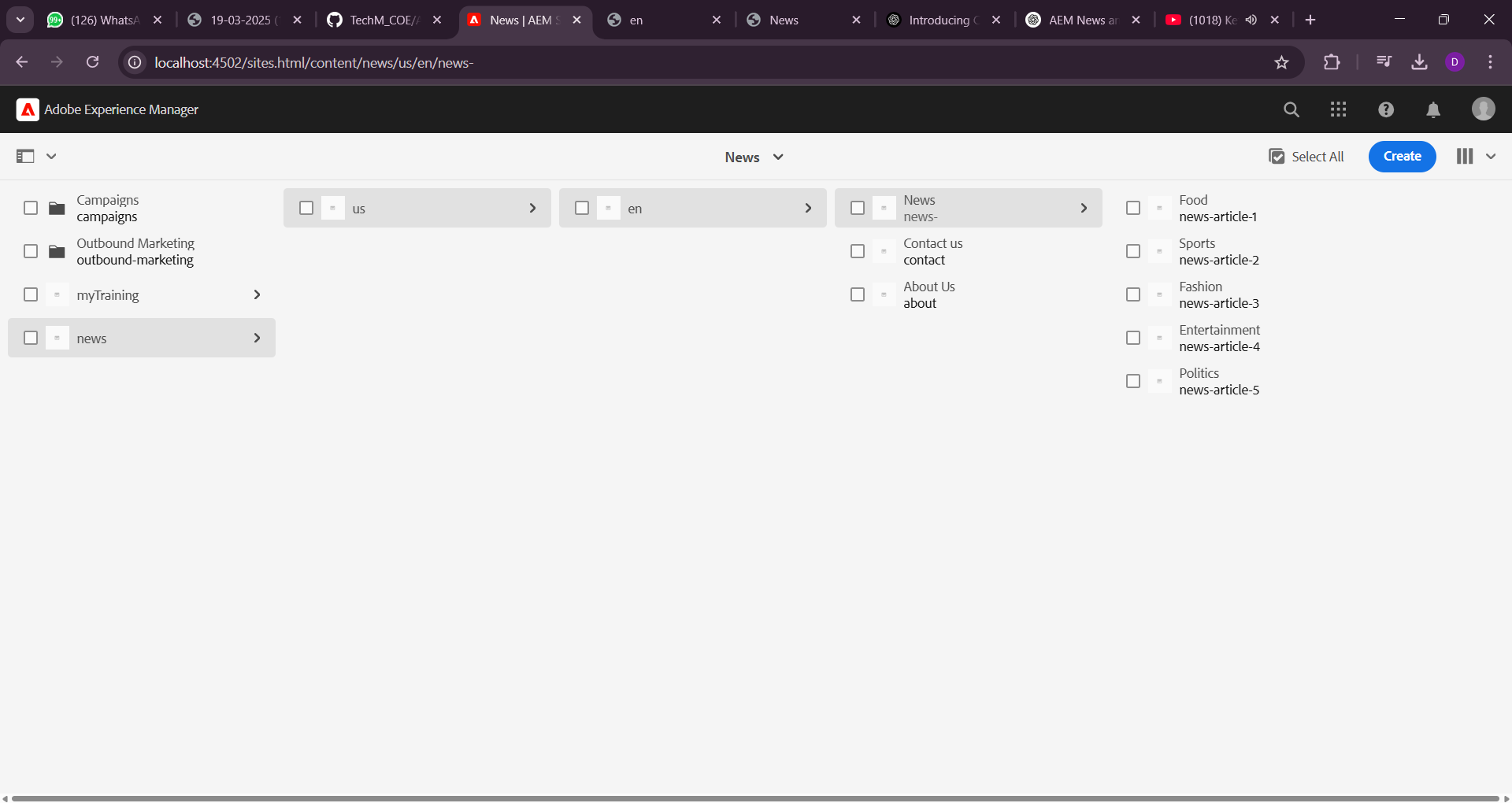
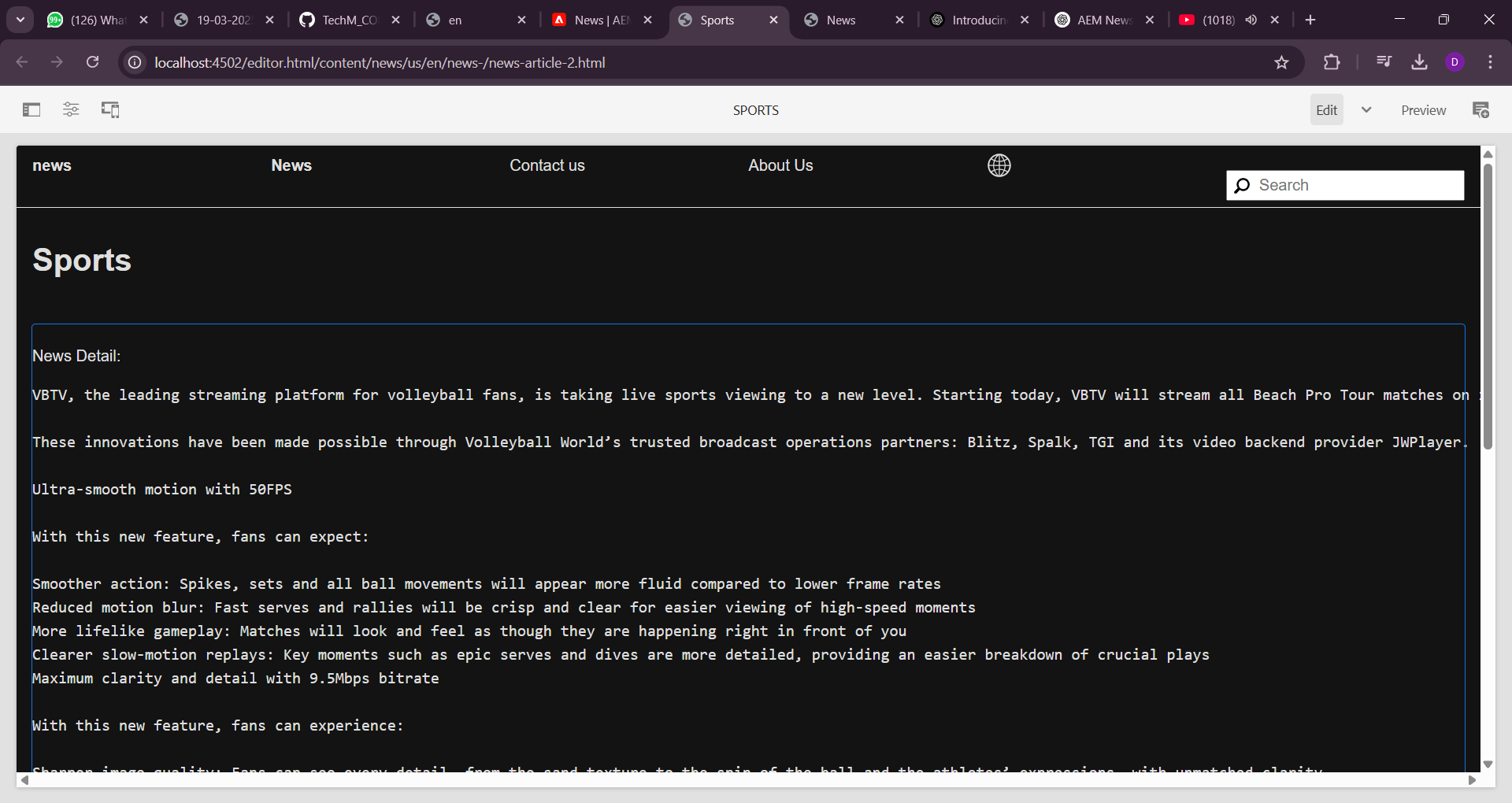
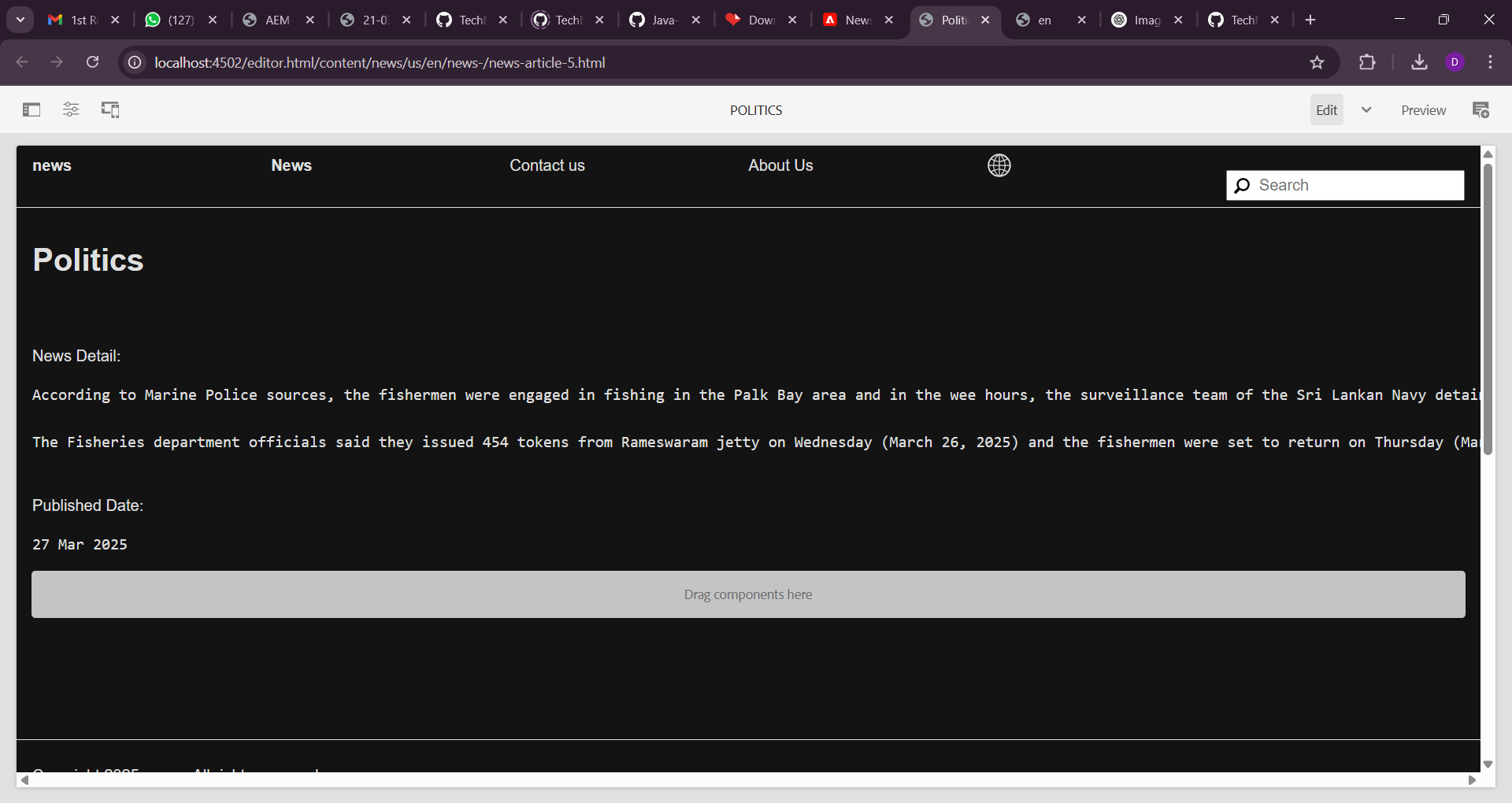
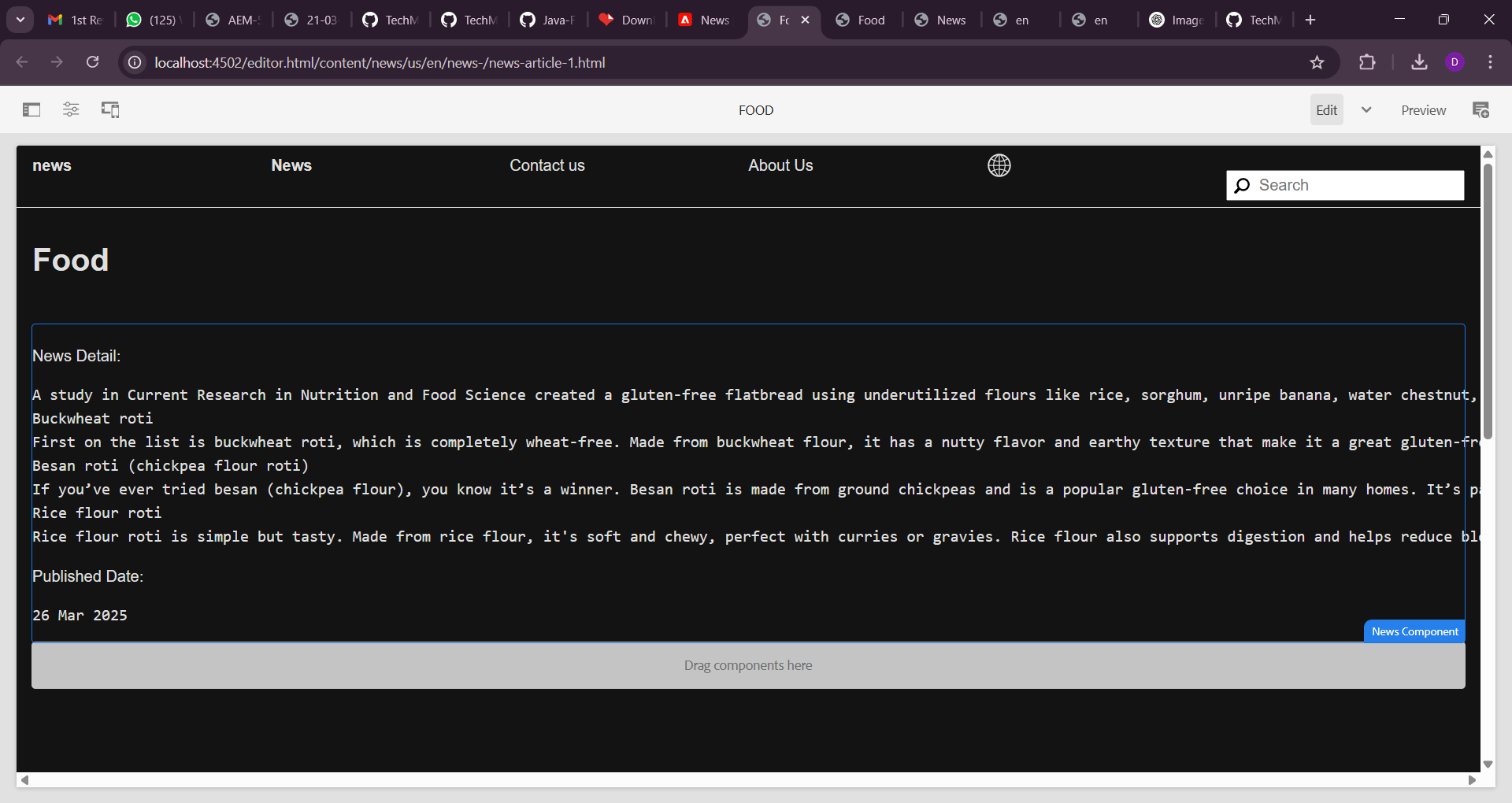
1.Create 5 News Article Pages under /content/us/en/news

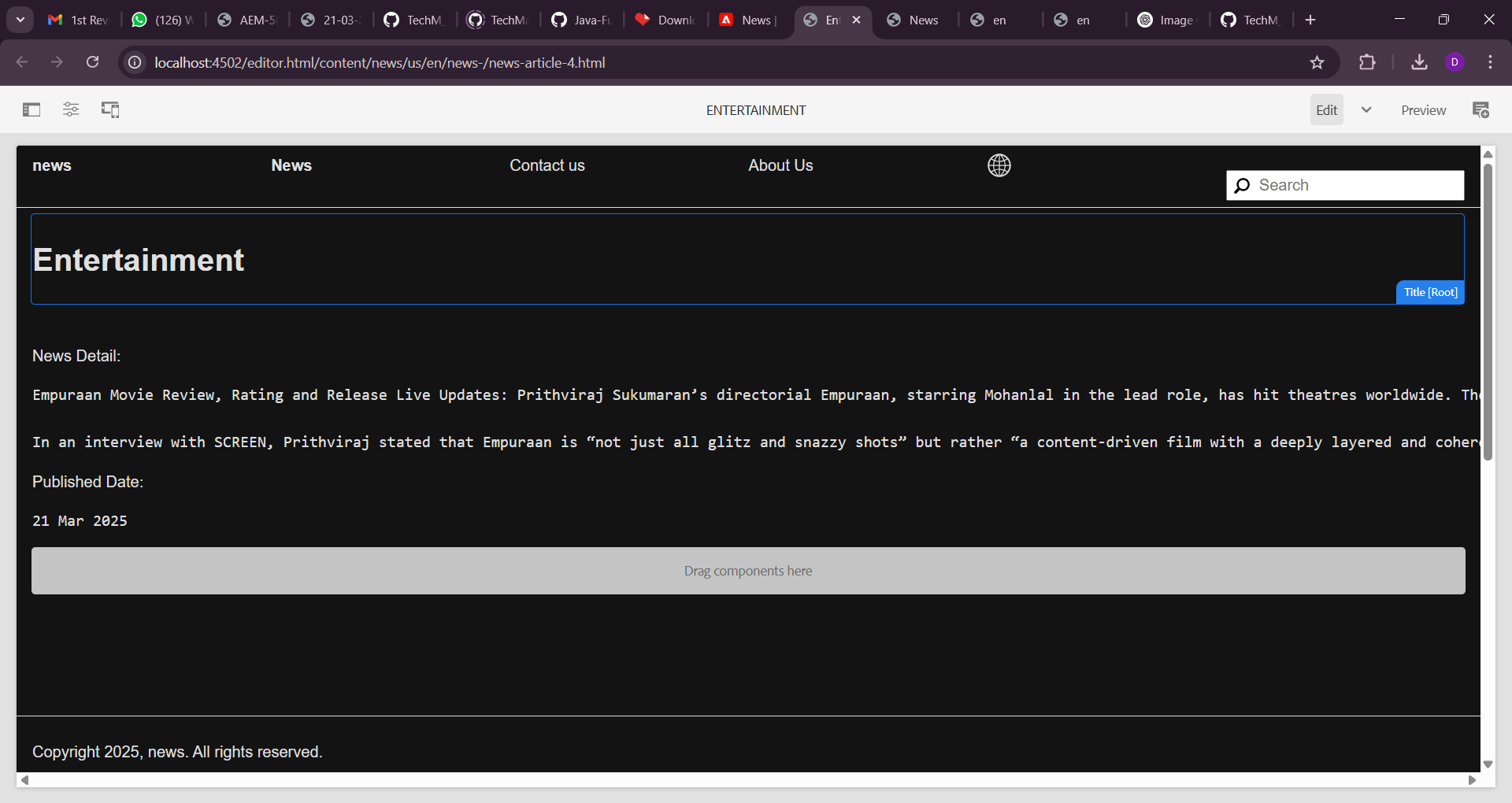
Page for the news article to be displayed, The 5 news article page is displayed below



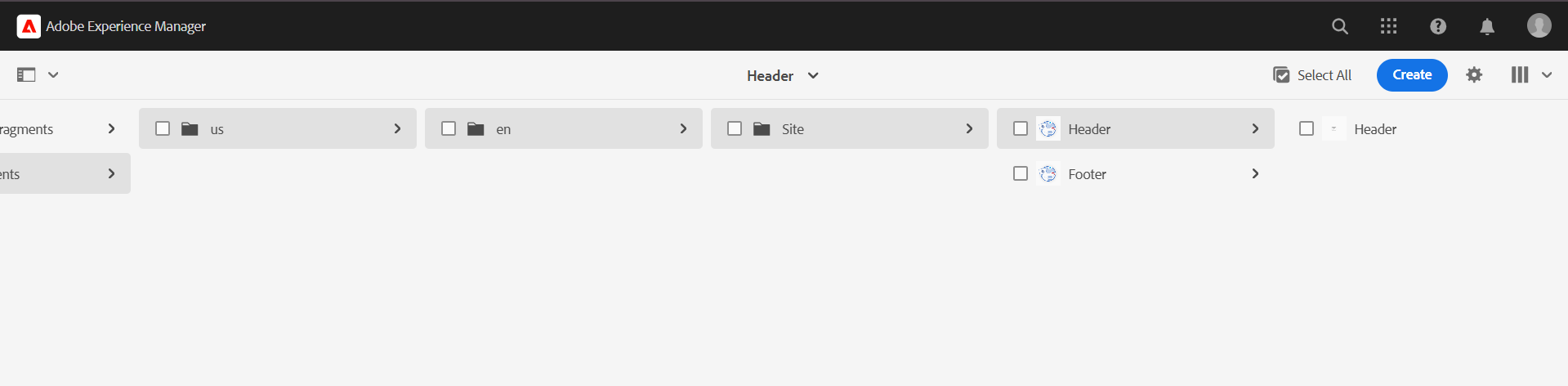


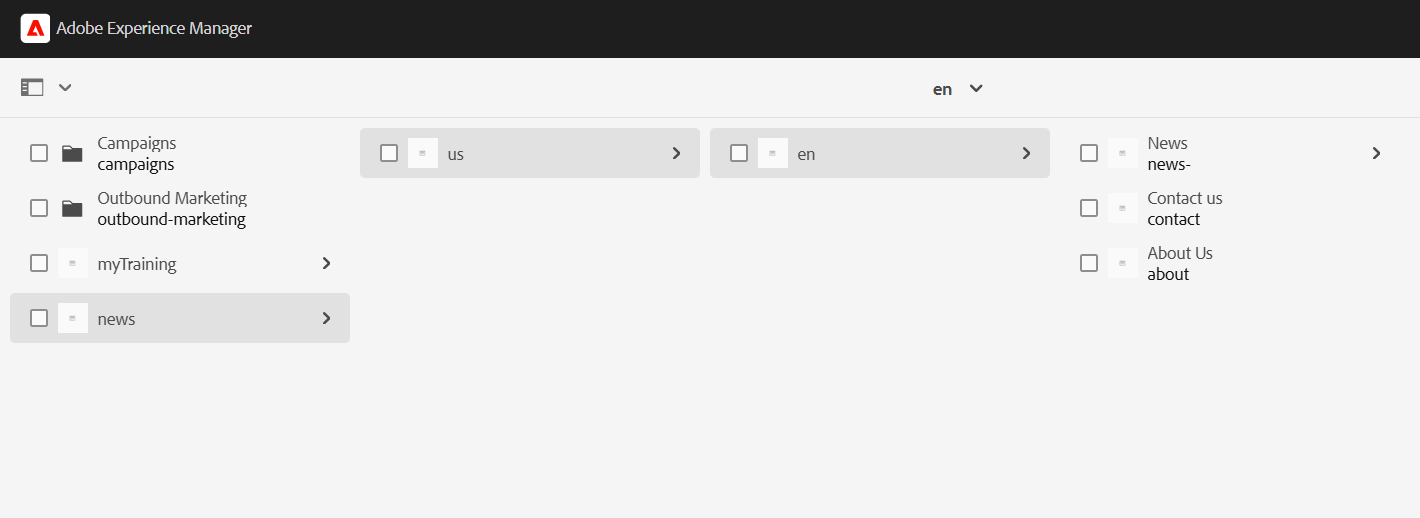




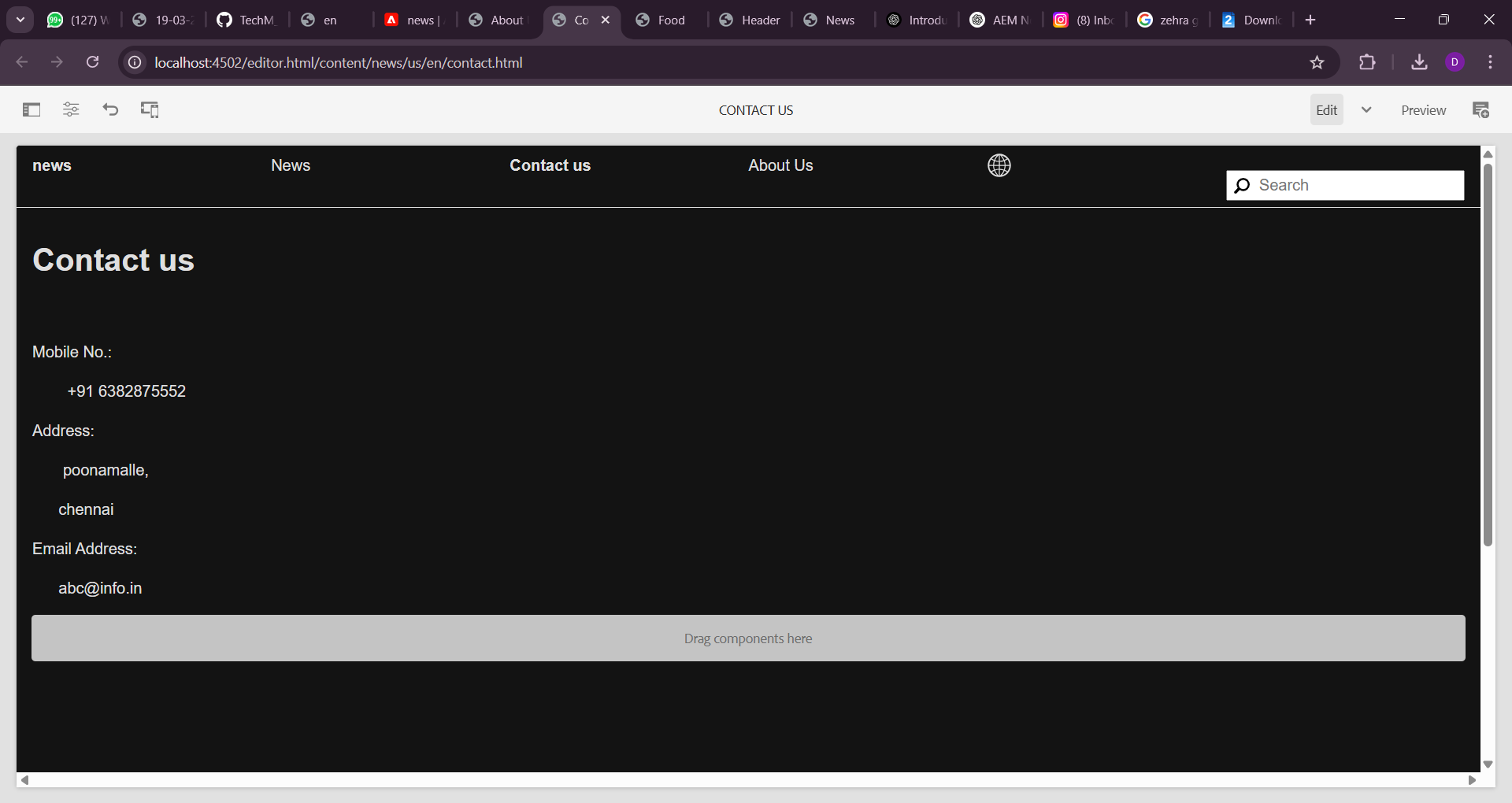


**2. Create Header Experience fragment for header This will include: 1. News Menu – Links to the 5 news article pages you created. 2. Contact Us Page – With office address, email, or mobile number :**

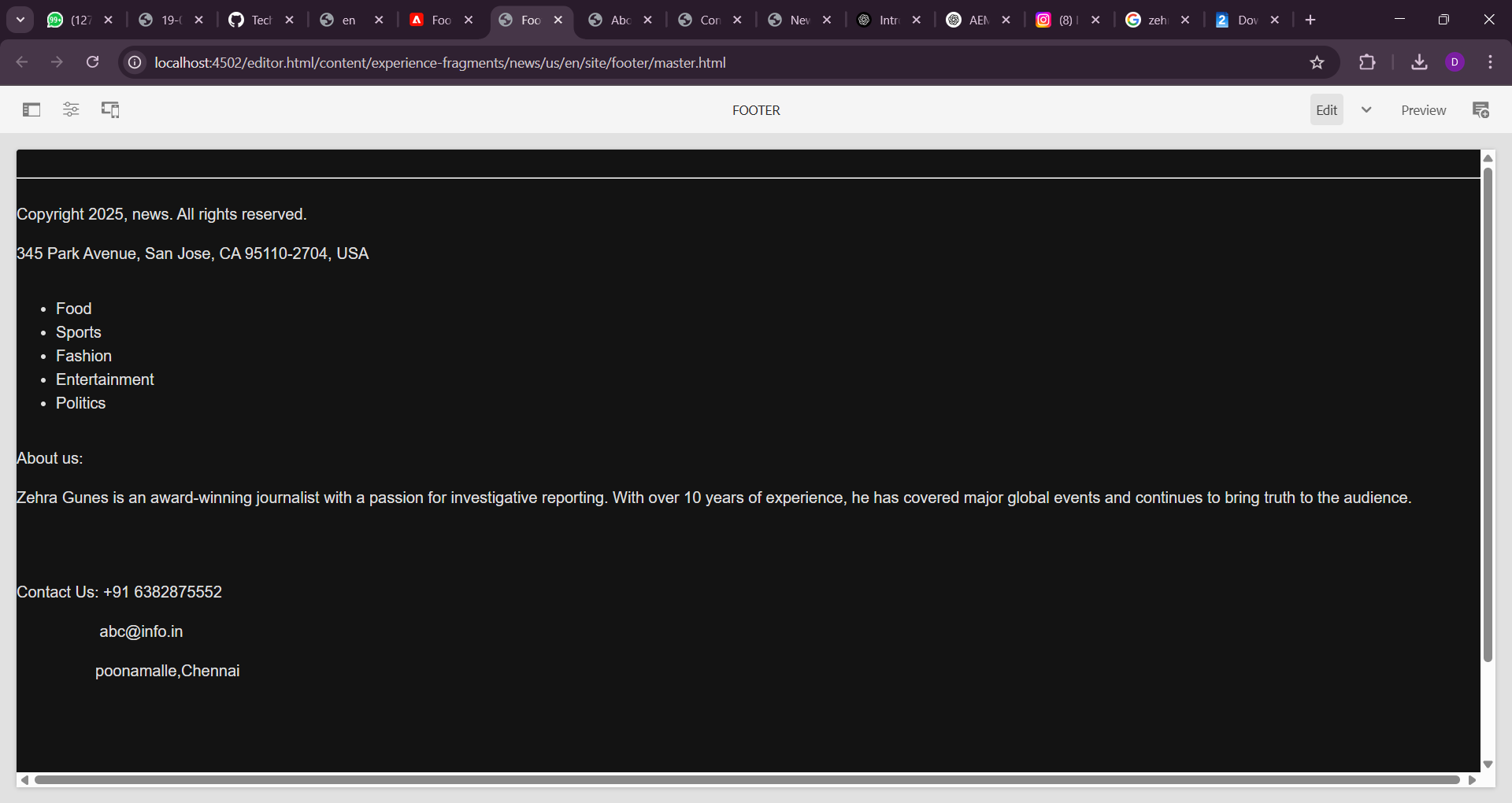
****

****

**4.Contact Us :**

****

**5.Footer :**

****

**6. Create a custom service to print Hello World and call this service from the news component sling model and print this value in logs as well.**

Create a service class:   
Path: /apps/newssite/core/src/main/java/com/newssite/core/services

**HelloWorldService.java**

package com.newssite.core.services;

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Component(service = HelloWorldService.class)

public class HelloWorldService {

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

    public String getMessage() {

        String message = "Hello World from Newsroom Service";

        LOG.info(message);

        return message;

    }

}

Update the NewsModel.java to call the service:

package com.newssite.core.models;

import com.newssite.core.services.HelloWorldService;

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

import org.apache.sling.models.annotations.Model;

import org.apache.sling.models.annotations.injectorspecific.OSGiService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Model(adaptables = Resource.class)

public class NewsModel {

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

    @OSGiService

    private HelloWorldService helloWorldService;

    public String getServiceMessage() {

        String message = helloWorldService.getMessage();

        LOG.info("Service message: {}", message);

        return message;

    }

}

**7. Create custom configurations where I can provide the 3rd party API. For example(https://jsonplaceholder.typicode.com/posts) and I can see the data as json. And this data should be printed in logs.**

Create the configuration:   
Path: /apps/newssite/core/src/main/java/com/newssite/core/config

**NewsAPIConfig.java**

package com.newssite.core.config;

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

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

@ObjectClassDefinition(name = "Newsroom API Configuration")

public @interface NewsAPIConfig {

    @AttributeDefinition(name = "API Endpoint")

    String apiEndpoint() default "https://jsonplaceholder.typicode.com/posts";

}

Create the service to call the API:   
Path: /apps/newssite/core/src/main/java/com/newssite/core/services

**NewsAPIService.java**

package com.newssite.core.services;

import com.newssite.core.config.NewsAPIConfig;

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

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

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

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.util.EntityUtils;

@Component(service = NewsAPIService.class, configurationPolicy = ConfigurationPolicy.REQUIRE)

public class NewsAPIService {

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

    @Reference

    private NewsAPIConfig config;

    @Activate

    @Modified

    protected void activate(NewsAPIConfig config) {

        this.config = config;

    }

    public void fetchAPIData() {

        try (CloseableHttpClient client = HttpClients.createDefault()) {

            HttpGet request = new HttpGet(config.apiEndpoint());

            CloseableHttpResponse response = client.execute(request);

            String jsonResponse = EntityUtils.toString(response.getEntity());

            LOG.info("API Response: {}", jsonResponse);

        } catch (Exception e) {

            LOG.error("Failed to fetch API data", e);

        }

    }