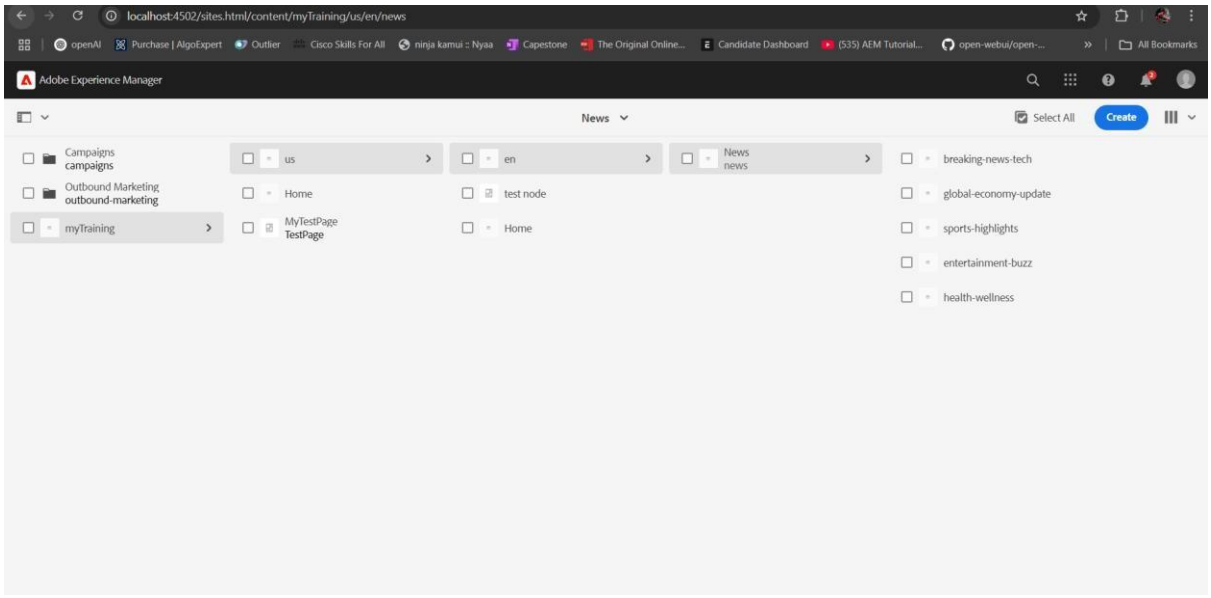
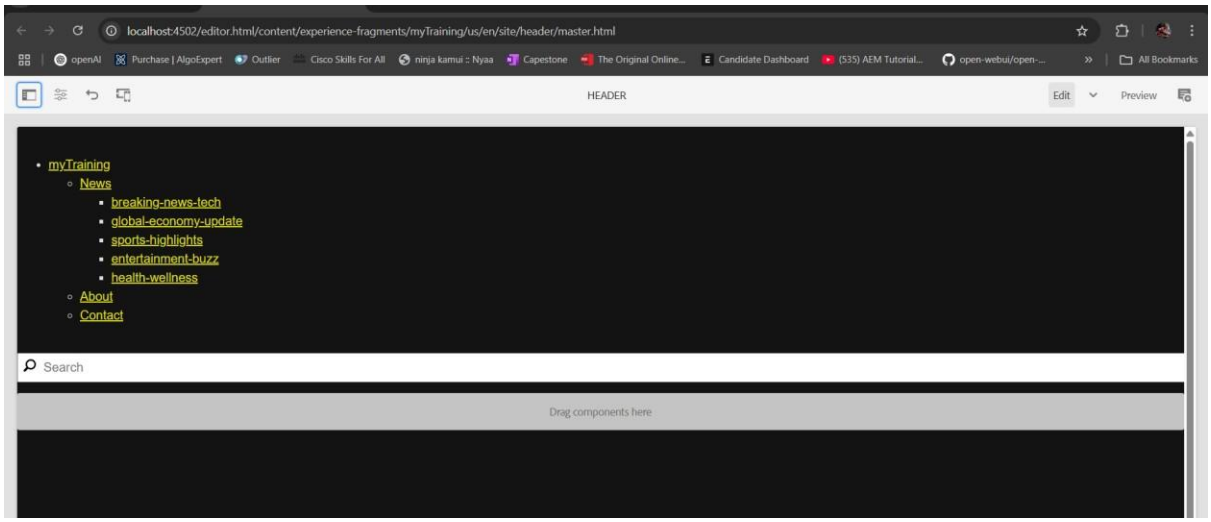
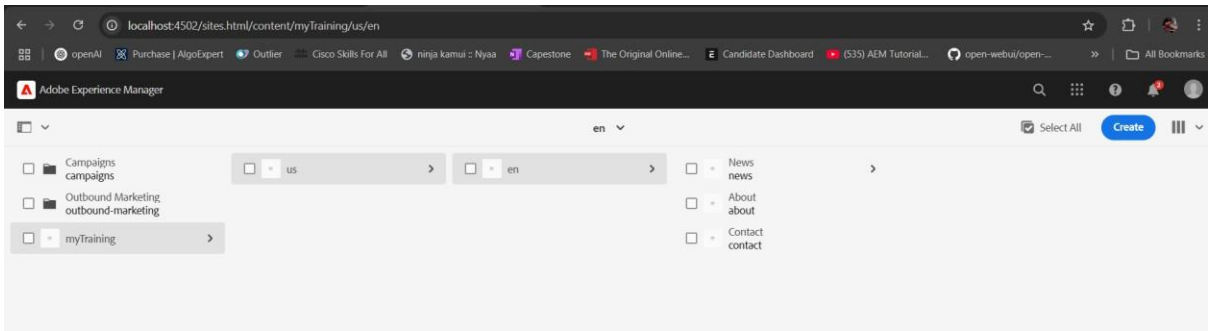
Task 1: Create 5 News Article Pages under /content/us/en/news



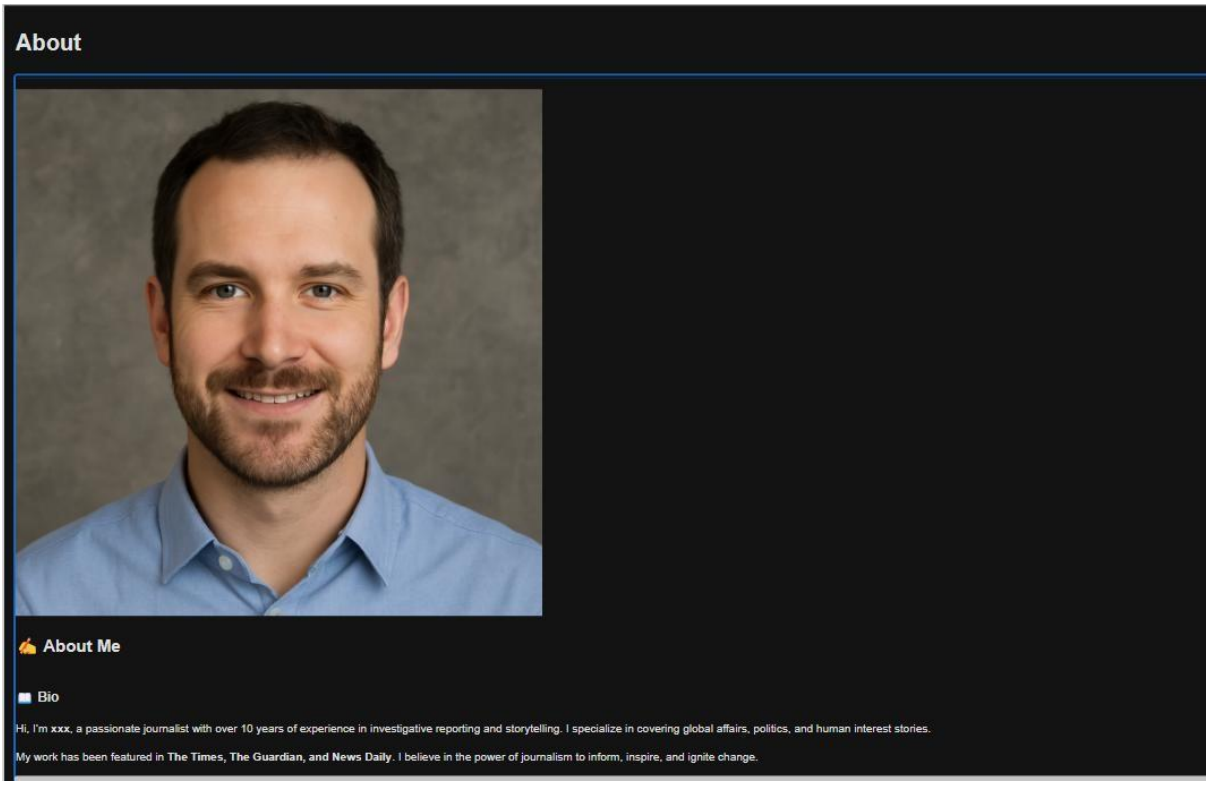
Task 2: Create Header Experience Fragment and Use These Pages as Menu



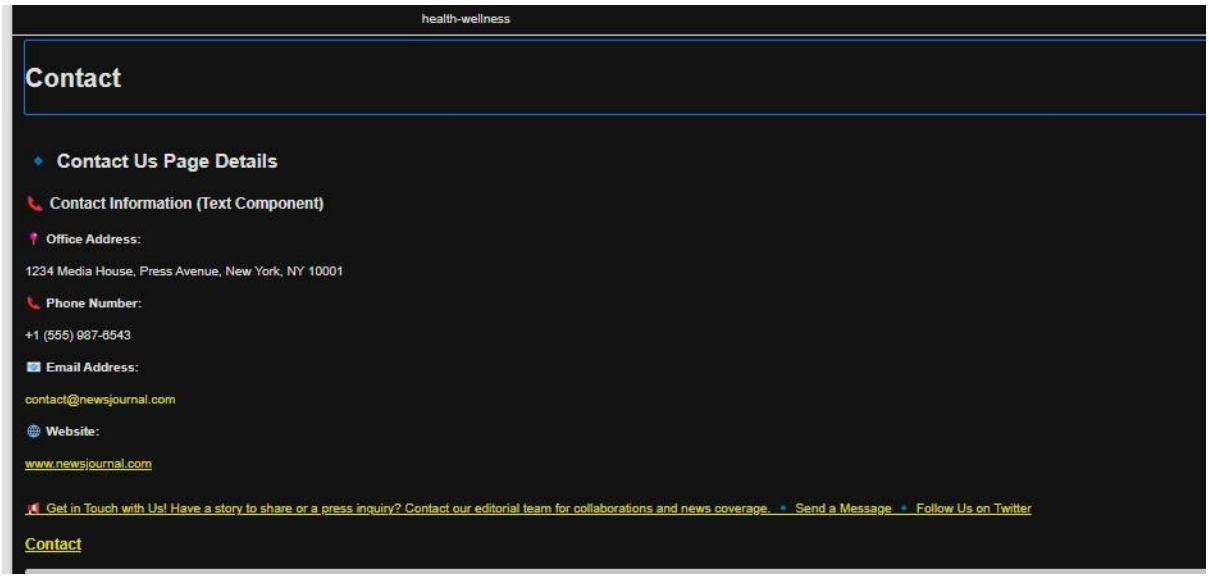
Task 3: Create "Contact Us" and "About Me" Pages



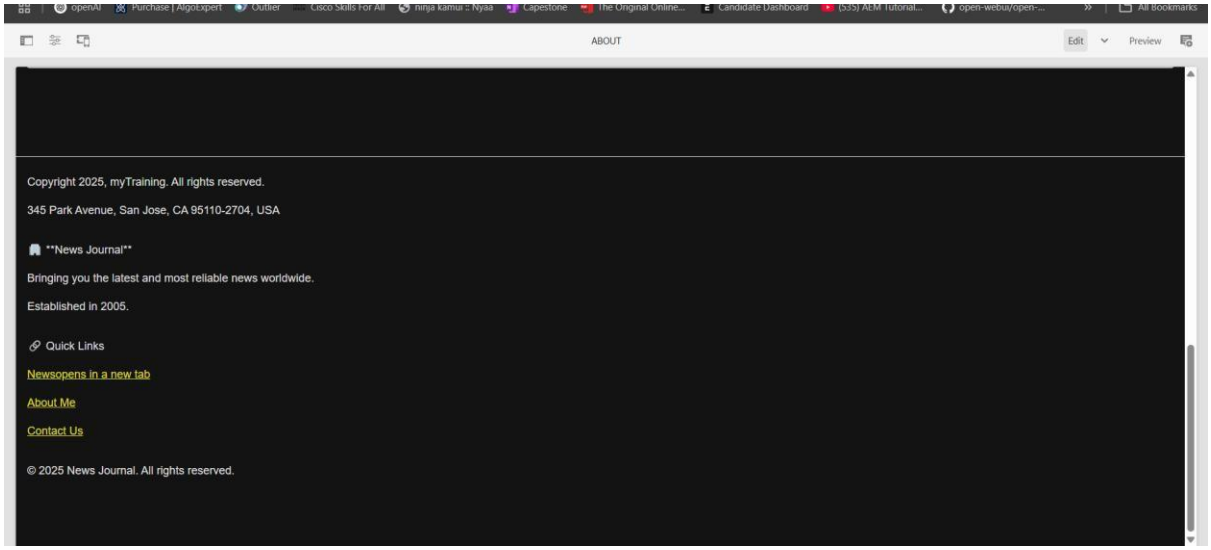
About me page:



Contact page:



Task 4: Create Footer Experience Fragment with 4 Sections



Task 5: Create a Custom Service to Print Hello World and Call this Service from News Component Sling Model

1. Create the Service:

Path: /apps/newsroom/core/services/HelloWorldService.java

```
package com.newsroom.core.services;
public interface HelloWorldService {
String getHelloWorld();
}
```

2. Implement the Service:

Path: /apps/newsroom/core/services/HelloWorldServiceImpl.java

```
package com.newsroom.core.services;
import org.osgi.service.component.annotations.Component;
@Component(service = HelloWorldService.class)
public class HelloWorldServiceImpl implements HelloWorldService {
    @Override
    public String getHelloWorld() {
        return "Hello World from Newsroom Service!";
    }
```

```
}
```

3. Call the Service in News Component Sling Model:

Path: /apps/newsroom/core/models/NewsItemModel.java

```
package com.newsroom.core.models;

import com.newsroom.core.services.HelloWorldService;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.Model;

import javax.inject.Inject;

@Model(adaptables = Resource.class)
public class NewsItemModel {

    @Inject
    private HelloWorldService helloWorldService;

    public void logHelloWorld() {
        System.out.println(helloWorldService.getHelloWorld());
    }
}
```

Task 6. Create a custom service to print hello world and call this service from news component sling model and print this value in logs as well.

1. Create the Service Interface

First, we define the service interface which will provide a method for returning "Hello World".

File: /apps/newsroom/core/services/HelloWorldService.java

```
package com.newsroom.core.services;
public interface HelloWorldService {
String getHelloWorld();
}
```

• Explanation: This interface declares a method getHelloWorld() that returns a String.

2. Implement the Service

Now, we provide the implementation for this service where we will return "Hello World".

File: /apps/newsroom/core/services/HelloWorldServiceImpl.java

```
package com.newsroom.core.services;
import org.osgi.service.component.annotations.Component;
@Component(service = HelloWorldService.class)
public class HelloWorldServiceImpl implements HelloWorldService {
@Override
public String getHelloWorld() {
return "Hello World from Newsroom Service!";
}
}
```

• Explanation:

o We use the @Component annotation to register this class as an OSGi service in AEM.

o This class implements the HelloWorldService interface and provides the implementation for the getHelloWorld() method, which returns the string "Hello World from Newsroom Service!".

3. Call the Service from News Component Sling Model

Now, we modify the News Component Sling Model to inject and call the custom service. We will also log the result of the service using System.out.println().

File: /apps/newsroom/core/models/NewsItemModel.java

```
package com.newsroom.core.models;
import com.newsroom.core.services.HelloWorldService;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.Model;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import javax.inject.Inject;
@Model(adaptables = Resource.class)
public class NewsItemModel {
```

```
private static final Logger LOG = LoggerFactory.getLogger(NewsItemModel.class);
@Inject
private HelloWorldService helloWorldService;
// Method to call the service and log the result
public void logHelloWorld() {
String message = helloWorldService.getHelloWorld();
LOG.info(message); // Logs the message
}
}
```

• Explanation:

o The HelloWorldService is injected into the NewsItemModel using @Inject.

o The logHelloWorld() method calls the getHelloWorld() method of the service and logs the returned string using SLF4J (LOG.info()).

o We use SLF4J (Simple Logging Facade for Java) to log the message in a proper way, which is the standard logging framework in AEM.

The log will appear in the AEM error.log file (or wherever logs are configured for your AEM instance).

4. Call the logHelloWorld() Method in HTL (HTML Template Language)

Finally, you need to call the logHelloWorld() method from the Sling Model in your HTL template (HTML).

File: /apps/newsroom/components/news/news-item.html

```
<data-sly-use.newsModel="com.newsroom.core.models.NewsItemModel" />
<!-- Call the method to log Hello World -->
<sly data-sly-call="${newsModel.logHelloWorld}" />
```

• Explanation:

o The data-sly-use tag is used to create a reference to the NewsItemModel class in the HTL file.

o data-sly-call calls the logHelloWorld() method from the model, which will print "Hello World from Newsroom Service!" in the AEM logs.

5. Check the Logs

Once everything is set up, the output "Hello World from Newsroom Service!" will be logged in AEM’s error.log or request.log files.

You can find these logs in the crx-quickstart/logs/ directory of your AEM instance, specifically

in:
crx-quickstart/logs/error.log

Task 7. Create Custom Configurations for Third-Party API

Steps:
1. Create Configuration Interface:
Path: /apps/newsroom/core/config/ThirdPartyApiConfig.java

```
package com.newsroom.core.config;
public interface ThirdPartyApiConfig {
String getApiUrl();
}
```

2. Create Configuration Implementation:
Path: /apps/newsroom/core/config/ThirdPartyApiConfigImpl.java

```
package com.newsroom.core.config;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.metatype.annotations.Designate;
@Designate(ocd = ThirdPartyApiConfigImpl.class)
@Component(service = ThirdPartyApiConfig.class)
public class ThirdPartyApiConfigImpl implements ThirdPartyApiConfig {
    private String apiUrl;

    @Override
    public String getApiUrl() {
        return apiUrl;
    }

    // Bind method to inject the config
    @Activate
    @Modified
    public void activate(String apiUrl) {
        this.apiUrl = apiUrl;
    }
}
```

3. Call the API and Print in Logs:

Modify the Sling Model to fetch and print the data.

```
package com.newsroom.core.models;

import com.newsroom.core.config.ThirdPartyApiConfig;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.Model;

import javax.inject.Inject;

@Model(adaptables = Resource.class)
public class NewsItemModel {

    @Inject
private ThirdPartyApiConfig thirdPartyApiConfig;
public void fetchApiDataAndLog() {
String apiUrl = thirdPartyApiConfig.getApiUrl();
// Code to call the API and log the response (e.g., using HttpClient or any HTTP library)
System.out.println("Fetching data from API: " + apiUrl);
}
}
```