

University of Engineering and Technology, Lahore

(New Campus)

Documentation

Project Title

|  |
| --- |
| MiniTube |

Department of Computer Science

CS165L Software Engineering

Group Members

Hashir Imtiaz (2022CS663)

Waqar Ali (2022CS666)

Malik Shujaat (2022CS679)

Submitted to: Sir. Usman Ghani

Table of Contents

[Introduction 4](#_Toc185166889)

[Project Overview 4](#_Toc185166890)

[Objectives 4](#_Toc185166891)

[Scope 4](#_Toc185166892)

[System Overview 4](#_Toc185166893)

[Features 4](#_Toc185166894)

[Technologies Used 5](#_Toc185166895)

[Use Case Diagrams 6](#_Toc185166896)

[Use Cases 6](#_Toc185166897)

[System Processes 7](#_Toc185166898)

[Database Design 8](#_Toc185166899)

[Tables and Relationships 8](#_Toc185166900)

[Code Structure 10](#_Toc185166901)

[Layers 10](#_Toc185166902)

[Key Classes 10](#_Toc185166903)

[System Design 12](#_Toc185166904)

[Class Diagrams 12](#_Toc185166905)

[Sequence Diagrams 13](#_Toc185166906)

[Installation Guide 15](#_Toc185166907)

[Prerequisites 15](#_Toc185166908)

[Steps to Run the Application 16](#_Toc185166909)

[Testing and Validation 16](#_Toc185166910)

[Test Cases 16](#_Toc185166911)

[Future Enhancements 17](#_Toc185166912)

[Appendices 17](#_Toc185166913)

[Code Snippets 17](#_Toc185166914)

[References 18](#_Toc185166915)

Table of Figures

[Figure 1: MiniTube Use Case Diagram 6](#_Toc185166785)

[Figure 2: MiniTube Class Diagram 12](#_Toc185166786)

[Figure 3: Video Upload Sequence Diagram 13](#_Toc185166787)

[Figure 4: Video Playback Sequence Diagram 14](#_Toc185166788)

# Introduction

## Project Overview

MiniTube is a simplified video platform inspired by YouTube. It is a desktop application created using WPF (Windows Presentation Foundation) with a focus on ease of use and essential features. The goal is to offer users a way to browse, search, and watch videos, as well as to manage their uploaded content in a streamlined and efficient manner. The platform is designed to be lightweight and user-friendly, targeting desktop users who need a simplified alternative to larger platforms.

## Objectives

The main objectives of MiniTube are:

* To provide users with a platform to explore, search, and view video content.
* To enable users to upload and manage their videos through an intuitive studio view.
* To offer essential video interactions like comments and likes.
* To ensure a seamless experience with minimal system requirements and optimal performance.

## Scope

MiniTube is tailored for users who primarily operate on desktop systems and require basic video-sharing capabilities without the complexities of a full-scale platform like YouTube. Its core functionalities include user authentication, video playback, content management, and social interaction features like commenting and liking. The application caters to both casual users who want to browse and interact with videos, and content creators who want to manage their uploads in a simple environment. While it does not replicate advanced features like live streaming or ad integration, it lays the groundwork for such enhancements in the future.

# System Overview

## Features

MiniTube includes a wide range of features designed to simplify video interactions for users and administrators alike. Below is a breakdown of its core features:

**For General Users**

* **User Authentication**: Users can log in to access personalized content and maintain their activity securely.
* **Video Browsing**: A home screen displays curated or randomized videos for users to explore easily.
* **Search Functionality**: A robust search bar lets users find videos based on keywords or titles quickly.
* **Video Playback**: A dedicated playback screen offers controls like play, pause, stop, and full-screen mode for an immersive viewing experience.
* **Likes and Comments**: Users can like videos or leave comments to interact with the content.

**For Content Creators**

* **Studio View**: Provides a centralized interface for users to upload, edit, and manage their videos efficiently.
* **Insights**: Allows content creators to view analytics related to likes and comments on their videos.

## Technologies Used

MiniTube relies on modern tools and technologies to deliver a robust and efficient solution:

* **Programming Language**: C# is used as the primary language for its strong integration with WPF and .NET.
* **Frameworks**: The application uses WPF for building the user interface and Entity Framework Core for database interactions.
* **Database**: SQL Server is used to store and manage application data securely and efficiently.
* **Development Tools**: The project is developed in Visual Studio with version control handled through Git.
* **Other Technologies**: XAML is used for defining the user interface layout and styling.

# Use Case Diagrams

## Use Cases

MiniTube caters to multiple types of users including regular users and content creators. Below are the key use cases highlighting their interactions with the system:

**Login and Registration**

* **Actors**: User
* **Description**: Users can create accounts and log in to access personalized features.
* **Purpose**: Ensures secure access to individual accounts and associated content.

**Browse and Search Videos**

* **Actors**: User
* **Description**: Users can explore videos through browsing and use the search bar to find specific content.
* **Purpose**: Provides an easy way to discover content on the platform.

**Video Playback**

* **Actors**: User
* **Description**: Users can play, pause, stop, and watch videos in full-screen mode.
* **Purpose**: Allows seamless video consumption.

**Interact with Content**

* **Actors**: User
* **Description**: Users can like videos and leave comments to engage with the content.
* **Purpose**: Facilitates interaction and engagement with the platform.

**Manage Videos**

* **Actors**: Content Creator (Admin-level user)
* **Description**: Creators can upload, edit, and delete their videos from the studio view.
* **Purpose**: Empowers creators to manage their content effectively.

**Diagram**

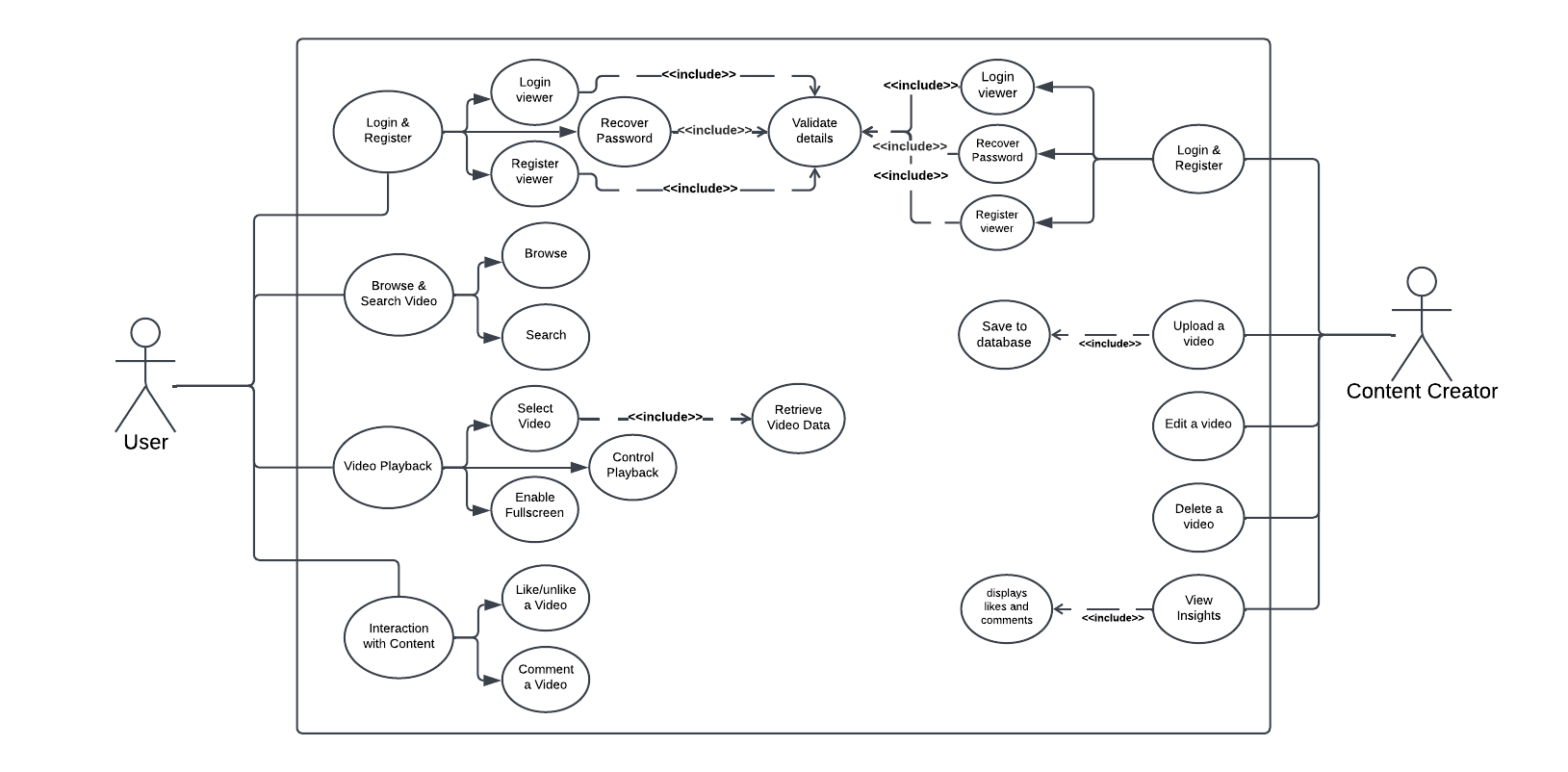


Figure : MiniTube Use Case Diagram

## System Processes

Use case diagrams will illustrate how the user roles interact with the core features of the application. Below are the main processes represented:

**User Role Interaction**

* **Login System**: Flow from authentication to accessing user-specific content.
* **Search Process**: Flow from entering a query to displaying search results.

**Content Creator Role Interaction**

* **Upload Process**: From selecting a video file to it being published on the platform.
* **Analytics**: Accessing insights on video performance through likes and comments.

# Database Design

## Tables and Relationships

**Users Table**

This table stores user account information.

*CREATE TABLE Users (*

*UserID INT IDENTITY(1, 1) PRIMARY KEY,*

*Username VARCHAR(255) NOT NULL,*

*Email VARCHAR(255) NOT NULL,*

*Password VARCHAR(255) NOT NULL,*

*Role VARCHAR(50) NOT NULL*

*);*

**Videos Table**

This table stores video-related data.

*CREATE TABLE Videos (*

*VideoID INT IDENTITY(1, 1) PRIMARY KEY,*

*UserID INT NOT NULL,*

*Title VARCHAR(255) NOT NULL,*

*Description TEXT,*

*Thumbnail VARBINARY(MAX),*

*VideoFile VARBINARY(MAX),*

*UploadDate DATETIME DEFAULT GETDATE(),*

*FOREIGN KEY (UserID) REFERENCES Users(UserID)*

*);*

**Comments Table**

This table captures user interactions through comments.

*CREATE TABLE Comments (*

*CommentID INT IDENTITY(1, 1) PRIMARY KEY,*

*VideoID INT NOT NULL,*

*UserID INT NOT NULL,*

*CommentText TEXT NOT NULL,*

*CommentDate DATETIME DEFAULT GETDATE(),*

*FOREIGN KEY (VideoID) REFERENCES Videos(VideoID),*

*FOREIGN KEY (UserID) REFERENCES Users(UserID)*

*);*

**Likes Table**

This table tracks likes on videos.

*CREATE TABLE Likes (*

*LikeID INT IDENTITY(1, 1) PRIMARY KEY,*

*VideoID INT NOT NULL,*

*UserID INT NOT NULL,*

*FOREIGN KEY (VideoID) REFERENCES Videos(VideoID),*

*FOREIGN KEY (UserID) REFERENCES Users(UserID)*

*);*

**Relationships**

1. **Users and Videos**: Each user can upload multiple videos, forming a one-to-many relationship.
2. **Videos and Comments**: A video can have many comments, but each comment belongs to a specific video.
3. **Videos and Likes**: Users can like multiple videos, creating a many-to-many relationship between users and videos.
4. **Users and Comments**: Users can comment on videos, creating a one-to-many relationship between users and comments.

# Code Structure

The MiniTube application is organized following the MVVM (Model-View-ViewModel) architectural pattern, ensuring a clean separation of concerns and maintainability.

## Layers

* **View**: This layer consists of XAML files and their corresponding code-behind files. It is responsible for the user interface and user interactions. The UI components are designed to display data and respond to user inputs. Examples include the main window, login page, video player, and upload interfaces.
* **Model**: This layer contains the Entity Framework models that interact with the database. It defines the structure of the database tables and manages data retrieval and storage. The models represent the entities such as users, videos, comments, and likes.
* **ViewModel**: This layer implements the business logic and acts as a bridge between the View and Model layers. It processes user inputs, retrieves data from the Model, and updates the View. It also ensures that the application adheres to the MVVM pattern, enhancing scalability and testability.

## Key Classes

1. **DbMiniTubeContext**:
   * This class serves as the database context for the application, managing the connection to the database and executing queries.
   * It leverages Entity Framework to map models to database tables.
   * Example:

*public class DbMiniTubeContext : DbContext*

*{*

*public DbSet<User> Users { get; set; }*

*public DbSet<Video> Videos { get; set; }*

*public DbSet<Comment> Comments { get; set; }*

*public DbSet<Like> Likes { get; set; }*

*protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)*

*{*

*optionsBuilder.UseSqlServer("YourConnectionStringHere");*

*}*

*}*

1. **Entity Classes**:
   * These classes define the structure and relationships of database tables.
   * **User Class**: Represents the Users table in the database.

*public class User*

*{*

*public int UserID { get; set; }*

*public string Username { get; set; }*

*public string Email { get; set; }*

*public string Password { get; set; }*

*public string Role { get; set; }*

*public ICollection<Video> Videos { get; set; }*

*}*

* + **Video Class**: Represents the Videos table, storing video-related information.

*public class Video*

*{*

*public int VideoID { get; set; }*

*public string Title { get; set; }*

*public string Description { get; set; }*

*public byte[] Thumbnail { get; set; }*

*public byte[] VideoFile { get; set; }*

*public DateTime UploadDate { get; set; }*

*public int UserID { get; set; }*

*public User User { get; set; }*

*public ICollection<Comment> Comments { get; set; }*

*public ICollection<Like> Likes { get; set; }*

*}*

* + **Comment and Like Classes**: Define the structure for user interactions such as comments and likes.

# System Design

## Class Diagrams

The class diagram outlines the relationships between core entities in the MiniTube application, illustrating the system's architecture and the interaction between components. Below is an overview of the key classes:

* **User**: Represents application users with properties such as UserID, Username, Email, Password, and Role. It is associated with multiple Videos, Comments, and Likes.
* **Video**: Represents video content uploaded by users. Includes properties like VideoID, Title, Description, Thumbnail, VideoFile, and UploadDate. Videos have relationships with the User, Comment, and Like classes.
* **Comment**: Represents user-generated comments on videos. Includes properties like CommentID, CommentText, CommentDate, and foreign keys linking to User and Video classes.
* **Like**: Tracks user likes on videos. Includes properties like LikeID and foreign keys linking to User and Video classes.
* **DbMiniTubeContext**: Manages database interactions and provides DbSet collections for the entities.

**Diagram**

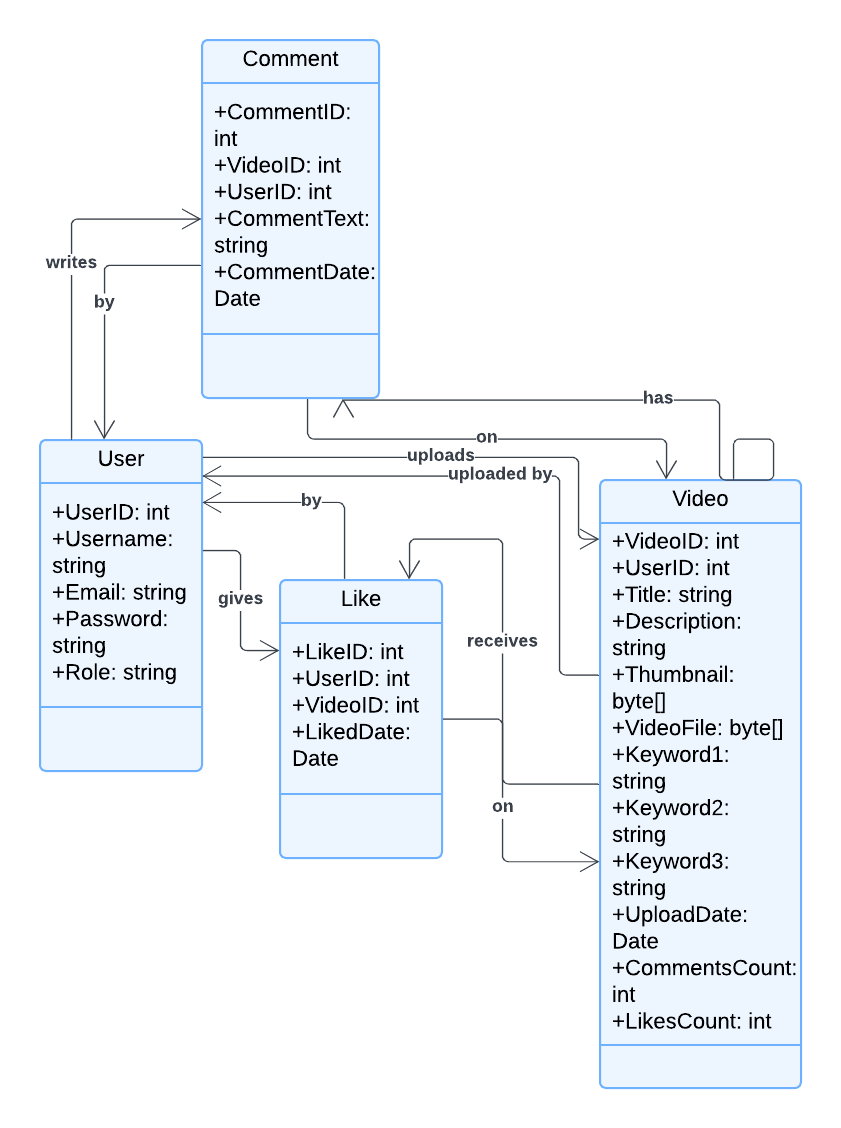


Figure : MiniTube Class Diagram

## Sequence Diagrams

Sequence diagrams depict the step-by-step interaction between components during specific processes, such as video upload or playback.

1. **Video Upload Process**:
   * **Actors**: User, UI (View), ViewModel, Database (Model).
   * **Steps**:
     + User selects a video file to upload.
     + The UI sends the video details (title, description, thumbnail) and the file to the ViewModel.
     + The ViewModel validates the data and interacts with the DbMiniTubeContext to save the video metadata and file.
     + DbMiniTubeContext stores the video details in the Videos table and associates it with the User.
     + The ViewModel updates the UI to confirm successful upload.

**Diagram**

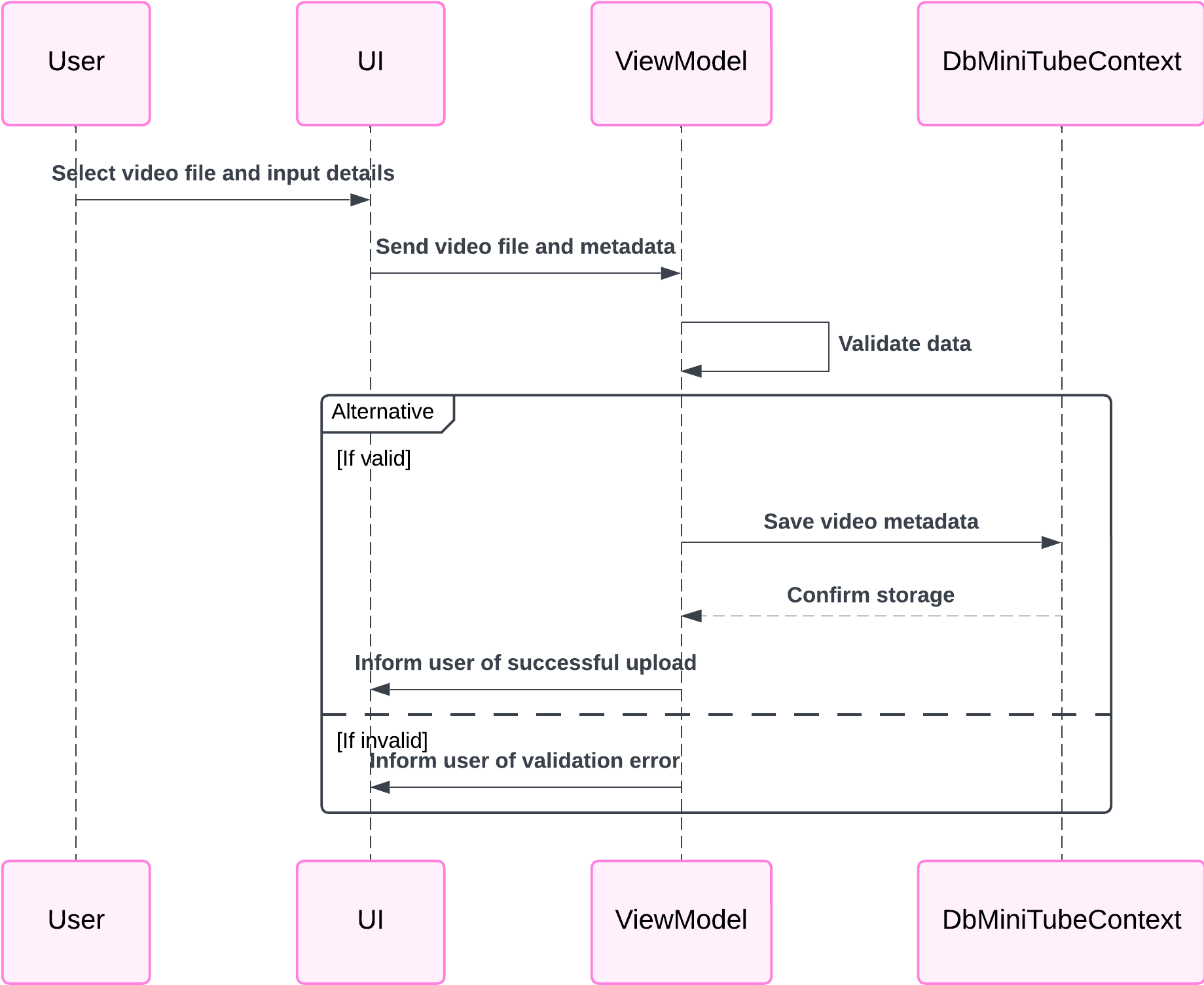


Figure : Video Upload Sequence Diagram

1. **Video Playback Process**:
   * **Actors**: User, UI (View), ViewModel, Database (Model).
   * **Steps**:
     + User selects a video to play.
     + The UI requests video details (e.g., title, description, file path) from the ViewModel.
     + The ViewModel retrieves the video details from the database via DbMiniTubeContext.
     + The ViewModel sends the video file data to the UI.
     + The UI plays the video using the embedded media player.

**Diagram**

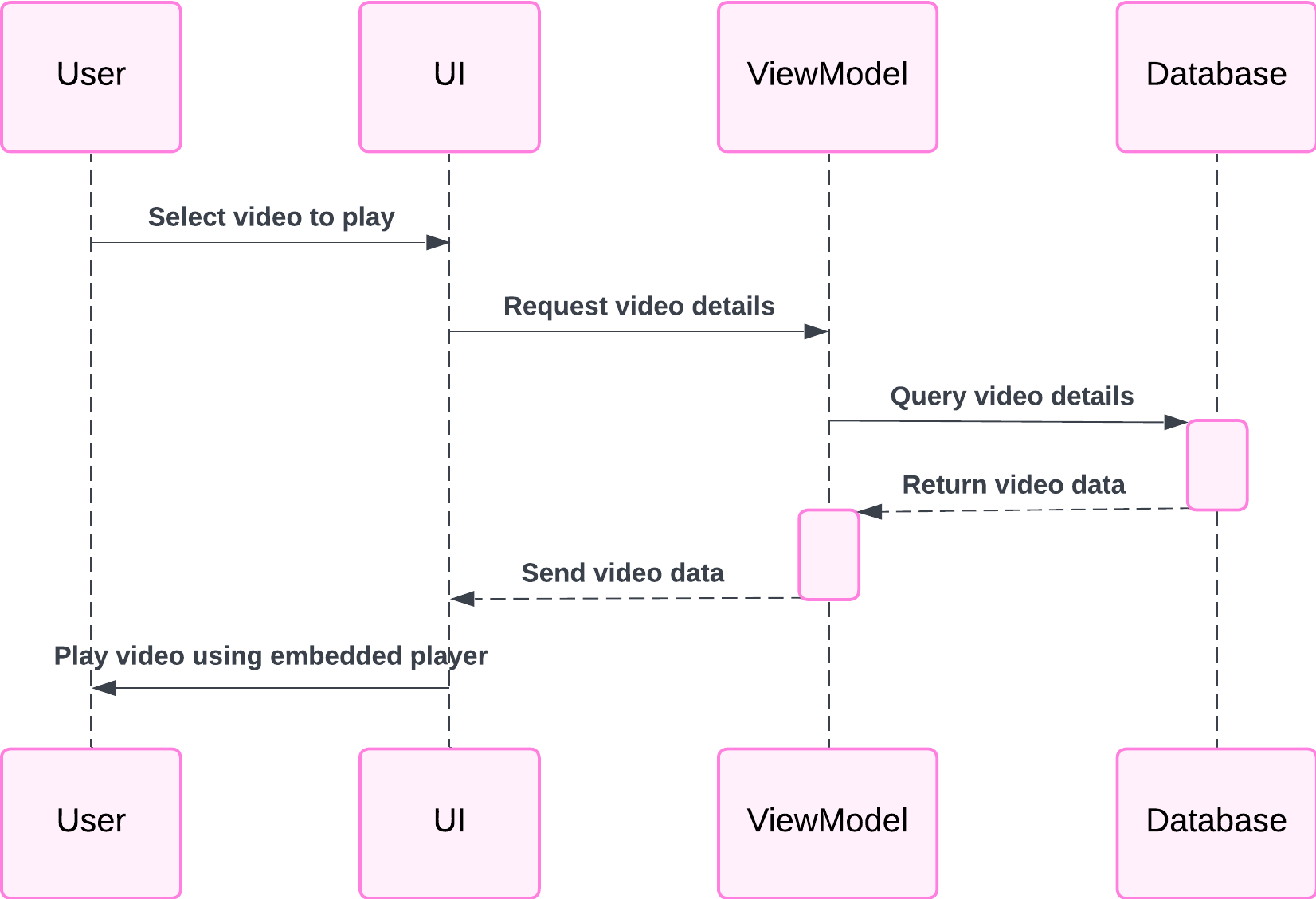


Figure : Video Playback Sequence Diagram

# Installation Guide

This section provides a step-by-step guide to set up and run the MiniTube application on your local machine.

## Prerequisites

To run the application, ensure that the following tools and frameworks are installed:

1. **Visual Studio**: A development environment to build and run the application.
2. **SQL Server**: Required for the database backend.
3. **.NET Core SDK**: Provides the runtime and libraries needed to execute the application.

## Steps to Run the Application

Follow these steps to set up and execute the MiniTube application:

1. **Clone the Repository**:
   * Open a terminal or command prompt.
   * Clone the application’s repository from GitHub using the following command:

*git clone https://github.com/MalikShujaatAli/MiniTube.git*

1. **Setup Database**:
   * Open SQL Server Management Studio (SSMS).
   * Create a new database for the application.
   * Run the provided SQL scripts (included in the repository) to create the necessary tables: Users, Videos, Comments, and Likes.
2. **Build and Run**:
   * Open the cloned solution in Visual Studio.
   * Restore the NuGet dependencies by building the solution.
   * Verify that the database connection string in appsettings.json or the relevant configuration file matches your SQL Server setup.
   * Start the application by pressing **F5** or selecting the **Run** option.

# Testing and Validation

## Test Cases

The application has been tested with the following scenarios to ensure core functionality operates as intended:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Result | Status |
| TC1 | Login with valid details | Successful login | Passed |
| TC2 | Upload a video | Video visible in Studio | Passed |
| TC3 | Search functionality | Relevant videos appear | Passed |
| TC4 | Login with invalid details | Error message is displayed | Passed |
| TC5 | Comment on a video | Comment appears immediately | Passed |
| TC6 | Like a video | Like count updates | Passed |

**Results**

The MiniTube application has passed all core test cases. The system demonstrates stable performance under normal usage conditions. Additional performance and stress tests may be required for future scalability validation.

# Future Enhancements

The following enhancements are proposed to improve user experience and system performance:

1. **Add Playlists and Subscriptions**: Allow users to organize videos into playlists and follow other creators for updates.
2. **Enable Video Analytics**: Provide creators with insights into views, likes, and other engagement metrics.
3. **Implement Live Streaming**: Support real-time video broadcasting to enhance user interaction.
4. **Lessen Memory Consumption**: Optimize resource usage to reduce the application's memory footprint during extended use.
5. **Performance Optimization**: Improve database queries and UI responsiveness for smoother functionality.

# Appendices

## Code Snippets

Below are key snippets demonstrating critical application features:

**Login Logic**:

*public bool ValidateUser(string email, string password)*

*{*

*return dbContext.Users.Any(user => user.Email == email && user.Password == password);*

*}*

**Video Upload**:

*public void UploadVideo(Video video)*

*{*

*dbContext.Videos.Add(video);*

*dbContext.SaveChanges();*

*}*

**Search Functionality**:

*public List<Video> SearchVideos(string query)*

*{*

*return dbContext.Videos*

*.Where(video => video.Title.Contains(query) || video.Description.Contains(query))*

*.ToList();*

*}*

# References

1. [WPF Documentation](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/)
2. [Entity Framework Core Documentation](https://docs.microsoft.com/en-us/ef/core/)