ChillnFree

Software Architecture Document

Version <0.1>

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 16/11/2021 | 0.1 | Fill section 1,2,3,4 | Võ Hoàng Bảo Duy  Trần Ngọc Lam  Lê Minh Sĩ  Đặng Kiệt Hào  Lê Quốc Trọng |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**Introduction**](#_gjdgxs) **4**

[**Architectural Goals and Constraints**](#_30j0zll) **4**

[**Use-Case Model**](#_1fob9te) **4**

[**Logical View**](#_2et92p0) **5**

[Component: View](#_tyjcwt) 5

[Component: Controller](#_tr3nyqk164o1) 5

[Component: Model](#_rgajn42cfy3d) 6

[Component: Database](#_sg3fkwc3elhh) 7

[Component: Media Service](#_y7f2k2f9i7hs) 7

[**Deployment**](#_3dy6vkm) **7**

[**Implementation View**](#_1t3h5sf) **7**

Software Architecture Document

# Introduction

● Purpose: The documents provide a comprehensive view about system architecture, using different architecture models to describe different aspects of this system, to have a glimpse and vital decision being implemented on the system

● Scope: This software architecture document provides an architectural overview of the music website management to efficiently evaluate success of this particular software, having an overall look for developers to continuously make progress.

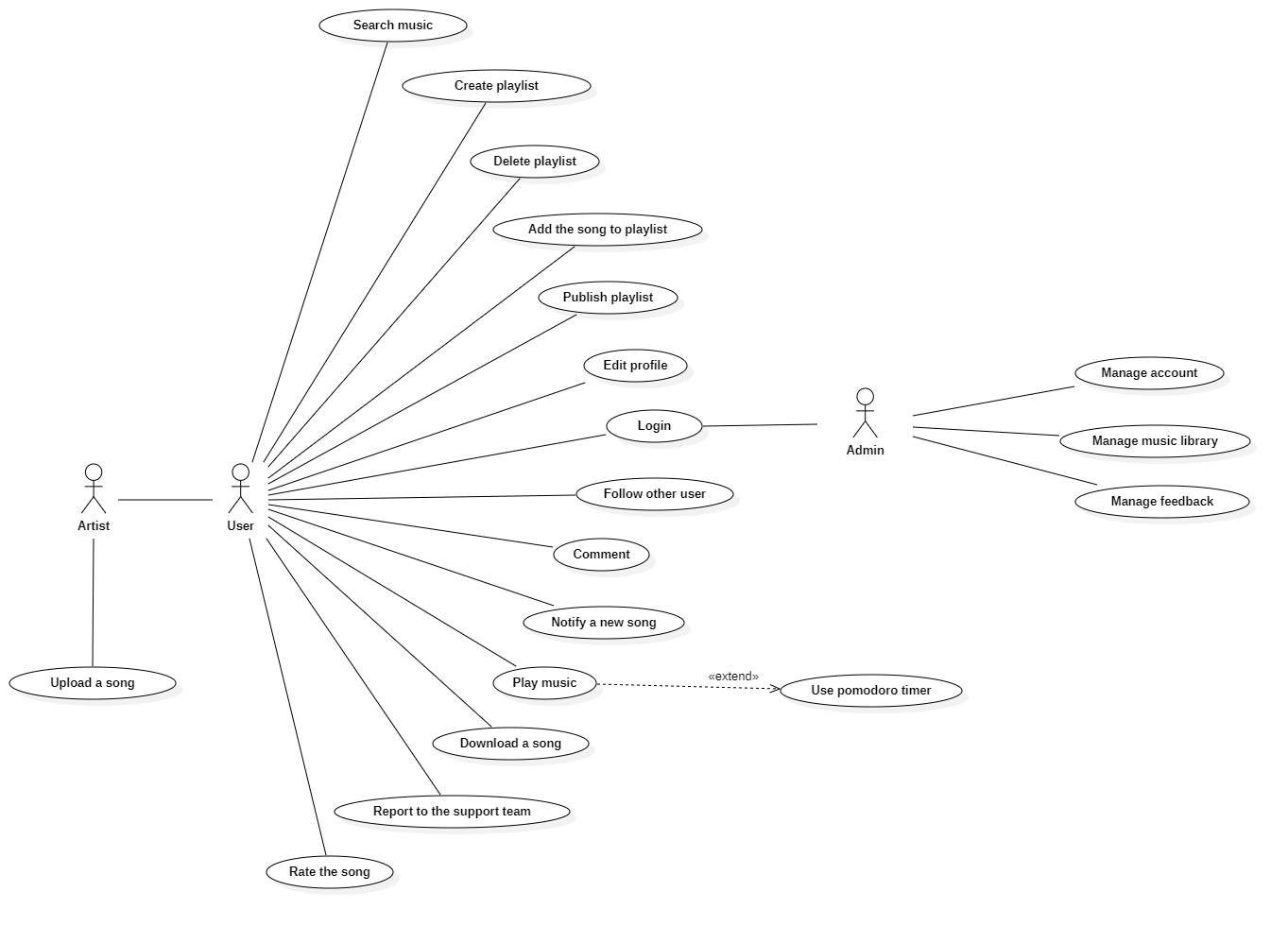
● Definitions, acronyms, abbreviations: See Glossary (4)

# Architectural Goals and Constraints

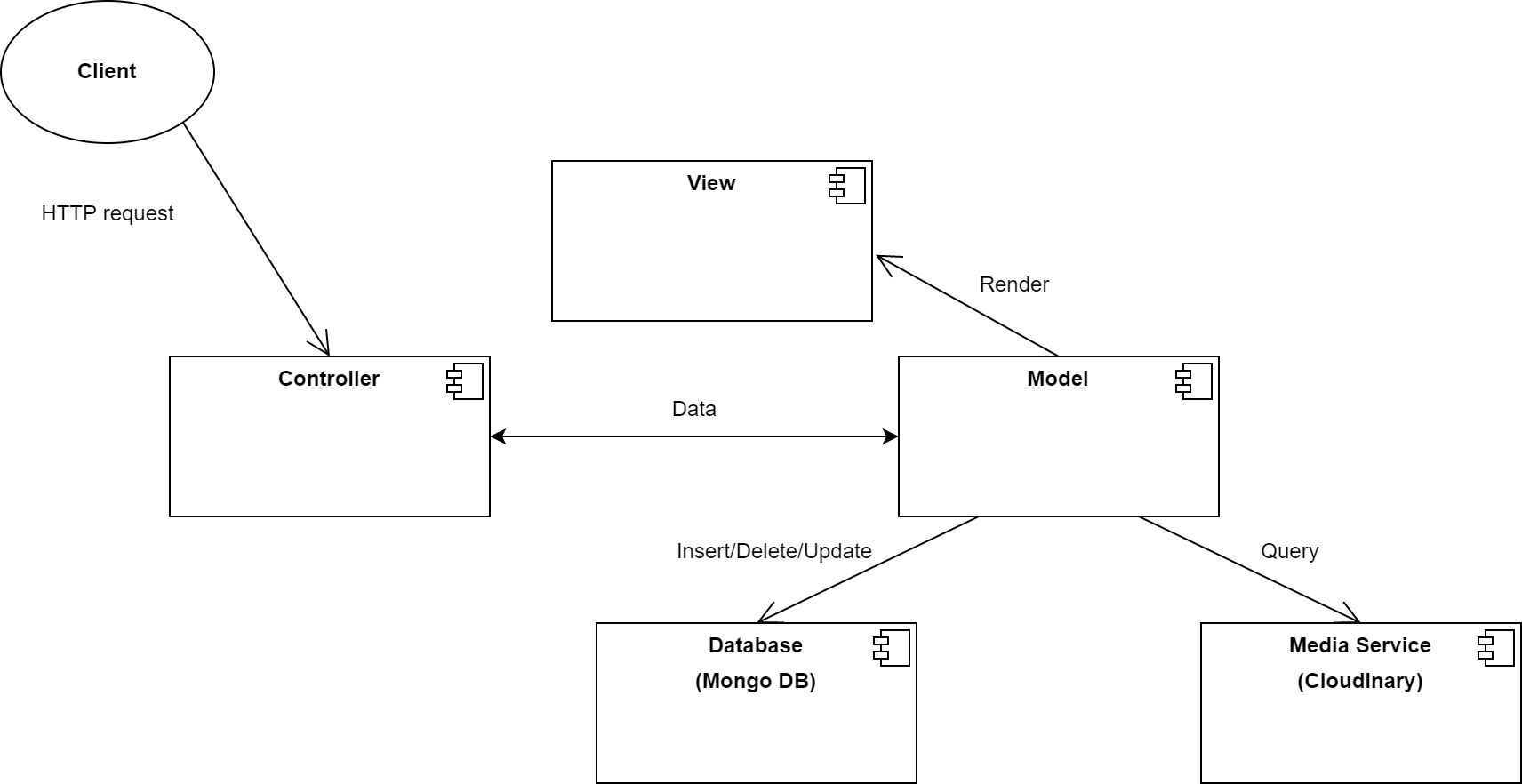
There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The existing google’s oauth2 code in their manual is needed for the login system.
2. All users’ functionality must be accessible through the internet.
3. The login system is secured by google, the rest of the user’s information must be protected from unauthorized access.
4. All server’s performance that is addressed in the Vision document must be considered in the development of the software.

# Use-Case Model



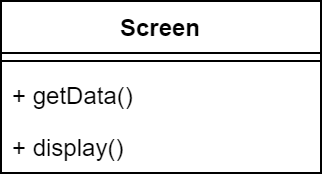
# Logical View

**

* Component View - Where HTML components are generated, display the screen to the Client.
* Component Controller – Where to handle user requests sent to the Model.
* Component Model - Where to store objects (object class).
* Component Database - A place to store user information, songs, and links to songs, images, ...
* Component Media Service – Where to store data including songs, images, ...

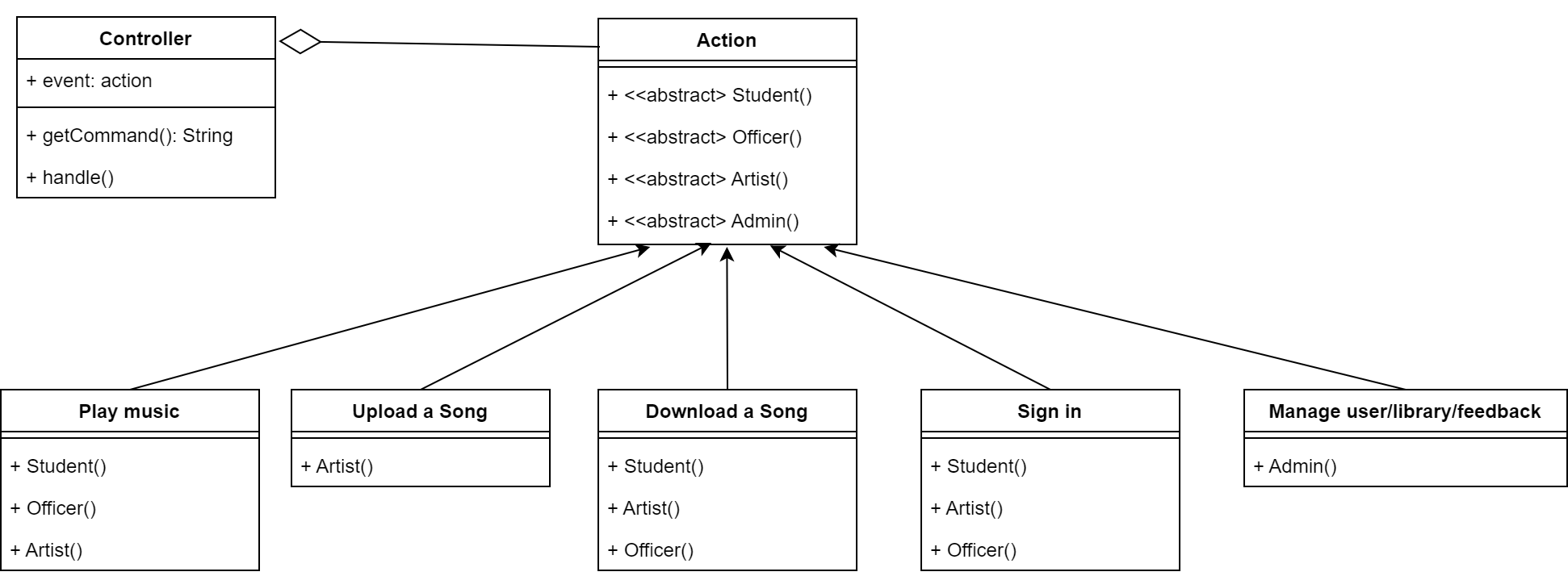
## Component: View

* Screen: the Html elements are generated and displayed to the client.



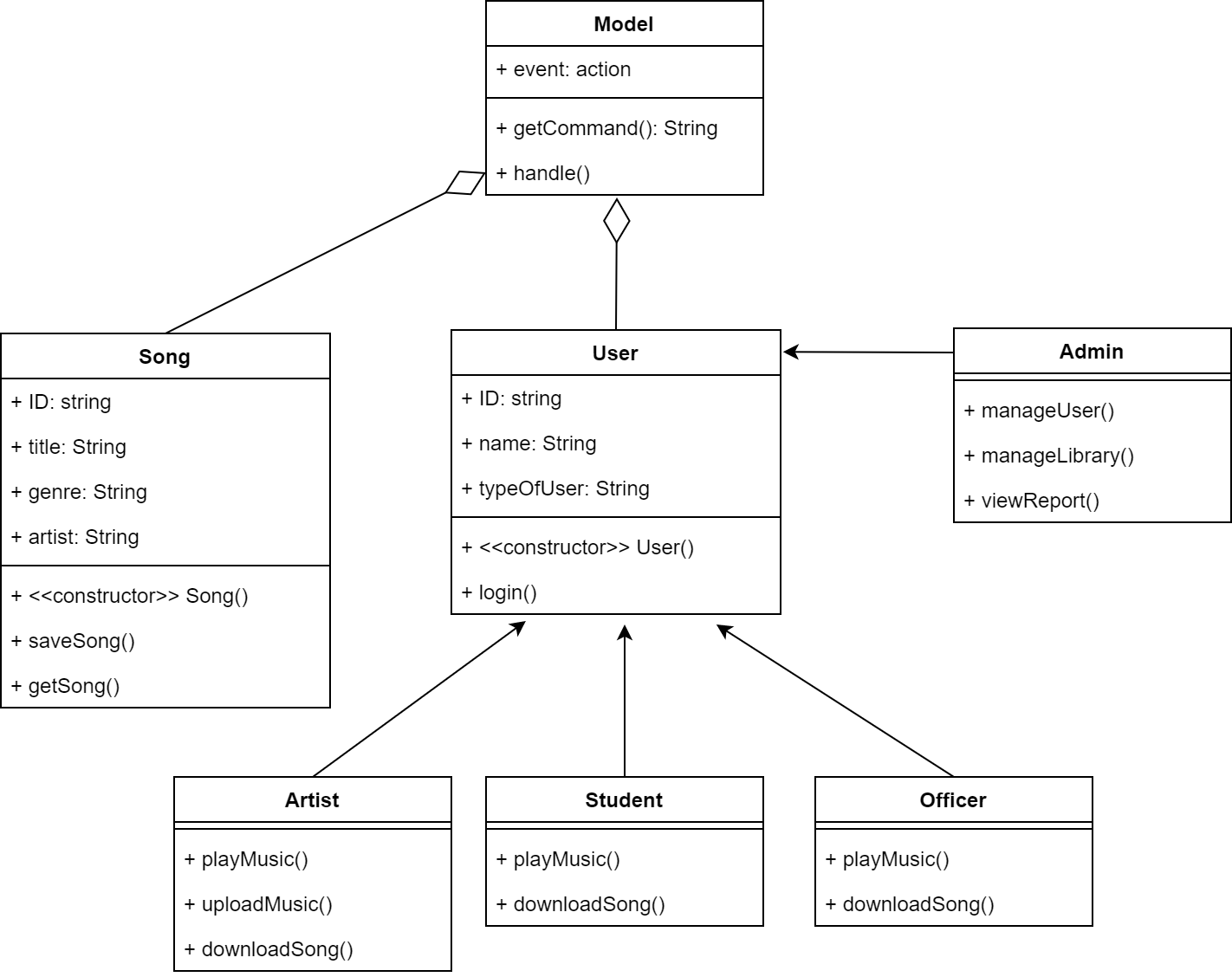
## Component: Controller

* Action: an abstract class that provides functions on 4 actors: student, worker, artist, admin.
* Controller: receive commands from client and handle user requests sent to model.



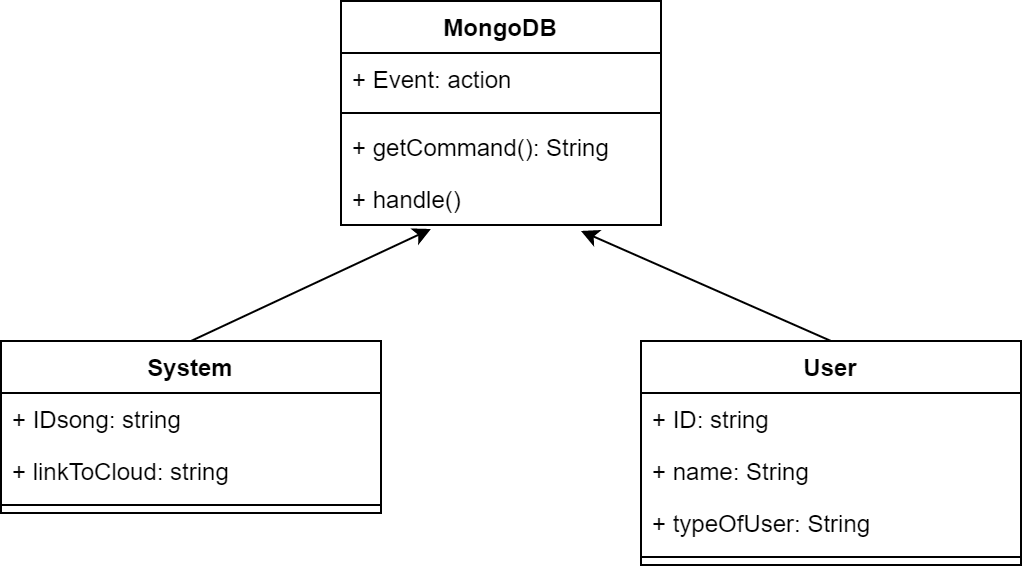
## Component: Model

The key class is User, which provides a login function and keeps data of a user logged in to the system. Its child classes have functions that can be manipulated with the database depending on the user role.



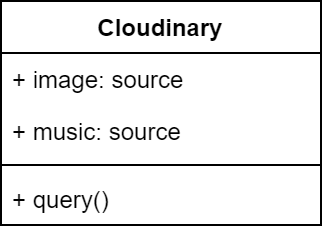
## Component: Database

* MongoDB: store user information, songs, and links to songs, images



## Component: Media Service

* Cloudinary: store songs, images



# Deployment

# Implementation View