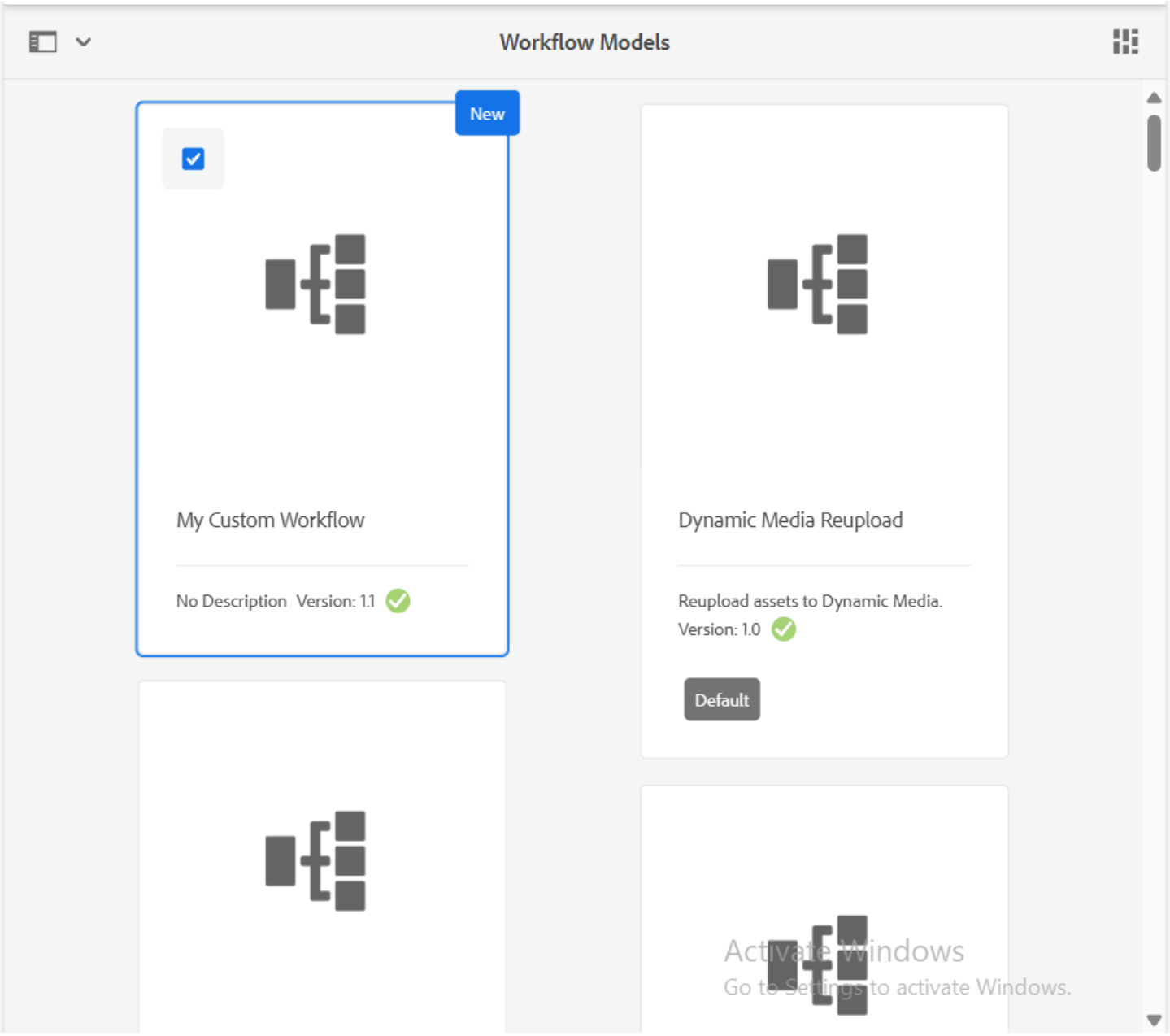
ASSIGNMENT(26.03.2025)

1.Create Custom Workflow (my custom workflow) and



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

package com.myTraining.workflow;

```
import com.adobe.granite.workflow.WorkflowException;
import com.adobe.granite.workflow.exec.WorkItem;
import com.adobe.granite.workflow.exec.WorkflowSession;
import com.adobe.granite.workflow.metadata.MetaDataMap;
import com.adobe.granite.workflow.model.WorkflowProcess;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.api.resource.ResourceResolver;
import org.osgi.service.component.annotations.Component;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@Component(service = WorkflowProcess.class,
        property = {"process.label=Custom Log Page Title"})
public class PageTitleLogger implements WorkflowProcess {

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

    @Override
    public void execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap
args) throws WorkflowException {
        String payloadPath = workItem.getWorkflowData().getPayload().toString();
        ResourceResolver resolver = workflowSession.adaptTo(ResourceResolver.class);

        if (resolver != null) {
            Resource resource = resolver.getResource(payloadPath + "/jcr:content");
            if (resource != null) {
```

```
            String title = resource.getValueMap().get("jcr:title", String.class);
            LOG.info("Page Title: {}", title);
            LOG.info("Metadata: {}", resource.getValueMap());
        } else {
            LOG.warn("Resource not found at: {}", payloadPath);
        }
    } else {
        LOG.error("Failed to adapt WorkflowSession to ResourceResolver.");
    }
  }
}
```

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

```
package com.myTraining.event;

import org.apache.sling.api.SlingConstants;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.api.resource.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",
        ResourceChangeListener.CHANGES + "=" + SlingConstants.TOPIC_RESOURCE_ADDED,
        ResourceChangeListener.CHANGES + "=" + SlingConstants.TOPIC_RESOURCE_CHANGED,
        ResourceChangeListener.CHANGES + "=" + SlingConstants.TOPIC_RESOURCE_REMOVED
    }
)
public class ResourceChangeListenerImpl implements ResourceChangeListener {

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

    @Override
    public void onChange(List<ResourceChange> changes) {
        for (ResourceChange change : changes) {
            LOG.info("Resource Change Detected: {} - Path: {}", change.getType(), change.getPath());
        }
    }
}
```

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

```
package com.myTraining.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,
    property = {
        JobConsumer.PROPERTY_TOPICS + "=com/myTraining/jobs/helloWorld"
    }
)
public class HelloWorldJob implements JobConsumer {

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

    @Override
    public JobResult process(Job job) {
        LOG.info("Hello World! This is a Sling Job running.");
        return JobResult.OK;
    }
}
```

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

```
package com.myTraining.scheduler;

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.ObjectClassDefinition;
import org.osgi.service.metatype.annotations.Designate;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@Component(service = Runnable.class, immediate = true)
@Designate(ocd = YellowWorldScheduler.Config.class)
public class YellowWorldScheduler implements Runnable {

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

    @Reference
    private Scheduler scheduler;

    private String cronExpression;
    private org.apache.sling.commons.scheduler.JobHandle jobHandle;

    @ObjectClassDefinition(name = "Yellow World Scheduler Configuration")
    public @interface Config {
```

```
    @AttributeDefinition(name = "Cron Expression",
        description = "Cron expression for scheduler execution")
    String cronExpression() default "0 */5 * * * ?";
}

@Activate
@Modified
protected void activate(Config config) {
    this.cronExpression = config.cronExpression();
    scheduleJob();
}

@Deactivate
protected void deactivate() {
    if (jobHandle != null) {
        scheduler.unschedule(jobHandle);
    }
}

private void scheduleJob() {
    ScheduleOptions options = scheduler.EXPR(cronExpression);
    options.name("YellowWorldSchedulerJob");
    options.canRunConcurrently(false);
    jobHandle = scheduler.schedule(this, options);
    LOG.info("Yellow World Scheduler Job Scheduled with Cron: {}", cronExpression);
}
```

```
    @Override
    public void run() {
        LOG.info("Yellow World!");
    }
}
```

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.