ATYPON

Docker Assignment

Made by: Samer AlHamdan

Instructors:

Motasim Aldiab

Fahed Jubair

1. Introduction

This report offers a comprehensive analysis for the Video Streaming System that has 5 parts that will be explained in detail. during Atypon's Training in May 2023. A comprehensive web site that facilitates the seamless uploading, storage, and streaming of video content. This system encompasses five integral components, each playing a crucial role in ensuring a secure and efficient video management process

2. The Assignment

The assignment offers a selection between two options: a containerized microservices data collection and analytics system, and a containerized Video Streaming System. I have opted for the latter, which involves crafting a cohesive system comprising five interdependent containers.

1. Upload Video (Web App): This component empowers users to effortlessly upload video files in various formats, such as MP4, following a meticulous authentication process via the Authentication Service. Video details, including names and filesystem paths, are meticulously stored in the MySql Service, while the actual video files are meticulously written to a dedicated file storage via the File System Service.
2. Video Streaming (Web App): Designed for seamless video consumption, this module enables authenticated users to access and view videos with ease. Video validation is conducted through the Authentication Service, while information about available videos and their corresponding paths are retrieved from the MySql DB Service. The videos themselves can be smoothly accessed and streamed from the file storage through the File System Service.
3. Authentication Service: A fundamental service dedicated to verifying user credentials, ensuring a secure and controlled access environment throughout the system.
4. File System Service: This service, indispensable to the system's functionality, manages the read and write operations of files to and from the designated file storage mediums, whether it be traditional HDDs or modern cloud storage solutions like AWS S3.
5. MySql DB Service: Central to the system's organization, this service maintains an updated record of available videos, along with their corresponding paths or URLs on the chosen file storage system.

3. Authentication Service

This implements an authentication service with user registration, login, and verification functionalities. It also includes components related to file storage using Amazon S3. The authentication service uses JWT tokens for secure communication and integrates with a user database.

3.1 User

The `User` class models a user entity with fields for a unique identifier (userId), username, password, and name. The class is equipped with Lombok annotations to reduce boilerplate code, JPA annotations to facilitate database mapping, and sequence-based generation for the user ID. The provided constructors and methods enable easy handling of user-related information within the application.

3.2 UseController

The `UserController` class is a Spring REST controller responsible for managing user-related operations such as login, registration, retrieving user details, and verifying user authentication using JWT tokens. It interacts with the `UserService` to handle business logic and communication with the service layer. The class is designed to facilitate secure user authentication and user-related CRUD operations in the application.

3.3 UserService

The `UserService` class serves as the main service layer for user-related operations, authentication, and JWT token generation. It interacts with the `appUserRepo` to handle data retrieval and manipulation, while also providing authentication and token generation functionality. The class is designed to facilitate secure user interactions and authentication within the application, including user creation, login, and information retrieval.

3.4 UserRepo

The `UserRepo` interface extends the `JpaRepository` interface and provides a custom method for querying and managing `User` entities. It acts as a bridge between the application and the underlying data source, allowing developers to perform database operations on `User` entities without writing explicit SQL queries. The custom method `findUserByUsername` retrieves a `User` entity by its username, leveraging Spring Data JPA's automatic query generation.

3.5 LoggedInUser

The `LoggedInUser` class is a simple data class representing a user's credentials during authentication. It uses Lombok annotations to automatically generate constructors, getter and setter methods, `equals`, `hashCode`, and `toString` methods for its fields. This class is likely used to transport user authentication credentials between components of the application, such as between the client and server during login operations.

4. File System Service

This implements a file storage service using Amazon S3. The `StorageConfig` class sets up the Amazon S3 client, the `StorageController` provides REST endpoints for file upload and deletion, and the `StorageService` performs the actual file operations using the Amazon S3 client. This architecture allows for easy integration of Amazon S3 for scalable and reliable file storage.

4.1 StorageConfig

The `StorageConfig` class is a Spring configuration class that provides a bean definition for creating and configuring an Amazon S3 client instance. It reads the necessary access key, secret key, and region values from the application configuration using `@Value` annotations. This configured Amazon S3 client can then be injected into other components of the application, allowing seamless integration with AWS S3 for storing and retrieving files.

4.2 StorageController

The `StorageController` class is a Spring REST controller that handles file upload and deletion operations using endpoints "/file/upload" and "/file/delete/{fileName}". It relies on a `StorageService` to perform these actions. The class is designed to facilitate file-related interactions between clients and the application, allowing files to be uploaded and deleted from the system.

4.3 StorageService

The `StorageService` class is a Spring service that facilitates file upload and deletion to/from an Amazon S3 bucket. It uses the Amazon S3 client to interact with the S3 storage service. The class provides methods for converting `MultipartFile` to `File`, uploading files, generating URLs, and deleting files. It's designed to handle file storage operations in a cloud-based environment.

5. Upload Service

This supports video uploading, storage, and retrieval, with features such as view counting and user-specific video lists. It employs Spring Boot, Lombok, and RESTful APIs to build a functional video management system.

5.1 Video

The `Video` class models a video entity with fields for an identifier (`id`), links to the video and its thumbnail, name, views count, author, and associated user ID. It's designed to be used in an application where users can upload, share, and view videos. The provided annotations help with database mapping and boilerplate code reduction, making it easier to manage video-related information within the application.

5.2 VideoController

The `VideoController` class is a Spring REST controller that handles video-related operations such as uploading, retrieving, and managing videos. It communicates with the `videoService` to perform business logic and interact with the underlying data. This class is designed to facilitate user interactions with video content in the application, allowing videos to be uploaded, viewed, and managed.

5.3 VideoRepo

The `VideoRepo` interface extends `JpaRepository` and provides methods for performing standard CRUD operations on `Video` entities. It also includes custom query methods for specific operations like updating view counts and retrieving videos by user ID. This interface facilitates interaction with the underlying database using Spring Data JPA's automatic query generation and execution capabilities.

5.4 VideoService

The `VideoService` class is a Spring service that encapsulates various video-related operations. It interacts with the `videoRepo` to perform CRUD operations on `Video` entities. Additionally, it communicates with authentication and file upload services to retrieve user information and upload files. This class is designed to handle the business logic related to video management and interaction in the application, providing a centralized point for video-related operations.

6. Streaming Service

features a home page where users can explore a curated list of videos. Upon selecting a video, users are seamlessly taken to the dedicated video streaming page. This immersive video streaming page not only plays the chosen video but also intelligently suggests related videos for continuous entertainment. Facilitating seamless navigation, a system header empowers users to effortlessly switch between the dynamic home page and their personalized profile. This fluid user experience is made possible by Vuex, the state management library, ensuring efficient management of video data and state consistency. Additionally, Vue Router orchestrates the seamless flow between distinct views, harmonizing the user's journey throughout the application.

6.1 HomePage

This Vue.js single-file component defines the user interface for displaying a list of videos. It dynamically renders each video using the `<VideoDisplay>` component and provides interactivity to navigate to the video streaming page when a video is clicked. The component uses Vuex for state management, interacts with routing, and applies styles to create a visually appealing video listing layout. It's part of a larger Vue.js application dedicated to video sharing or streaming functionality.

6.2 PlayVideo

This Vue.js single-file component is responsible for rendering the user interface for viewing individual videos. It displays the selected video's details, related videos, and provides navigation using the "Back" button. The component uses Vuex for state management and interacts with routing to handle navigation. Additionally, it utilizes CSS styling to achieve a visually appealing and responsive layout. It's a part of a larger Vue.js application dedicated to video sharing or streaming functionality.

6.3 UploadHeader

This Vue.js single-file component defines a system header with navigation links. It provides an intuitive user interface for accessing the home page and user profile page. The component uses Vue's templating, styling, and methods to create a responsive header that enhances user navigation. The component is part of a larger Vue.js application and contributes to the overall user experience of the system.

6.4 VideoDisplay

This Vue.js single-file component provides a structured and styled representation of a video item. It displays the video's thumbnail, title, creator name, and view count. The component is designed to be reusable within a list or grid layout of videos. It contributes to the overall user interface of a video streaming or sharing application by offering an appealing and consistent way to showcase videos.

6.5 indexStore

This Vuex store module effectively manages the state of videos in a video streaming or sharing application. It provides a way to retrieve videos, update their view counts, and store the video data in the state. The module follows the Vuex pattern of separation between state, getters, mutations, and actions, which helps maintain a clear and organized state management structure.

6.6 indexRouter

This Vue Router configuration module sets up the navigation routes for a web application. It defines routes for the home page and video streaming page. The module uses the `createRouter` function to create a router instance and `createWebHistory` to configure the router's history mode. The routes are associated with specific components, enabling smooth navigation between different views within the application.

7. Upload Web

This is a video streaming system that allows users to register, log in, upload videos, and view their uploaded videos. The Vue Router controls navigation between different views while ensuring proper access based on user authentication status. The Vuex store manages user authentication and data persistence. The various components provide the user interface for different functionalities of the application.

7.1 LoginPage

This Vue.js component represents a login page for an application. It provides a form for users to enter their email and password, along with validation checks and error messages. The component interacts with Vuex actions for authentication and Vue Router for navigation. It uses scoped CSS for styling, ensuring that styles are isolated to this component.

7.2 RegisterPage

This Vue.js component represents a user registration page for an application. It provides a form for users to enter their first name, last name, email, and password, along with validation checks and error messages. The component interacts with Vuex actions for user registration and Vue Router for navigation. It uses scoped CSS for styling, ensuring that styles are isolated to this component.

7.3 UserVideo

This Vue.js component represents a video management page where users can view and manage their uploaded videos. It provides buttons for video uploading, navigating to the homepage, and video streaming. The component utilizes a modal for video uploading with file upload functionality powered by `vue-filepond`. It interacts with Vuex for user-related data and Vue Router for navigation. Scoped CSS ensures that styles are isolated to this component.

7.4 UploadHeader

This Vue.js component represents a header section for a streaming system's user interface. It includes a title button that leads to the homepage and a "Log Out" button. The component uses scoped CSS for styling, ensuring that the styles are isolated to this component. The methods in the script section handle navigation and user logout. It communicates with Vuex to perform the logout action. Overall, this component provides users with navigation options and a way to log out of the system.

7.5 VideoDisplay

This Vue.js component represents a video display card that showcases a video's thumbnail, title, and author information. The component's template and styles ensure that the card's elements are visually appealing and responsive. The `props` property allows the component to receive video data from a parent component, making it versatile for displaying various videos. This card design could be used in a list or grid layout to present multiple videos in a user-friendly manner.

7.6 indexStore

This Vuex store module manages user authentication, user data, and persistent state across page reloads. It provides actions for user registration, login, and logout, along with getters for accessing various pieces of state. The `createPersistedState` plugin ensures that selected state properties are stored securely in local storage for a seamless user experience even after page reloads. This store module is designed to integrate with a Vue.js application to manage authentication-related data and state changes.

7.7 indexRouter

This code configures Vue Router for the application with routes defined for `'/login'`, and `'/register`. It also implements a navigation guard that controls access to routes based on user authentication status. If a user is not authenticated, they are redirected to the login page when trying to access other routes. If a user is authenticated, they are redirected to the home page when trying to access the login or register pages. This ensures proper navigation behavior based on authentication state.