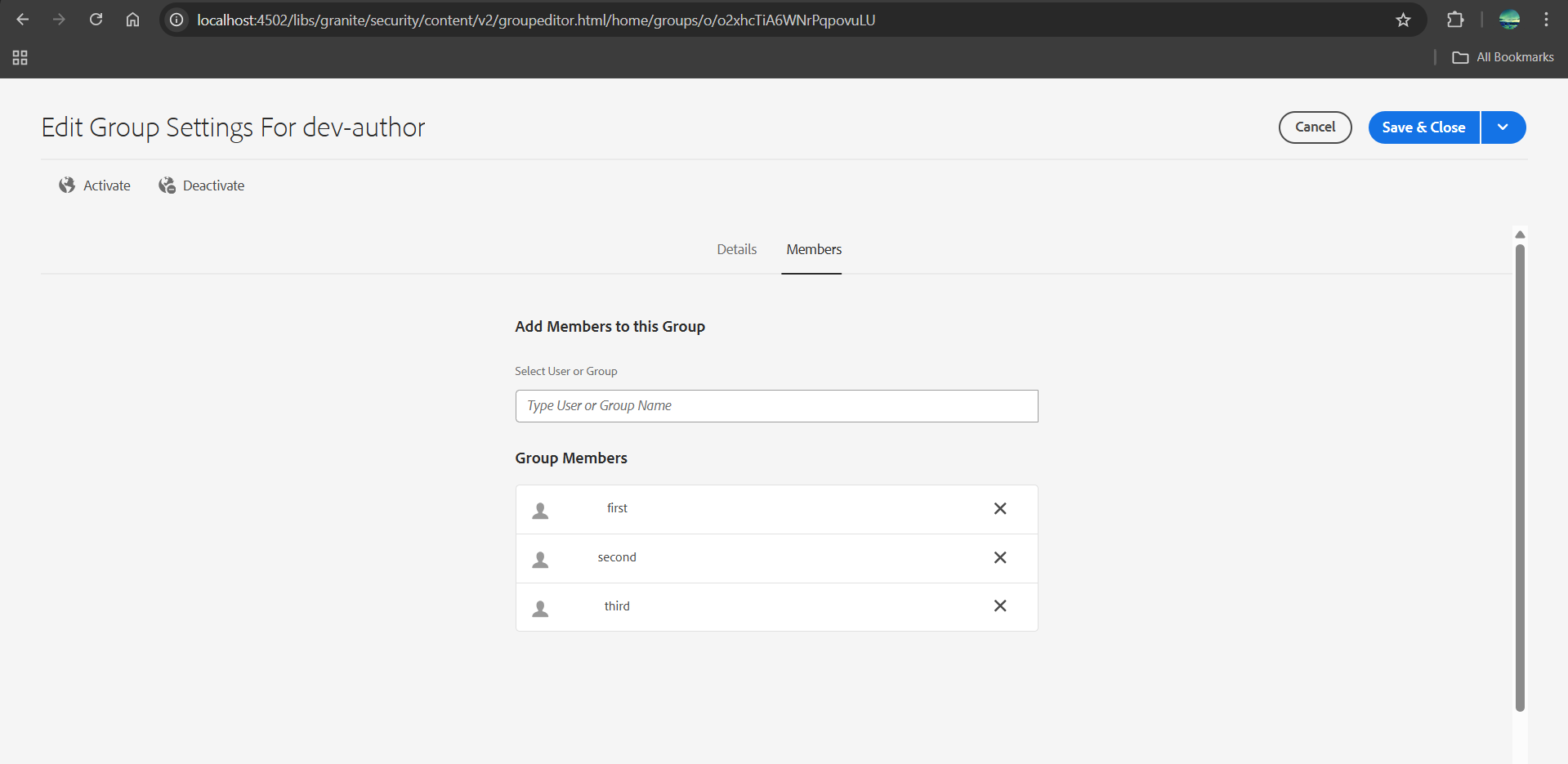
LOG.info("💛 Yellow World");

}}

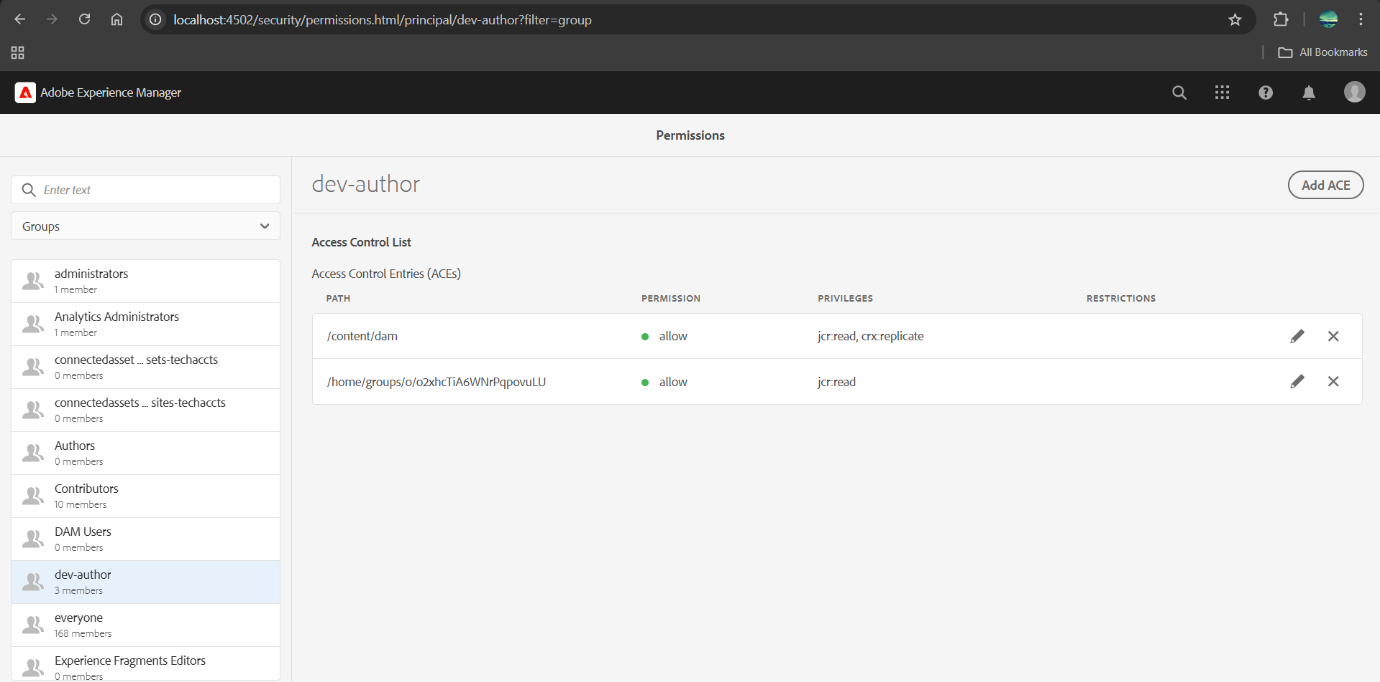
**6. Create 3 users and add them in a group(Dev author create this new group)**

****

****

****

**give permission to read only for /content/dam folder only and they should have replication access as well.**

****