YouView

Like YouTube But Simpler

Jonathan Glaab

Minh Triet Vu

Elizabeth Mokrusa

BACKGROUND

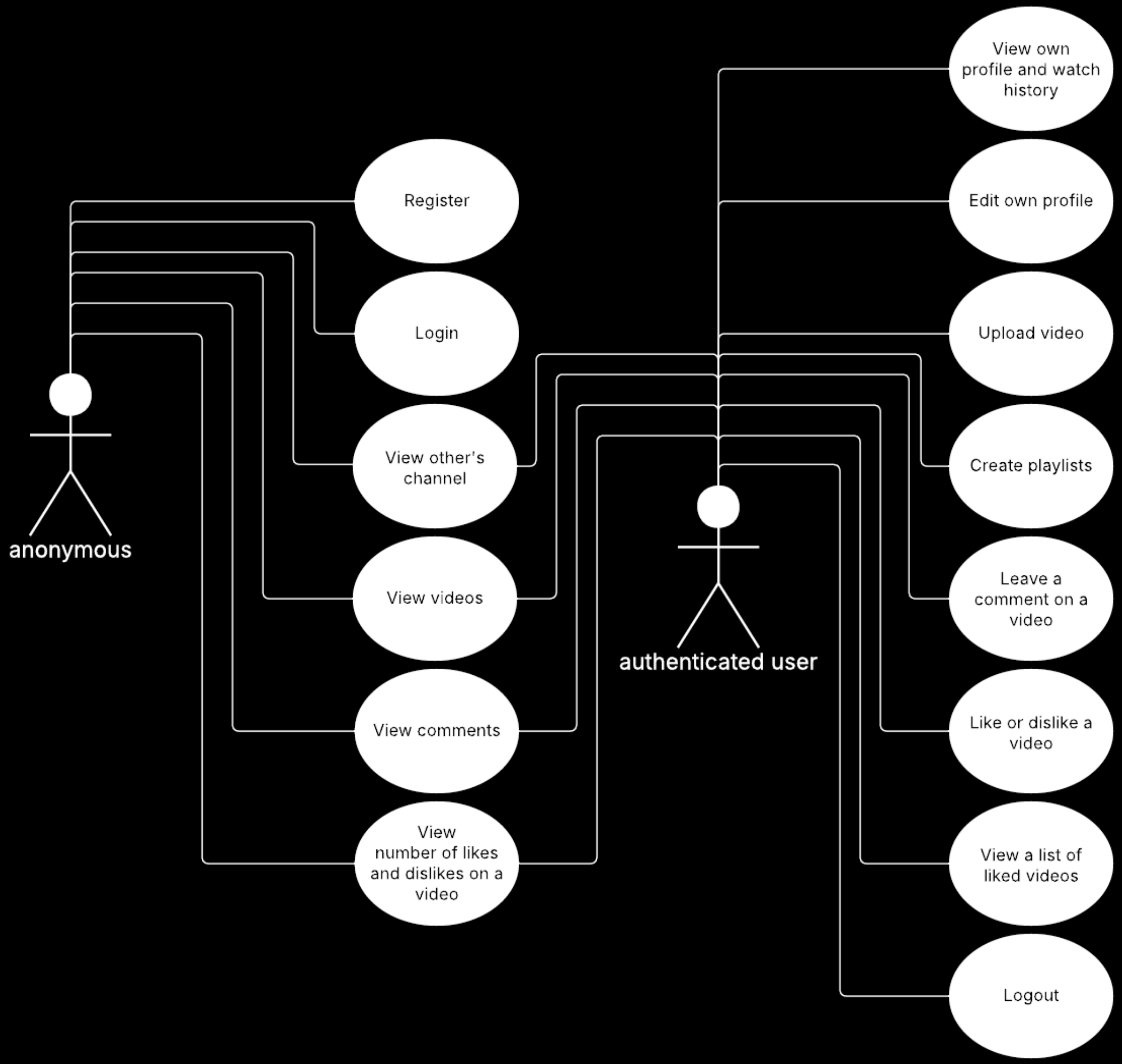
- Some people are seeking alternatives to YouTube and larger sites
- YouTube can be opaque and difficult for creators to break into since there is already so much competition for attention there
- We wanted to create an alternative platform for sharing and storing videos
- To be competitive we have implemented video processing and AI for summaries and previews
- To stay viable we have ad-based revenue streams to keep the service free for end-users

OUR SOLUTION

- Users are able to:
  - Register and manage profile information including: updating photo, editing user information and bio, changing and verifying a new email address and setting a new password.
  - See trending videos or scroll/search all videos.
  - Ability to like and comment on videos.
  - View creator channel with their videos and playlists.
  - Ability to view a history of watched videos.
  - See all liked videos.

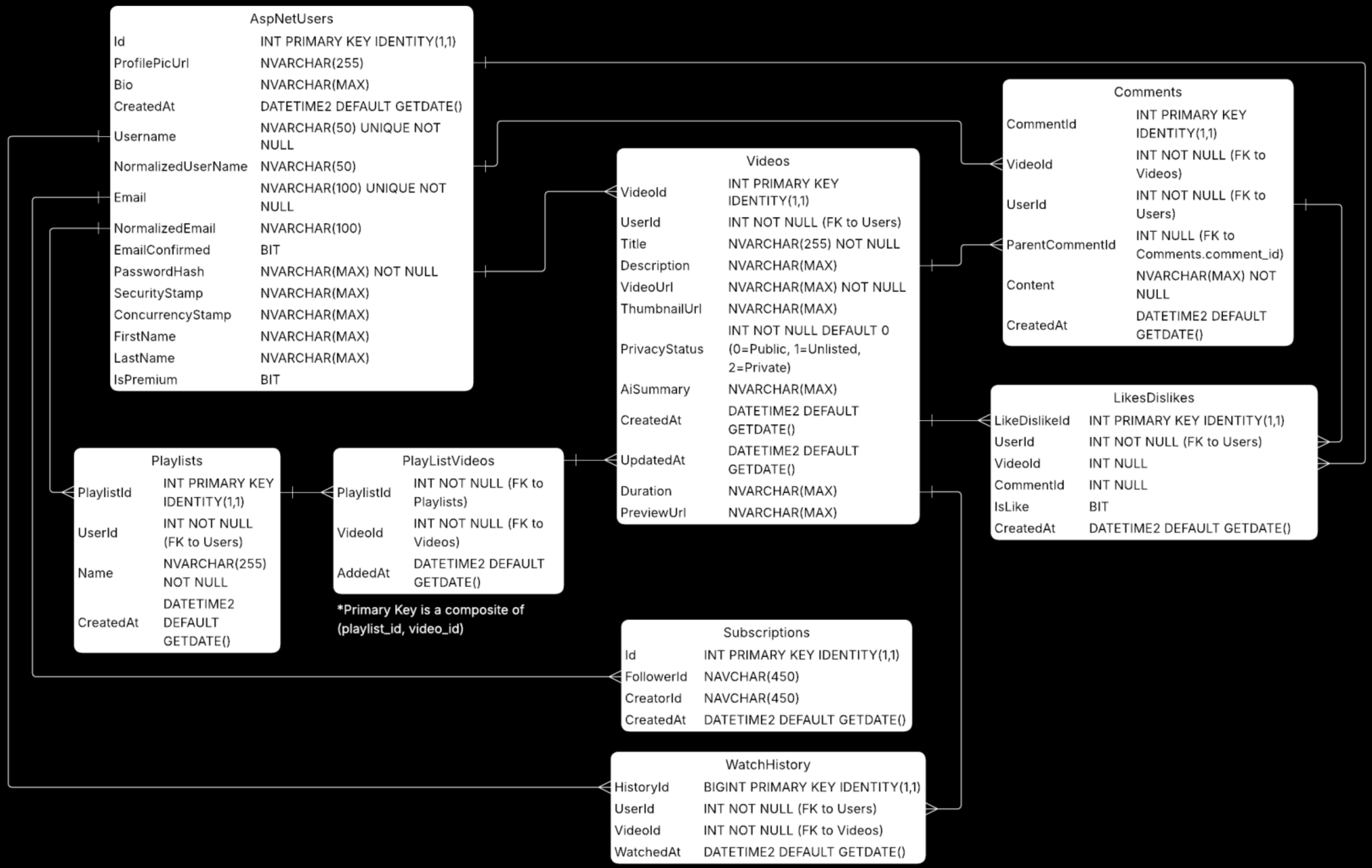
OUR SOLUTION

- Use case diagram



CHALLENGES AND SOLUTIONS: DATABASE

- Azure SQL



AspNetUsers

Id	INT PRIMARY KEY IDENTITY(1,1)
ProfilePicUrl	NVARCHAR(255)
Bio	NVARCHAR(MAX)
CreatedAt	DATETIME2 DEFAULT GETDATE()
Username	NVARCHAR(50) UNIQUE NOT NULL
NormalizedUserName	NVARCHAR(50)
Email	NVARCHAR(100) UNIQUE NOT NULL
NormalizedEmail	NVARCHAR(100)
EmailConfirmed	BIT
PasswordHash	NVARCHAR(MAX) NOT NULL
SecurityStamp	NVARCHAR(MAX)
ConcurrencyStamp	NVARCHAR(MAX)
FirstName	NVARCHAR(MAX)
LastName	NVARCHAR(MAX)
IsPremium	BIT

Comments

CommentId	INT PRIMARY KEY IDENTITY(1,1)
VideoId	INT NOT NULL (FK to Videos)
UserId	INT NOT NULL (FK to Users)
ParentCommentId	INT NULL (FK to Comments.comment_id)
Content	NVARCHAR(MAX) NOT NULL
CreatedAt	DATETIME2 DEFAULT GETDATE()

Videos

VideoId	INT PRIMARY KEY IDENTITY(1,1)
UserId	INT NOT NULL (FK to Users)
Title	NVARCHAR(255) NOT NULL
Description	NVARCHAR(MAX)
VideoUrl	NVARCHAR(MAX) NOT NULL
ThumbnailUrl	NVARCHAR(MAX)
PrivacyStatus	INT NOT NULL DEFAULT 0 (0=Public, 1=Unlisted, 2=Private)
AiSummary	NVARCHAR(MAX)
CreatedAt	DATETIME2 DEFAULT GETDATE()
UpdatedAt	DATETIME2 DEFAULT GETDATE()
Duration	NVARCHAR(MAX)
PreviewUrl	NVARCHAR(MAX)

LikesDislikes

LikeDislikeId	INT PRIMARY KEY IDENTITY(1,1)
UserId	INT NOT NULL (FK to Users)
VideoId	INT NULL
CommentId	INT NULL
IsLike	BIT
CreatedAt	DATETIME2 DEFAULT GETDATE()

Playlists

PlaylistId	INT PRIMARY KEY IDENTITY(1,1)
UserId	INT NOT NULL (FK to Users)
Name	NVARCHAR(255) NOT NULL
CreatedAt	DATETIME2 DEFAULT GETDATE()

PlayListVideos

PlaylistId	INT NOT NULL (FK to Playlists)
VideoId	INT NOT NULL (FK to Videos)
AddedAt	DATETIME2 DEFAULT GETDATE()

*Primary Key is a composite of (playlist_id, video_id)

Subscriptions

Id	INT PRIMARY KEY IDENTITY(1,1)
FollowerId	NAVCHAR(450)
CreatorId	NAVCHAR(450)
CreatedAt	DATETIME2 DEFAULT GETDATE()

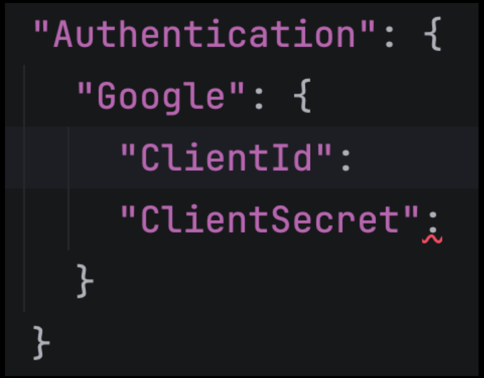
WatchHistory

HistoryId	BIGINT PRIMARY KEY IDENTITY(1,1)
UserId	INT NOT NULL (FK to Users)
VideoId	INT NOT NULL (FK to Videos)
WatchedAt	DATETIME2 DEFAULT GETDATE()

CHALLENGES AND SOLUTIONS: GOOGLE AUTH

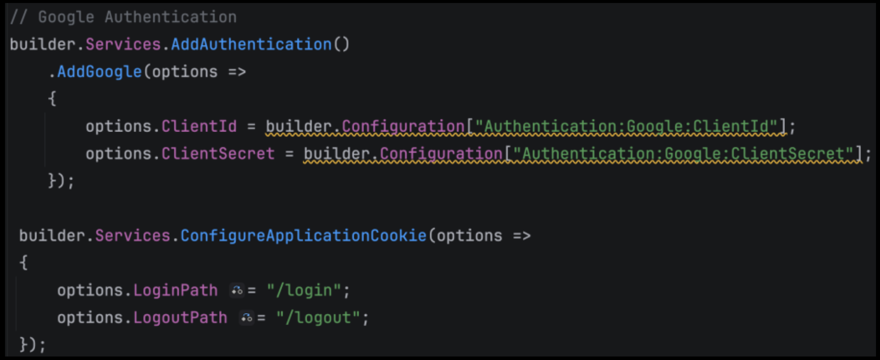
- Challenge: Most websites now allow third party authentication and users expect to have the option.

appsettings.json

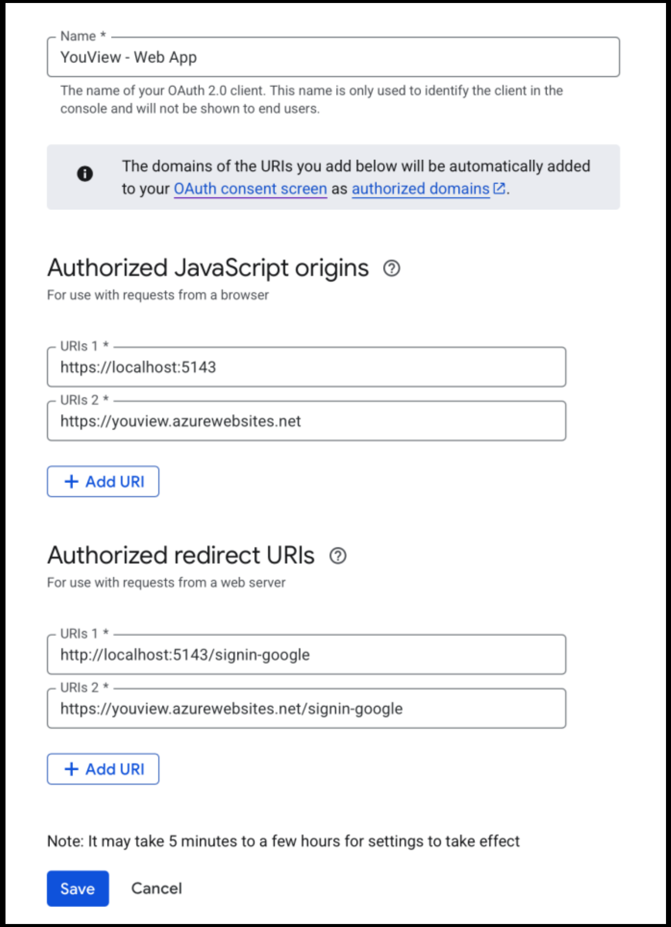


```
"Authentication": {
  "Google": {
    "ClientId":
    "ClientSecret":
  }
}
```

Program.cs



```
// Google Authentication
builder.Services.AddAuthentication()
    .AddGoogle(options =>
    {
        options.ClientId = builder.Configuration["Authentication:Google:ClientId"];
        options.ClientSecret = builder.Configuration["Authentication:Google:ClientSecret"];
    });

 builder.Services.ConfigureApplicationCookie(options =>
 {
     options.LoginPath = "/login";
     options.LogoutPath = "/logout";
 });
```



Name *: YouView - Web App

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

URIs 1 *: https://localhost:5143

URIs 2 *: https://youview.azurewebsites.net

Authorized redirect URIs

For use with requests from a web server

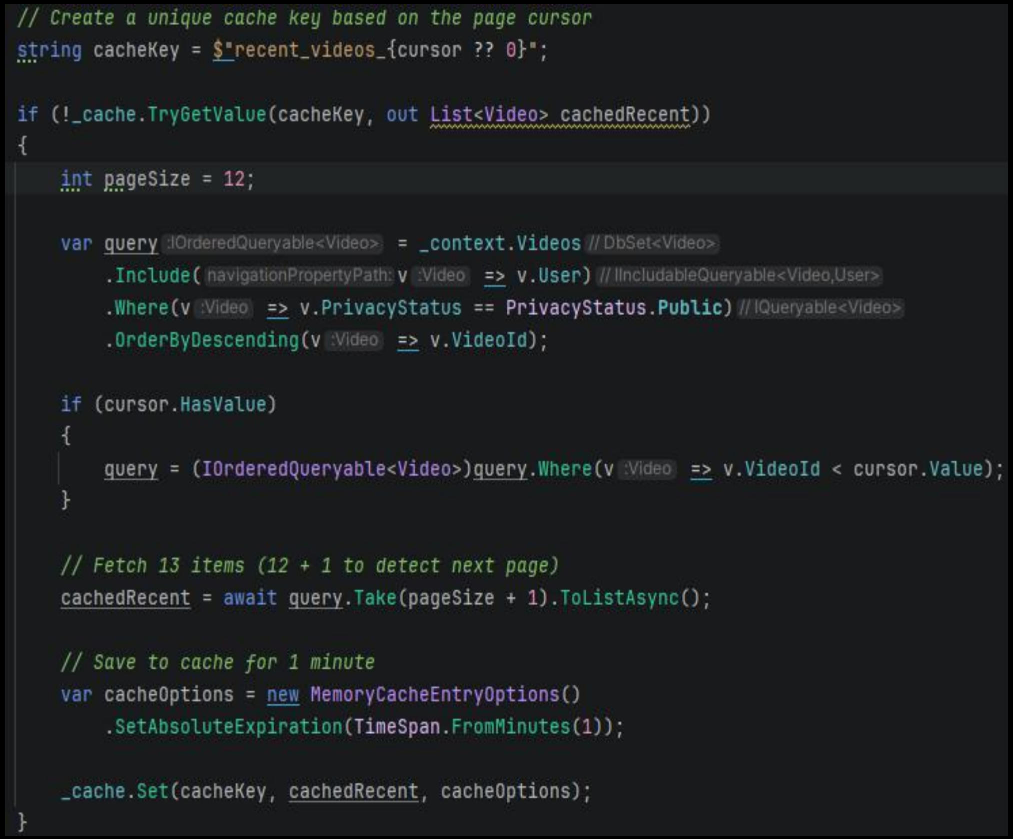
URIs 1 *: http://localhost:5143/signin-google

URIs 2 *: https://youview.azurewebsites.net/signin-google

Note: It may take 5 minutes to a few hours for settings to take effect

CHALLENGES AND SOLUTIONS:QUERIES VIDEOS FOR INDEX

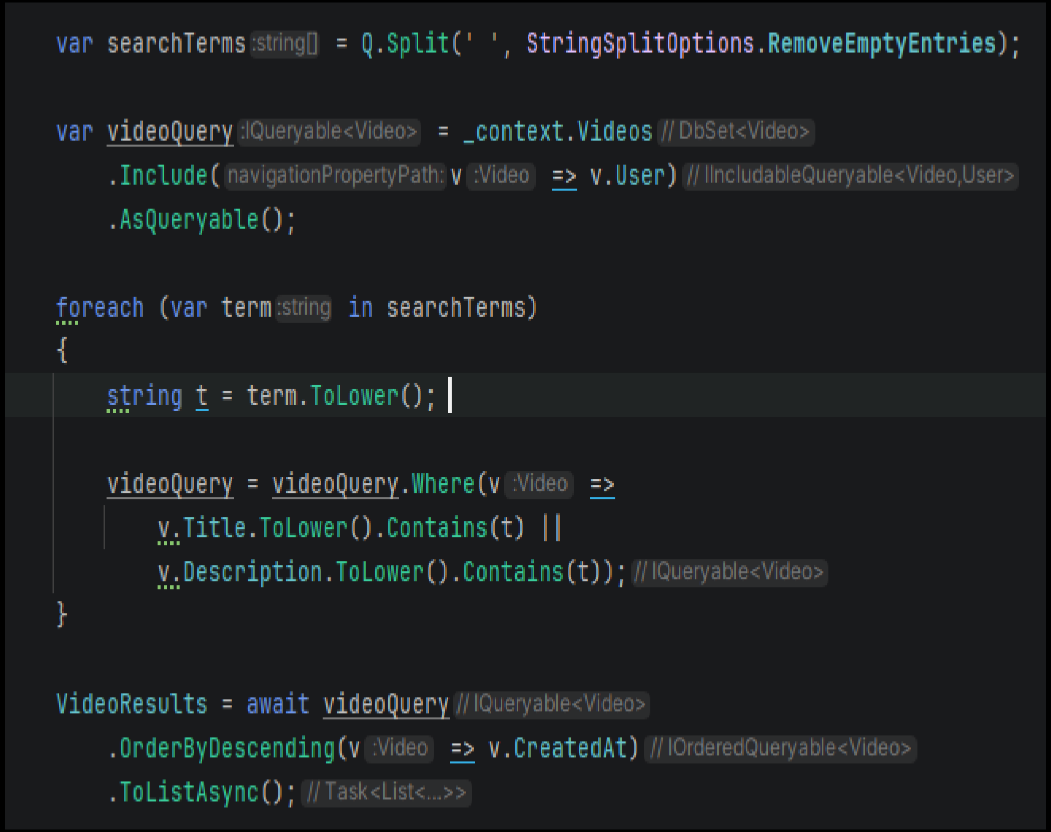
- Performance Bottleneck: The "Trending" logic requires sorting the entire database by Comments.Count (an expensive operation). We resolved this by implementing Look-Aside Caching (IMemoryCache), storing the heavy query result for 15 minutes to prevent DB saturation.
- Keyset Pagination: Replaced standard Skip/Take (Offset pagination) with Cursor-based pagination (Where VideoId < Cursor). This keeps query performance O(1) regardless of how deep a user scrolls.
- Next Page Detection: We fetch PageSize + 1 items (13 videos) to strictly determine if a "Next Page" exists without running a separate, expensive Count() query.



```
// Create a unique cache key based on the page cursor
string cacheKey = $"recent_videos_{cursor ?? 0}";

if (!_cache.TryGetValue(cacheKey, out List<Video> cachedRecent))
{
    int pageSize = 12;

    var query :IOrderedQueryable<Video> = _context.Videos // DbSet<Video>
        .Include( navigationPropertyPath: v :Video => v.User) // IIncludableQueryable<Video,User>
        .Where(v :Video => v.PrivacyStatus == PrivacyStatus.Public) // IQueryable<Video>
        .OrderByDescending(v :Video => v.VideoId);

    if (cursor.HasValue)
    {
        query = (IOrderedQueryable<Video>)query.Where(v :Video => v.VideoId < cursor.Value);
    }

    // Fetch 13 items (12 + 1 to detect next page)
    cachedRecent = await query.Take(pageSize + 1).ToListAsync();

    // Save to cache for 1 minute
    var cacheOptions = new MemoryCacheEntryOptions()
        .SetAbsoluteExpiration(TimeSpan.FromMinutes(1));

    _cache.Set(cacheKey, cachedRecent, cacheOptions);
}
```

CHALLENGES AND SOLUTIONS: QUERIES SEARCHBAR

- **Dual-Entity Querying:** The controller executes two distinct LINQ queries against the Videos (Title/Description) and Users (Username) tables, returning disjoint datasets to populate the "Videos" and "Creators" tabs independently
- **Multi-Word Tokenization:** Instead of a simple string match, the backend splits the user's input into tokens (e.g., ["asp.net", "tutorial"]). We dynamically chain LINQ .Where() clauses for each token, ensuring an "AND" operator that finds matches even if words are non-adjacent.



```
var searchTerms :string[] = Q.Split(' ', StringSplitOptions.RemoveEmptyEntries);

var videoQuery :IQueryable<Video> = _context.Videos // DbSet<Video>
    .Include( navigationPropertyPath: v :Video => v.User) // IIncludableQueryable<Video,User>
    .AsQueryable();

foreach (var term :string in searchTerms)
{
    string t = term.ToLower();

    videoQuery = videoQuery.Where(v :Video =>
        v.Title.ToLower().Contains(t) ||
        v.Description.ToLower().Contains(t)); // IQueryable<Video>
}

VideoResults = await videoQuery // IQueryable<Video>
    .OrderByDescending(v :Video => v.CreatedAt) // IOrderedQueryable<Video>
    .ToListAsync(); // Task<List<...>>
```

WHAT WE LEARNED: FFMPEG LIBRARY

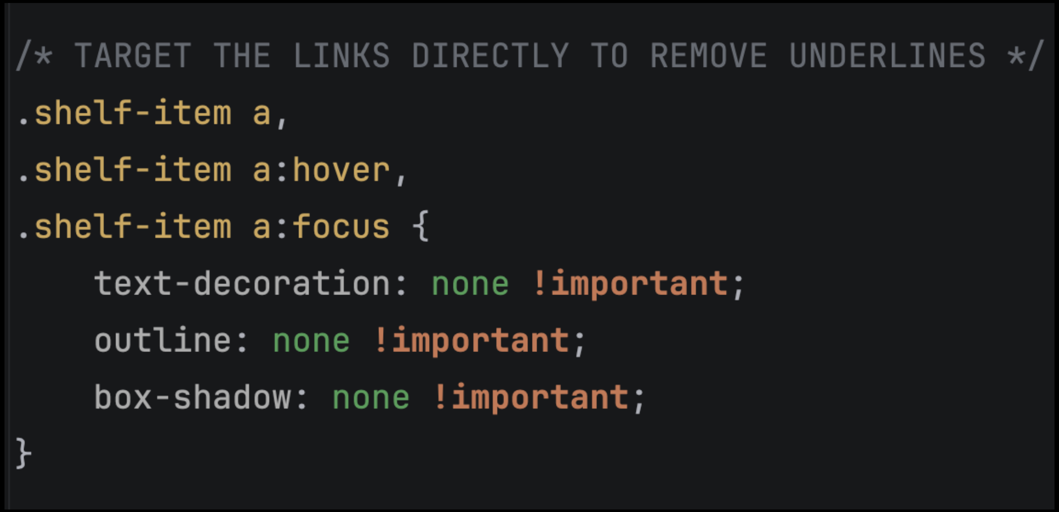
- **Asset Generation:** FFmpeg is used server-side to generate specific assets that HTML5 cannot create: a .jpg thumbnail, a .gif preview, and a compressed .mp3 audio track for AI processing.
- **Zero-Latency Previews:** The "Hover to Preview" feature works by pre-generating a GIF at upload time. The frontend simply swaps the <img> src attribute with the data-preview URL on mouseover, requiring no video buffering or loading during user interaction

WHAT WE LEARNED: AI PIPELINE

- Payload Optimization: We use FFmpeg to strip video data and extract a mono MP3 track before API calls. This reduces the data payload by ~90%, significantly lowering latency and bandwidth costs for the transcription service.
- Write-Once, Read-Many: The summarization pipeline (Audio -> Groq Transcription (Whisker v3 Large) -> Llama 3.3 Summary) runs strictly asynchronously during upload. The result is persisted to the SQL Videos table, ensuring the expensive AI operation never runs during page loads.

WHAT WE LEARNED: CSS ISOLATION

- Not as easy as we thought.
- Confusing hierarchy: Styles that are inside the .cshtml file have priority over all other styles but if you move the same block to it's own file (example: Index.cshtml.css) then certain elements prioritize Bootstrap style.



```css
/* TARGET THE LINKS DIRECTLY TO REMOVE UNDERLINES */
.shelf-item a,
.shelf-item a:hover,
.shelf-item a:focus {
    text-decoration: none !important;
    outline: none !important;
    box-shadow: none !important;
}
```

WHAT WE LEARNED - ADS

- Ads served with Google’s Interactive Media Ads SDK (IMA SDK)
- Checks if google is blocked so video can still play
- Sets a timer so video will play with or without ad
- Any errors force play the main video

FUTURE WORK

- Different pricing tires for subscription models
- Email notifications for when a new video is uploaded by a channel you follow
- Better interface for filtering videos on channels you subscribe to
- Ability to add keywords or categories to categorize videos
- Expanded thumbnail creation options
- Adding closed captions to videos

SUMMARY - YOUTUBE CLONE WEBSITE WITH THE FOLLOWING FEATURES:

- Ability to upload and play videos
- Creators can make custom playlist of their own content
- Users have watch and like histories (Logged in users only)
- Ability to comment, like, and dislike videos (must be logged in)
- Videos are processed when uploaded to provide meta-data, thumbnails, and short previews
- AI generated summaries based on audio from the uploaded file
- Ads served by Google
- Relief from ads by using Stripe to pay for premium
- Video search feature
- Email notifications

DISTRIBUTION OF WORK

Minh
- AI Summary
- Video processor
- Navbar
- Setup Database
- Upload page
- Setup the ads

Jonathan
- Stripe payment
- Watch history
- Liked videos
- Video uploading
- Email setup
- Initial setup (hosting and storage)

Elizabeth
- Profile page
- Home page
- View page
- Google authentication
- Channel page (videos and playlists)
- UI harmonization