CompareStream

Architecture/Design Document

**Table of Contents**

[1 Introduction 3](#_Toc131876418)

[2 Design Goals 4](#_Toc131876419)

[3 System Behavior 4](#_Toc131876420)

[4 Logical View](#_Toc131876421) 4

[4.1 High-Level Design (Architecture) 5](#_Toc131876422)

[4.2 Mid-Level Design](#_Toc131876424) 5

[4.3 Detailed Class Design](#_Toc131876425) 6

[5 Process View](#_Toc131876426) 6

[7 Physical View](#_Toc131876428) 7

[8 Use Case View](#_Toc131876429) 7

Change History

**Version:** <x.y>

**Modifier:** <Name>

**Date:** mm/dd/yyyy

**Description of Change:** <What was modified/added?>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** <x.y>

**Modifier:** <Name>

**Date:** mm/dd/yyyy

**Description of Change:** <What was modified/added?>

# Introduction

This document descries the architecture and design for the CompareStream application being developed for UMKC in the CS451R course.

The purpose of this document is to describe the architecture and design of the CompareStream application. It addresses the interests and concerns of major stakeholders.

Major stakeholders:

* Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
* Developers – they want an architecture that will minimize complexity and development effort.
* Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise.

The architecture of the CompareStream application is described from four different perspectives.

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

# Design Goals

There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The design priorities for the CompareStream application are:

* The design should minimize complexity and development effort.
* The design should encourage reusability. We should be able to reuse forms or other structures that are similar across pages.
* The design should be well documented so that anyone adding to it can easily keep consistency across the site.

# System Behavior

Our systems behavior will be mainly be based off of user interactions. Once the user input for streams has been inserted, the system behavior will be initiated in good standing. In order for our system to remain in good standing, it will heavily depend on being resilient and being able to recover when wrong inputs are inserted. Once data is passed to the compilers, it wouldn’t have to undergo any kind of error checking since were not allowing the input from the user to cause any complications. Providing input options would bypass any of that. Another entity that could affect our system behavior would be the web server. Operational problems on servers could lead to malfunctioning issues within our performance. Lastly, we must make sure the user interface layout accommodates our users need on what content they choose to go with. Overall our service should remain in a steady -state.

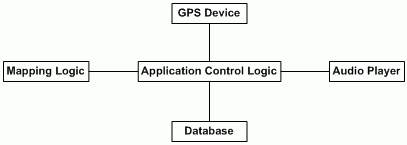
# Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

## High-Level Design (Architecture)

The high-level view or architecture consists of 5 major components:

****

**CompareStream Service**

**GUI Thread**

**.Net Framework**

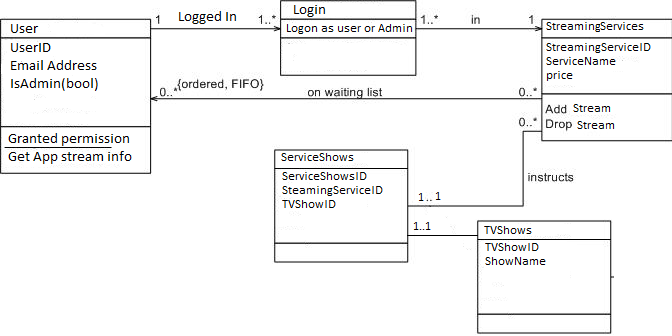
System Architecture

* The **CompareStream service** provides the user the ability to compare TV options from various
* The **Database** is a central repository for data on streaming services, TV packages, TV shows and price information that range across all these variables.
* The **Application Control Logic** is the main driver of the application. It presents information to the user and reacts to user inputs.
* The **.Net framework** is used as a primary framework base for the whole of the system.

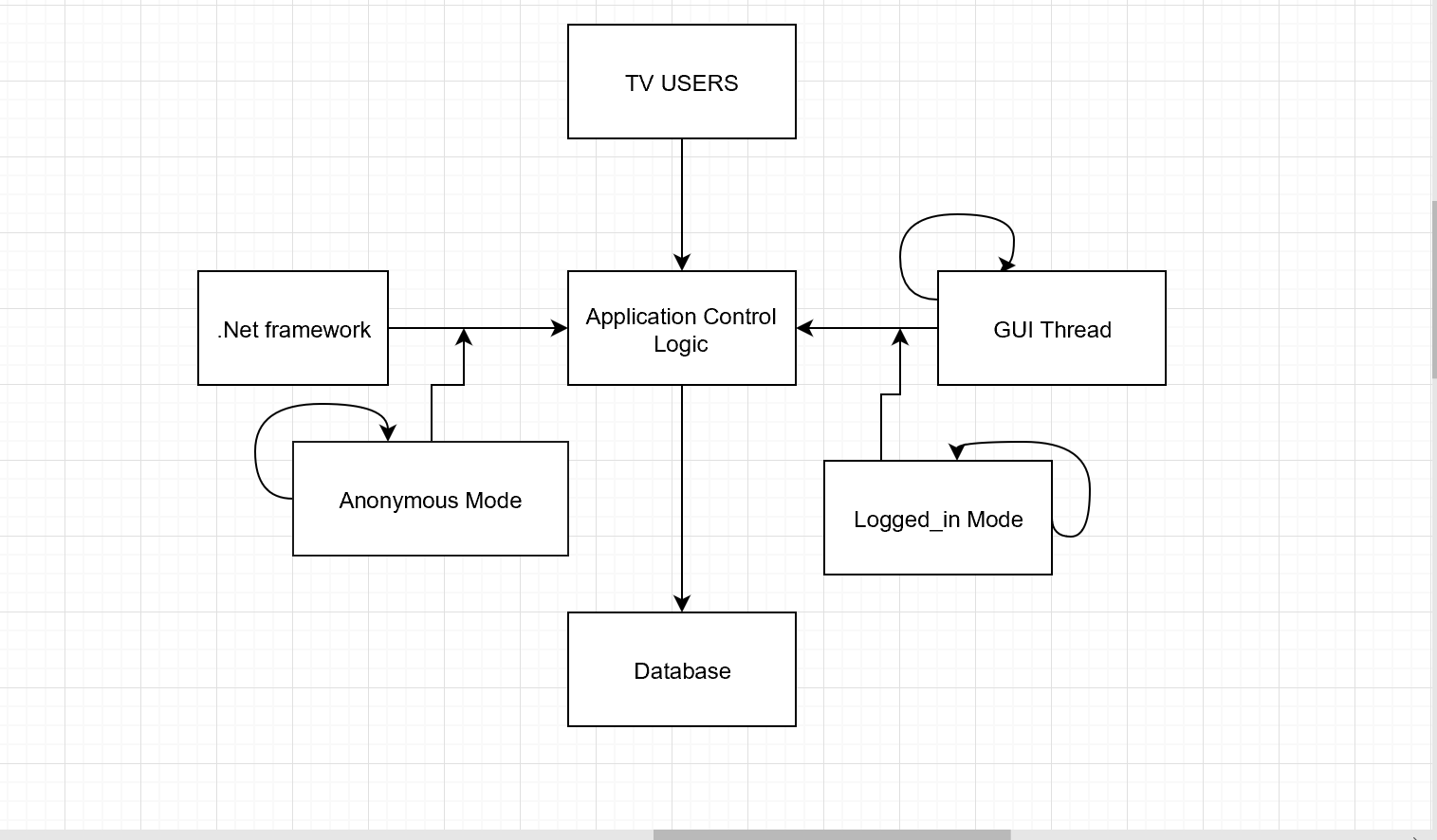
## Mid-Level Design

* + Split by classes
    - TV show
      * Dynamic:
        + editTV
      * Static:
        + Show ID
        + Name
    - StreamingService
      * Dynamic:
        + editStreamingService
        + earchStreamingService
      * Static:
        + service ID
        + name
        + price
    - User
      * Dynamic:
        + login
      * Static:
        + user ID
        + email
        + password
        + isAdmin
    - Report
      * Dynamic:
        + reportProblem
        + viewStatistics
        + viewReports
      * Static:
        + report ID
        + report description

## Detailed Class Design



# Process View



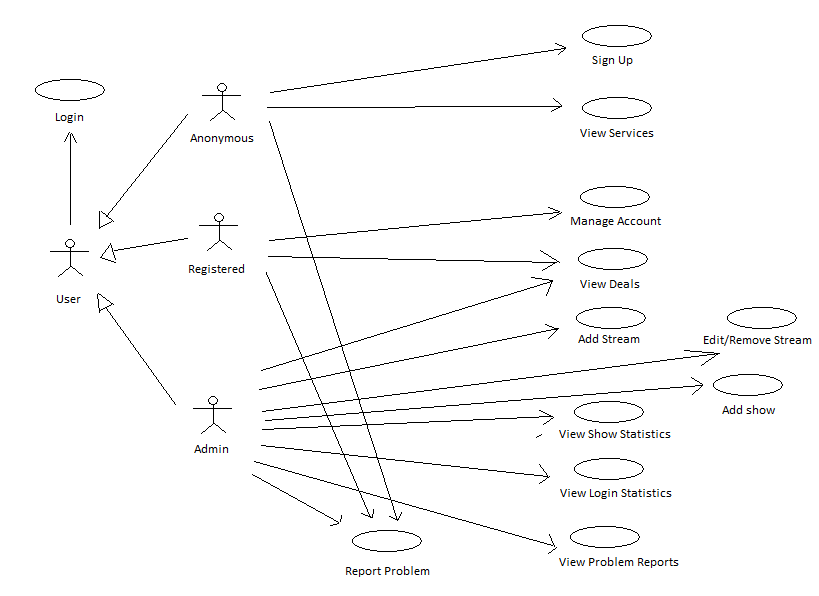
**TV Users**

Each loop represents a thread of control. The anonymous Mode only allows users to search shows and types of services. The Logged\_in Mode allow the users to view the best option for their favorite shows.

# Physical View

In web browsers on PC’s, laptops and mobile devices.

# Use Case View

******

***7.1.1 Login***

Brief Description:

This use case describes how a user logs into the TV CompareStream System. The actors starting this use case are Anonymous, Registered Users, and Admins.

***7.1.2 Sign Up***

Brief Description:

This use case allows an Anonymous User to register for access to the CompareStream Services. They will need to enter a valid email and choose a password. The actors starting this use case are Anonymous Users.

***7.1.3 View Services***

Brief Description:

This use case allows anonymous users to view services and TV shows are offered when the user has yet to register. The actor of this use case is Anonymous Users

***7.1.4 Manage Account***

Brief Description:

This use case allows a registered user to manage their account settings. The actor starting this use case is a Registered User.

***7.1.5 View Deals***

Brief Description:

This use case allows a registered users and admins to select their favorite TV shows. The system will display the best deals according to the users preferences. The main actors of this use case is Registered Users and Admins.

***7.1.6 Add Streams***

Brief Description:

This use case allows an Admin to create streaming services. The Admin chooses shows to add to the streaming service. The Admin is the actor of this use case.

***7.1.7 Edit/Remove Streams***

Brief Description:

This use case allows an Admin to remove out of date TV shows, or edit show availability. The actor in this use case is the Admin.

***7.1.8 Add Show***

Brief Description:

This use case allows an Admin to add additional shows to an existing streaming service. The actor in this use case is the Admin.

***7.1.9 View Show Statistics***

Brief Description:

This use case allows an Admin to observe what TV shows and services are most popular. The Admin may tailor other sections of the site to promote the most popular services to make the site stay as relevant as possible to site users. The actor in this use case is the Admin.

***7.1.10 View Login Statistics***

Brief Description:

This use case allows an Admin to see statistics related to who has logged in, at what time they logged in, and invalid login attempts to various accounts. The actor in this use case is the Admin.

***7.1.11 Report Problem***

Brief Description:

This use case allows all users to report problems encountered. The actors in this use case are the Anonymous, Registered, and Admin.

***7.1.12 View Problem Statistics***

Brief Description:

This use case allows an Admin to review all the problems reported on the system. The Admin is able to mark reports fixed and select which problems can be viewed, according to their attributes. The actor in this use case is the Admin.