**The Islamia University of Bahawalpur**

**Department of Computer Science**

****

**SOFTWARE DESIGN DESCRIPTION**

**(SDD DOCUMENT)**

**for**

**Streamify(Video Streaming Web App)**

##### Version 1.0

***By***

**Umar Ahmad**

**Sp20m2ba008**

**Spring 2020-24**

***Supervisor***

**Mam Saman Nawaz**

***Bachelor of Science in Computer Science***

Table of Contents

[**Introduction**](#_Toc121305837) **3**

[**Design methodology and software process model**](#_Toc121305838) **3**

[**System overview 3**](#_Toc121305839)

[**Design models [along with descriptions] 5**](#_Toc121305840)

[**Data design 9**](#_Toc121305841)

[**Human interface design 10**](#_Toc121305844)

[**Software requirements traceability matrix 11**](#_Toc121305843)

**Supervised by**

**<Supervisor’s Name>**

Signature

### Introduction

STREAMIFY is a web application designed for streaming videos on subscription where the users around the world can create account on the site that allow them to watch the videos by subscribing on per month subscription.

The SDD for the streamify includes information on the system architecture, design principles, software requirements, data models, and user interface design.

Overall, the SDD for the Streamify provides a detailed and comprehensive overview of the system design and specifications, helping developers and stakeholders understand the system's functionality and ensuring that the system is developed according to the project requirements.

### Design methodology and software process model

The OOP methodology focuses on objects and their interactions to solve complex problems, making it an ideal choice for developing the Streamify. The system involves multiple objects, such as users, applications, and authentication, subscriptions which need to interact with each other to facilitate the process.

By following the OOP methodology, the system can be designed in a modular and scalable way. Each object can be defined and implemented separately, which makes it easier to maintain, test and update the system. The use of OOP also offers several benefits, such as encapsulation, inheritance, and polymorphism, which make software development more efficient and flexible.

Furthermore, I recommend following an Agile software development process model for the Streamify application. The Agile process model is well suited for software projects with changing requirements and a high level of collaboration between developers and stakeholders. This process model emphasizes flexibility, iterative development, and continuous improvement, which is crucial for the successful development of a complex system like Streamify. It allows for frequent feedback and adjustments throughout the development process, which ultimately leads to a better end product.

### System overview

Streamify is a video streaming web application. The aim of this application is to provide video content of different categories like Music, Movies, Education and others. The web application includes user authentication, video browsing, subscription management, and payment management.

The Streamify application consist of several modules, including Signup, Login, payment and subscriptions. The application will allow user to register, subscribe, and stream the videos.

##### **Architectural design**

I have developed a modular program structure that is based on the Object-Oriented Programming (OOP) methodology. The system is composed of several modules that work together to achieve the complete functionality of the system.

The major subsystems of the application are as follows:

1. User Management Module:

* User Registration: Handles user sign-up and registration processes.
* User Authentication: Manages user login, session management, and authentication.
* User Profiles: Stores and manages user profile information, and settings.

1. Subscription and Payment Module:

* Subscription Management: Handles user subscriptions, including plan selection, payment processing, and subscription status.
* Payment Gateway Integration: Integrates with a payment gateway to securely process subscription payment.

1. Content Management Module:

* Content Storage: Stores and manages the video content, include videos, metadata, thumbnails.
* Content Delivery: Distributes the video content efficiently to users.

##### **Process flow/Representation**

##### 

The Process Flow diagram of Streamify represents the series of steps involved in the streaming process from the perspective of the various entities involved, i.e., User, Login/Signup, and content delivery system. The process starts with the user logging in to the application and subscription, along with the content delivery.

### Design models

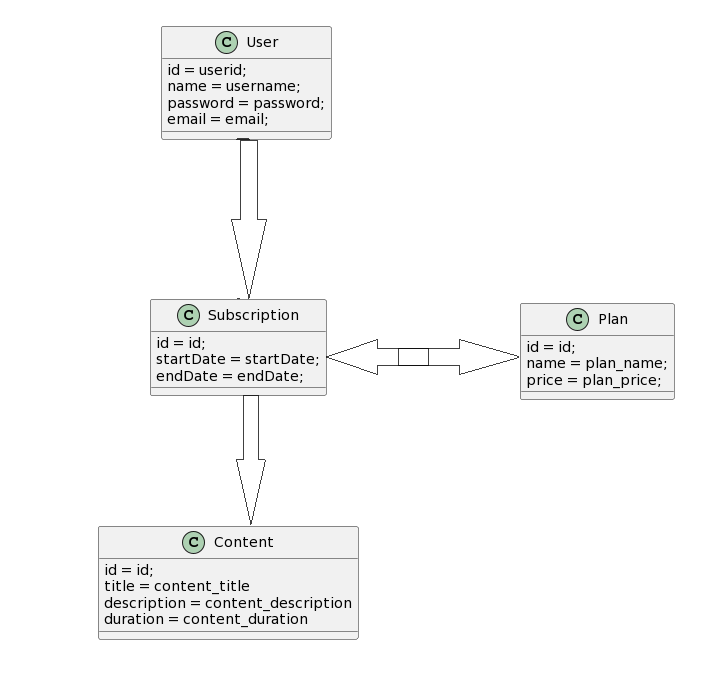
A design model for Streamify can be developed using the Unified Modelling Language (UML), which is a standardized visual language for creating models of software systems. The design model for Streamify should focus on the system's structure, behavior, and interaction between the components of the system.

Overall, the design model for application should provide a clear and comprehensive view of the system's structure, behavior, and interactions, allowing for effective communication between the stakeholders and facilitating the development and testing of the system.

The applicable models may include:

* Class Diagram
* Sequence Diagram
* State Transition Diagram
* Data Flow Diagram

1. **Class Diagram**



In the above example, we have four main classes: User, Content, Subscription, and Plan.

The User class represents a user of the application and includes properties such as id, username, password, email, and a list of subscriptions. It also has methods for login, logout, browsing content, and playing content.

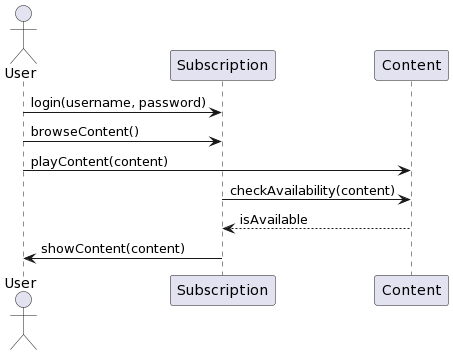
The Content class represents a video content item available for streaming. It includes properties like id, title, description, duration, video URL, thumbnail URL, subtitles, and genres. It provides methods for retrieving and managing the content details.

The Subscription class represents a user`s subscription to a particular plan. It has properties like id, user, plan, start date, and end date. It also includes methods for retrieving the plan, checking if the subscription is active, and extending the subscription.

The Plan class represents a subscription plan with properties like id, name, price, and features. It provides methods for retrieving and setting the plan details.

This is a simplified version of Streamify, the advanced version would likely have more classes and relationships between them. The aims to illustrate the basic structure and relationships of the main classes involved.

1. **Sequence Diagram**

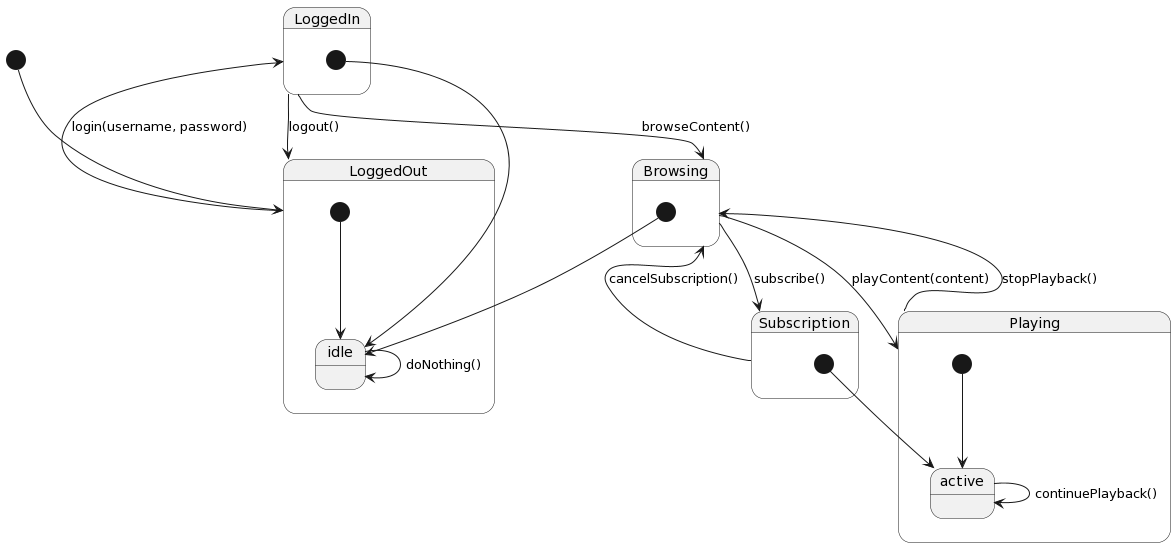


In this sequence diagram:

* The User actor initiates the interaction by logging in with a username and password.
* The User then browses the available content and selects a specific content item to play.
* The User's action triggers a request to the Subscription object to check the availability of the selected content.
* The Content object responds with the availability status to the Subscription.
* Finally, the Subscription informs the User about the availability of the content and shows it to the User.

The sequence diagram aims to illustrate the basic flow of events and communication between the User, Subscription, and Content classes.

**State Transition Diagram**

****

In this state transition diagram:

The User starts in the "LoggedOut" state and can transition to the "LoggedIn" state by successfully logging in with a username and password.

Once logged in, the User can transition to the "Browsing" state to browse the available content.

From the "Browsing" state, the User can transition to the "Playing" state by selecting and playing a specific content item.

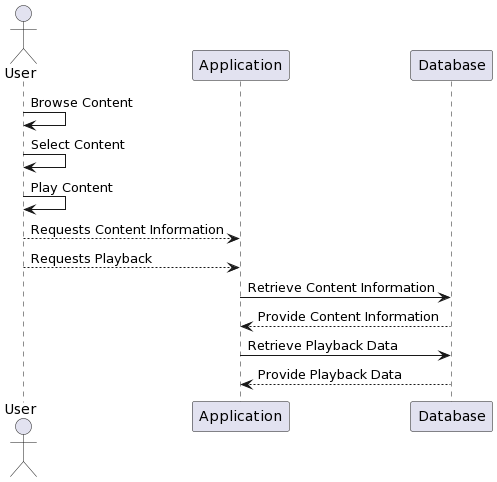
While in the "Playing" state, the User can transition back to the "Browsing" state by stopping the playback.

The User can also transition from the "LoggedIn" state to the "Subscription" state by subscribing to a subscription plan.

From the "Subscription" state, the User can transition back to the "Browsing" state by canceling the subscription.

The states "idle" and "active" represent internal states within the respective higher-level states. They indicate ongoing activity within the specific state, such as performing actions while browsing or continuing playback while in the "Playing" state.

**Data Flow Diagram**

****

In this DFD:

* The User is the primary actor interacting with the system.
* The User can browse content, select a specific content item, and initiate playback.
* The User sends requests for content information and playback data to the Application component.
* The Application component interacts with the Database to retrieve the requested information.
* The Database provides the requested content information and playback data back to the Application, which then forwards it to the User.

This DFD represents a basic interaction flow between the User, Application, and Database components in the video streaming application. The Database component is responsible for storing and retrieving data related to content information and playback data. Please note that this is a simplified example, and in a real-world scenario, there may be additional processes, data stores, and data flows involved in the system. The DFD can be expanded and refined to capture the complete data flow and interactions within the application.

### Data design

In Streamify, the information domain consists of various entities such as users, content, subscriptions, and plans. These entities and their attributes need to be transformed into data structures that can be stored, processed, and organized within the system. Here's an overview of how the major data or system entities can be handled:

**Data Description**

1. Users:

User information, including attributes like User\_ID, Username, Password, and Email, can be stored in a user database table.

The data structure for storing user information can be a collection of user objects or rows in the database table.

User-related operations, such as login, logout, and user management, can be implemented using CRUD (Create, Read, Update, Delete) operations on the user data.

1. Content:

Content information, including attributes like Content\_ID, Title, Description, and Duration, can be stored in a content database table.

The data structure for storing content information can be a collection of content objects or rows in the database table.

Content-related operations, such as browsing, searching, and retrieving content details, can involve querying and retrieving data from the content database.

1. Subscriptions:

Subscription information, including attributes like Subscription\_ID, StartDate, and EndDate, can be stored in a subscriptions database table.

The data structure for storing subscription information can be a collection of subscription objects or rows in the database table.

Subscription-related operations, such as creating a new subscription, managing subscription status, and checking subscription validity, can involve interacting with the subscription data.

1. Plans:

Plan information, including attributes like Plan\_ID, Name, and Price, can be stored in a plans database table.

The data structure for storing plan information can be a collection of plan objects or rows in the database table.

Plan-related operations, such as retrieving available plans, updating plan details, and associating plans with subscriptions, can involve working with the plan data.

The major data or system entities can be stored and organized using a relational database management system (RDBMS) where tables represent the entities and their attributes. The data structures can be implemented as database tables, and the relationships between entities can be established using primary and foreign keys.

To process and manipulate the data, the system can use database queries (e.g., SQL queries) to retrieve, update, and delete data from the tables. Additionally, application code can be developed to handle business logic and perform operations on the data structures, such as validating user input, calculating subscription durations, and managing content availability.

##### **Data dictionary**

A data dictionary is a documentation tool that provides a detailed description of the data elements used within a system. It serves as a reference guide for understanding the structure, meaning, and usage of the data.

**User:**

|  |  |
| --- | --- |
| **Field** | **Description** |
| User\_ID | Unique identifier for each user |
| Username | User’s username |
| Password | User’s password |
| Email | User’s email address |

**Content Table:**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Content\_ID | Unique identifier for content |
| Title | Title of the content |
| Description | Description of the content |
| Duration | Duration of the content |

**Subscription Table:**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Subscription\_ID | Unique identifier for subscription |
| StartDate | Start date of the subscription |
| EndDate | End date of the subscription |

**Plan Table:**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Plan\_ID | Unique identifier for a plan |
| Name | Name of the plan |
| Price | Price of the plan |

**Human Interface Design**

Human interface design, also known as user interface (UI) design, focuses on creating visually appealing and intuitive interfaces for users to interact with a web application. In the context of a video subscription web application, here are some elements and screens that users may encounter:

1. Landing Page:

Users are greeted with an attractive landing page that showcases popular content, featured movies or series, and promotional offers.

The page may include a search bar, navigation menu, and call-to-action buttons for signing up or logging in.

1. Registration/Sign-up:

New users are presented with a registration or sign-up form where they can create an account by providing their name, email address, and password.

The form may also include additional fields like username, date of birth, and payment information.

1. Login:

Returning users can log in using their registered email address and password.

Alternatively, the application may offer social media login options (e.g., Sign in with Google or Facebook) for added convenience.

1. Content Discovery:

Once logged in, users are presented with a content discovery page where they can browse through a variety of movies, TV shows, or other types of media available for streaming.

Content can be categorized based on genres, popularity, recommendations, or personalized suggestions based on the user's viewing history.

1. Content Details:

When a user clicks on a specific content item, they are taken to a detailed page that provides more information about the content, including a description, rating, cast, and reviews.

Users can also see related content, trailers, and options to add the item to their watchlist or start watching it immediately.

1. Playback Interface:

The playback interface allows users to control the video playback, including options for play/pause, volume control, fullscreen mode, and seeking to different parts of the video.

Additional features may include subtitle options, audio language selection, and video quality settings.

Subscription Management:

Users can access their subscription details and manage their subscription plans, including upgrading, downgrading, or canceling subscriptions.

The interface may also provide information about the user's billing history, upcoming payments, and the ability to update payment methods.

Account Settings:

Users can access and modify their account settings, such as updating personal information, changing passwords, and managing email preferences.

They may also have the option to customize their profile, set preferences for content recommendations, and connect or disconnect social media accounts.

The above elements and screens represent a general outline of what users may see and interact with while using a video subscription web application. The actual design and layout of these elements can vary based on the specific branding, visual style, and user experience considerations of the application. It is crucial to prioritize simplicity, ease of use, and responsiveness to ensure a seamless and enjoyable user experience.

### Software requirements traceability matrix

A Software Requirements Traceability Matrix (RTM) is a document that helps track and ensure the fulfillment of requirements of the application throughout the software development lifecycle. It establishes a traceable link between the requirements and the various stages of the development process, including design, implementation, testing, and deployment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement** | **Design** | **Implementation** | **Testing** |
| REQ001 | User Login | Login UI Design | User Authentication | Login functionality testing |
| REQ002 | Content Browsing | Browse UI Design | Content Retrieval and Display | Content Browsing testing |
| REQ003 | Subscription Management | Subscription UI Design | Subscription Process | Subscription management testing |
|  |  |  |  |  |