Web Application in Angular and Ionic

Mathias Rahikainen

Bachelor's thesis for Business Information Technology

Raseborg 2021

EXAMENSARBETE

Författare: Mathias Rahikainen

Utbildning och ort: Informationsbehandling, Raseborg

Handledare: Rolf Gammals

Titel: Webbapplikation i Angular och Ionic

Datum 19.09.2021 Sidantal 28 Bilagor 7

Abstrakt

Den här studien genomfördes för att ge användarna på YouTube en applikation där de enkelt kan iaktta plattformens videon tillsammans i real tid. Att försöka uppnå det här målet genom att som användare; spela och pausa videon samtidigt på YouTube.com är långt ifrån idealt.

Studiens mål uppnåddes genom att planera och programmera en webbapplikation som kan spela upp Youtube videon. Applikationen måste ha tillgång till samma videon som är tillgängliga på Youtube. Det skall också vara lätt att iaktta videon tillsammans. Applikationen uppnår det här genom att ha kanaler där en värd sköter alla handlingar det vill säga, sökandet, uppspelandet, pausande, spolande fram och bak. Det här skapar en miljö där kanal gästerna inte behöver göra något utan de kan enkelt iaktta exakt samma sak som värden.

Programmeringsspråket som valdes i den här studien är TypeScript. Applikationen använder ett SPA ramverk som heter Angular för att göra utveckling lättare. För att vidare förenkla utvecklingen används Ionic med sina färdiga komponenter.

Servern som hanterar all data som webbapplikationen sedan synkroniserar är ett Firebase projekt. Servern använder Firebase Realtime database för att snabbt kunna läsa och skriva data.

De specificerade målen med den här studien uppnåddes i slutet.

Språk: Svenska Nyckelord: TypeScript, Youtube, Firbase, Ionic

BACHELOR'S THESIS

Author: Mathias Rahikainen

Degree Programme: Business Information technology, Raseborg

Supervisor(s): Rolf Gammals

Title: Web application in Angular and Ionic

Date 19.09.2021 Number of pages 28 Appendices 7

Abstract

This study was done to give people an application where they could easily watch YouTube videos together. Trying to achieve that by starting and pausing videos simultaneously on YouTube.com is far from convenient.

The studies goal was achieved by planning and coding a web application that could play YouTube videos. The application had to have the same videos that are accessible on YouTube. It also had to make it easy to watch them together. The application achieves this by creating channels where the host will be in charge of searching, playing, pausing, fast forwarding and rewinding the videos. This creates a setting where the channel guests will just lay back and watch the same thing that the host is watching and controlling.

The programming language that was chosen for this study is TypeScript. The application uses a SPA framework called Angular to make it easier to develop. To further increase the simplicity, it uses Ionic with its pre-styled components.

The server that handles all the data that the web application then syncs together is a Firebase project. It uses the Firebase Realtime database for quick reads and writes.

The specified goals of this study were achieved in the end.

Language: English Key words: TypeScript, YouTube, Ionic, Firebase

---

Table of contents

List of abbreviations

REST	Representational State Transfer.
API	Application Programming Interface.
JSON	JavaScript Object Notation.
NULL	A special keyword indicating that something has no value.
CSS	Cascading Style Sheets.
HTTP	HyperText Transfer Protocol, is an application protocol used most of the time for transferring web pages over the internet.
URL	Uniform Resource Locator, An address of a unique resource on the web.
HTML	HyperText Markup Language, is a standardized system to transfer information of web pages across the internet.
URI	Uniform Resource Identifier is a string of characters that unambiguously identifies a particular resource.
IDE	An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development.
SPA	Single Page Application, is a web application that interacts with its user by dynamically rewriting the web page with new data from the server.
SQL	Structured Query Language, lets you access and manipulate databases.
NoSQL	Not only SQL, is an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases.
WebGL	Web Graphics Library, a JavaScript API for rendering interactive 2D and 3D graphics.

1 INTRODUCTION

This thesis was done to enhance the experience of watching YouTube videos together virtually. The idea of watching things together virtually came up during the COVID-19 pandemic and the restrictions that came with it. Since then, numerous platforms have adjusted their applications to include similar functionalities. This is a strong indication that the initial concept was good and that it could have had financial value if executed properly.

In the theoretical part of this thesis the technologies that were used to develop this application are covered as well as other technical challenges. The empirical part will cover how exactly I used the listed technologies to implement the application and how I planned it.

1.1 Background

We choose to be social; we choose to do things together. It satisfies the human need to belong, research suggests (Jolly, Tamir, Burum and Mitchell 2019). Having fun is also more enjoyable if it is shared rather than solitary fun, particularly if the sharing involves a friend (Reis, O'Keefe and Lane 2017). COVID-19 has made it increasingly more difficult to be social and share experiences, due to the social distancing rules and recommendations all over the world.

1.2 Objective

This study aims to create an application with a virtual social setting where the user is able to watch videos together with friends and people all over the world. This while still being able to distance oneself physically and follow social distancing recommendations. By doing so the goal is to bring social enjoyment to the users of this application.

The main objective of this thesis was to create an application that would solve the displeasure of watching YouTube videos simultaneously together with a group of people over the internet. The annoyance lies in the lack of a feature that would allow you to start, pause and jump to specific timelines of a said video with a group of people. This application solves this by creating an environment where you have access to all these features while still being able to watch the same video that one could watch on YouTube.

2 THEORETICAL BACKGROUND

This chapter will cover the theoretical details of this thesis project. It will include information about all the technologies used in the final application. It will also cover why these technologies were chosen for this project.

2.1 Technologies

The main technologies used for building this study's application will be covered in this chapter.

2.1.1 JavaScript

JavaScript is a programming language that lets programmers implement dynamically updating web pages. JavaScript can be executed in a JavaScript engine, initially only browsers had these engines but nowadays there are runtime systems like Node.js and Deno that are capable of running JavaScript on the servers. (Mozilla 2020)

Back in 1995 Brendan Eich created the first version of JavaScript in only ten days. Initially Eich wanted to put a programming language named Scheme in the browser. However, instead of doing so, he was asked to add a programming language similar to Java. Eich eventually came up with a programming language that combined the capabilities of Scheme with the prototype-based object-orientation of the computer language Self, as well as Java-like syntax. Brendan's initial implementation was a basic interpreter for JavaScript, this later on turned into the SpiderMonkey engine, that is still used by the Firefox browser today (Brendan 2017).

So, what exactly can JavaScript do? It can store valuable information in variables, perform operations on text strings, run specific code in response to events, and much more, just like most other programming languages.

JavaScript can access the browser's API, which exposes data from the computer's environments. It has access to the Document Object Model API, which allows you to dynamically create, change, and remove HTML and CSS elements. It also includes Canvas and WebGL APIs, allowing programmers to deal with 2D and 3D graphics. (Mozilla 2020)

2.1.2 TypeScript

The programming language TypeScript is a strict syntactical superset of JavaScript. It primarily offers static typing, tuples, interfaces, and classes. All JavaScript programs are also valid TypeScript programs since TypeScript is a superset of JavaScript. It is an open-source programming language that is developed and maintained by Microsoft (MicroSoft 2020). The lead architect behind TypeScript is Anders Hejlsberg whom is also the original designer behind the Microsoft programming language C# (GitHub 2020). Hejlsberg was the first one to announce the new Microsoft project TypeScript (Hejlsberg 2012).

The main benefit from adopting TypeScript into a JavaScript project is that it might result in a more robust software. A study showed that 15% of all JavaScript bugs can be detected by TypeScript (Gao, Bird and Barr 2017). TypeScript is also great for developer teams since it allows the programmers to expose their APIs with typed interfaces. This means that if a programmer uses an IDE that supports intellisens, he or she will not need to memorize all of the code and may instead rely on the IDE. (MicroSoft 2020).

Why was TypeScript chosen as the application's primary programming language? One of the key reasons for this is that TypeScript is set as a default option in Angular. (Google 2020). And, as previously stated, TypeScript helps to prevent bugs, and when compared to simple JavaScript, your IDE provides excellent support to you as a programmer. (Gao, Bird and Barr 2017). Another great advantage of TypeScript is that you can use newer features of JavaScript that are not yet supported by the browser but are supported by the TypeScript compiler (MicroSoft 2020).

2.1.3 Angular

The Angular team at Google produced Angular, an open-source web application framework built on TypeScript. (Google 2020). A web application framework is a software framework that is designed to aid in the development of a web application. It provides a standard method for developing and deploying web applications on the internet. (Intelegain 2019).

Angular is specifically a framework for building single-page client applications by using HTML, CSS and TypeScript. The basic building blocks of angular are *NgModules*, which provide a compilation context for components. An Angular app always has at least a root module that enables the bootstrapping of the application (Google 2020).

Components in Angular are classes that use TypeScript decorators. These decorators mark their type and provide metadata that tells Angular how to use them. The metadata consists of an association with a template that defines the view for that component. A template in Angular combines ordinary HTML with Angular directives and binding markup that allow programmers to modify the HTML before Angular renders it for display (Google 2020).

There are also services in Angular. A service is also a TypeScript class that uses a decorator. The metadata for a service is different though, it provides the information Angular requires making it available to components through dependency injection (Google 2020).

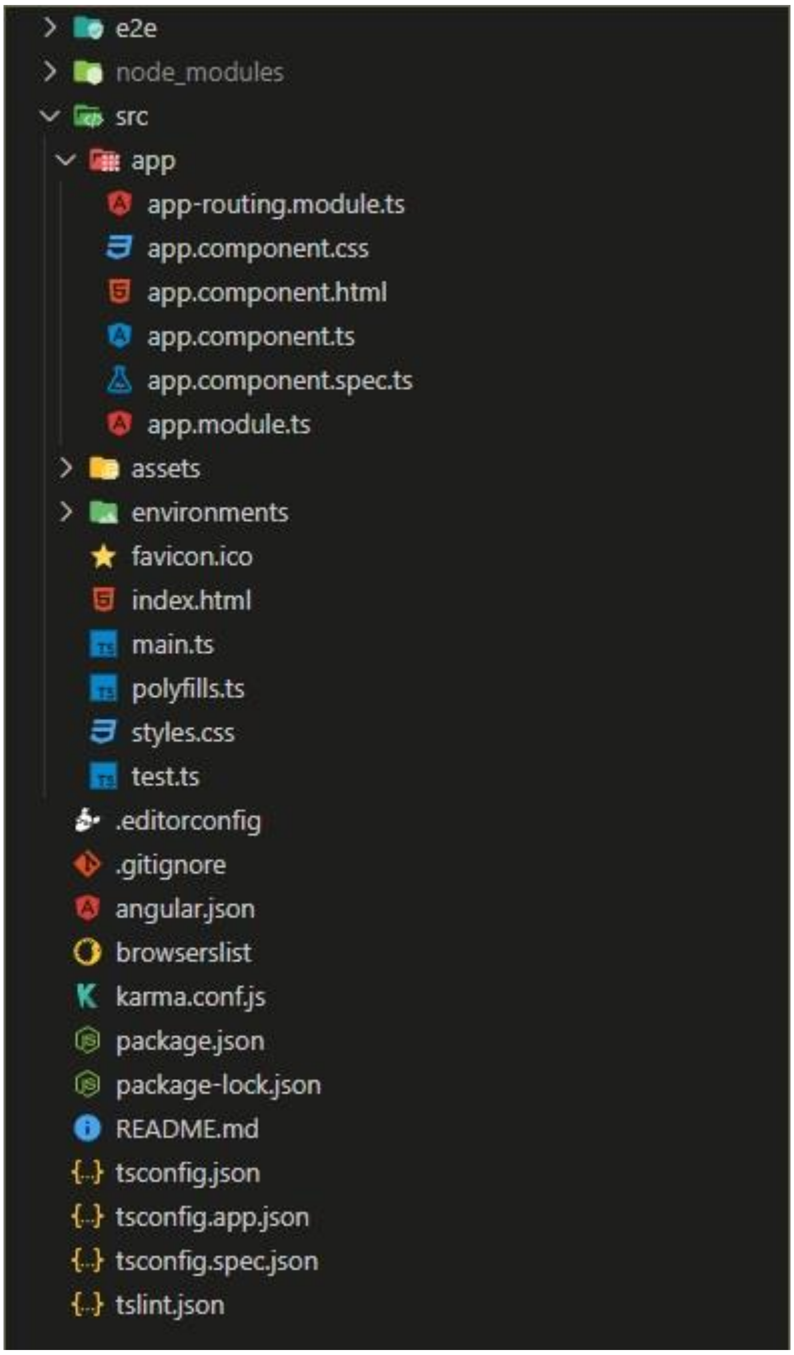
A powerful feature of Angular is the CLI tool. Programmers may use the CLI tool to initialize, develop, scaffold and maintain Angular projects directly from the command line (Google 2020). You can install the Angular CLI tool using the node package manager by executing:

npm install -g @angular/cli

When it is installed, the user has access to all the tools from Angular CLI. To list a few of the powerful commands you get with it:

- *ng serve* Builds the app and serves it locally on http://localhost:4200
- *ng generate* Can generate components, modules, pipes, services and more
- *ng build* Compiles an Angular app into an output directory named dist/
- *ng new* Creates a new workspace and an initial Angular app
- *ng test* Runs the unit tests in the Angular project
- *ng e2e* Builds and serves the Angular application then runs end-to-end tests using Protractor

The folder structure of a basic Angular project generated is shown in the Figure 1



```
> e2e
> node_modules
v src
  v app
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.ts
      app.component.spec.ts
      app.module.ts
  > assets
  > environments
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
  .editorconfig
  .gitignore
  angular.json
  browserslist
  karma.conf.js
  package.json
  package-lock.json
  README.md
  tsconfig.json
  tsconfig.app.json
  tsconfig.spec.json
  tslint.json
```

Figure 1. Angular Folder structure including files.

Why Angular? There are a lot of good options for JavaScript frameworks out there, to list a few famous ones other than Angular:

- React
- Vue
- Ember
- Svelte
- Backbone

Angular is still one of the most popular, and it comes with TypeScript. It performs well and is a fully built framework while some of these other frameworks lack out of the box features that Angular has. Because it is opinionated, it is ideal for independent programmers, the standard for designing Angular apps can be followed. (Google 2020). It would be easier for Angular developers to join in and start contributing if this project were ever open-sourced, as opposed to something that isn't opinionated, such as React. Before writing their own code, React developers would have to study the project's code structure and standards (React 2020).

2.1.4 RxJS

RxJS is the Reactive Extensions flavour for JavaScript and it is a library for reactive programming. It makes use of observables to make it easier to compose asynchronous code. In computing reactive programming is a declarative programming paradigm concerned with data streams and the propagation of change. This paradigm has the possibility to express static or dynamic data streams with ease (ReactiveX 2020).

RxJS is heavily integrated into Angular which makes this library a must, while developing applications with the Angular framework. The Angular HTTP client, for example, is used by the developer for making HTTP calls within an Angular project. As a result of HTTP method calls, the Angular HTTP client returns observables. This provides several advantages over promise-based HTTP APIs. For one the observables do not mutate the server response, some other advantages would be, the requests can be configured to get progress event updates and failed requests can be retried easily (Google 2020).

2.1.5 Ionic

Ionic is a developer friendly application platform for building cross-platform applications with one codebase, for any device, with the web included. It is an open-source framework that makes use of web components that can pair with any JavaScript framework including Angular. It makes it easier to build high-performance mobile and progressive web apps that work on any platform or device (Ionic 2020).

In this study's application, Ionic is used for its open-source UI toolkit that allows programmers to quickly build performant high-quality mobile and desktop apps using web technologies (Ionic 2020).

2.1.6 Firebase

In 2011, James Tamplin and Andrew Lee developed the Firebase platform. Back then they named it Envolve and it provided developers with an API that enables the integration of online chat functionality into their websites. After the release, Tamplin and Lee saw that instead of storing chat messages, their users were using their platform to sync application data such as game status in real time. They then decided to separate the chat system from the real-time architecture that powered it. After that they founded Firebase as a separate company in September 2011, it went public in April 2012. In October 2014 Firebase was acquired by Google and has been developed by them since then (Crunchbase 2020).

Firebase offers a broad application development platform. It does for example offer the Firebase Realtime Database that can store and sync application data in milliseconds. It does also come with Cloud storage, Hosting, User Authentication and more. Developers can improve application quality with the help of Crashlytics, performance monitoring and test lab also provided by Firebase (Google 2020).

3 Empirical

In this chapter of the thesis, I am going to conclude how I developed the application.

3.1 Requirements and system specificationtackv

To begin, an analysis was conducted to establish which characteristics the program should provide. The application feature list study yielded the following results:

- The user can create an account
- The user can sign in and out
- The user should have its own channel
- The user should be able to join other users' channels easily
- The users should be aware of the number of users in the current channel
- The user should have access to the same videos that he/she would have on YouTube
- The user should be able to start, paus, end, switch video and it should do the same for all guests simultaneously
- The user should be able to sign in with Google

After that the features were planned into Angular modules, components and services. The following list consist of those:

- A sign-in page
- A sign-up page
- An authentication service
- A Channel module containing the channel component
- A YouTube Data service using the YouTube data API
- A YouTube player service wrapping around the YouTube Framework API
- A Channel Data service connected to the Realtime database
- A YouTube player component displaying the iframe and player styles
- A presence service to keep track of the user state if they are online or offline

The following step was to examine the data model for the Firebase Realtime database. Since the Firebase Realtime database is a NoSQL database, the data must be denormalized in order for it to scale well with a large number of users.

3.2 Architecture

The architecture chapter will cover the architecture between the server and the client. In this thesis project, the backend is the Firebase platform with its services. The client then again is the web application built with Angular and Ionic. Firebase exposes many APIs that can be used in the client. Communication with the Firebase Realtime Database and Firebase Authentication are two of them.

When a new user wants to use the app, they must first register or sign-in anonymously. To register they can quickly do so by using the sign-in with Google button. Firebase Authentication service is what makes all this possible. The application is making use of the Authentication API provided by Firebase. On the client this logic is handled in the *auth.service.ts* Angular service.

When a guest user is receiving information about the currently playing video the *channel.service.ts* service implementation is looking in the Firebase Realtime database all the time for any changes. Such changes could for instance be that the video is paused, video has ended or just started playing. To keep this logic simple a RXJS Observable subscription is being used to listen for the changes.

If a host changes something in their channel the data will be updated in the Firebase Realtime database. However, before changes are done the Firebase Realtime Database will determine that the user has authority to do so. This process can be seen in *Figure 2* beneath.

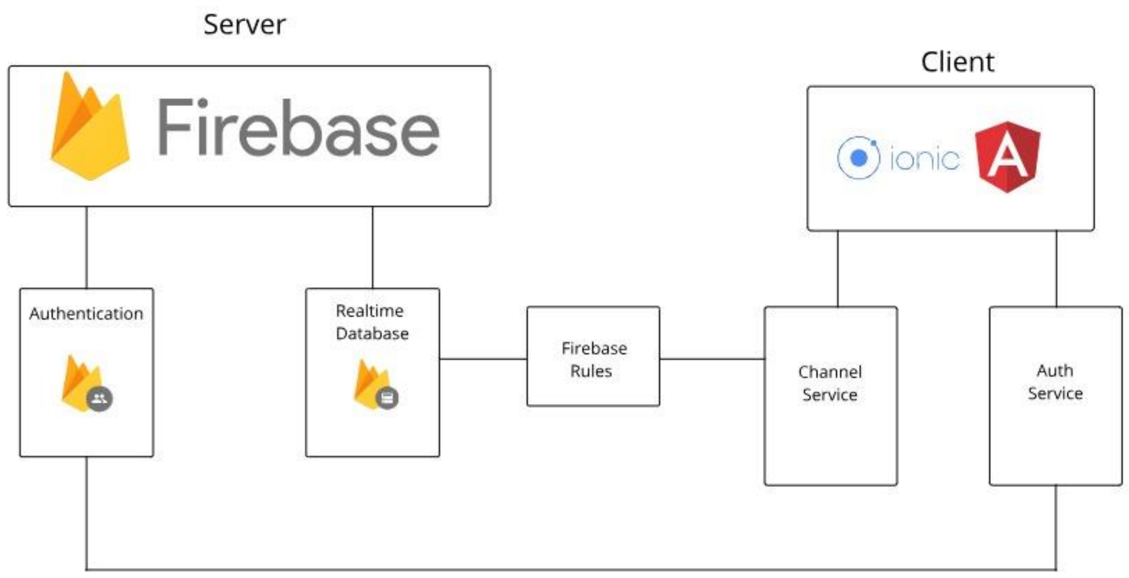


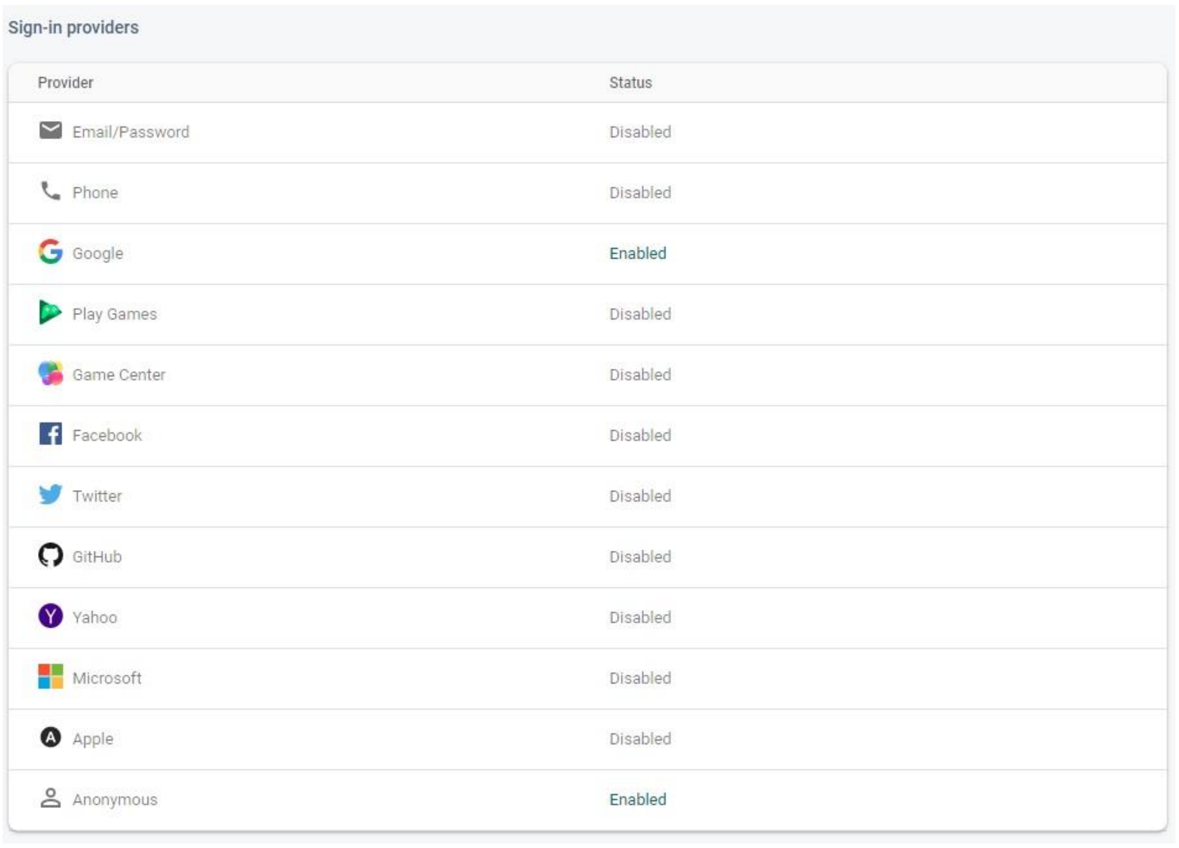
Figure 2. Application server architecture.

3.3 Implementation

This chapter will cover the coding of all Angular components and services. Angular components are building blocks of the application. Services are classes that can be used to handle logic. They can handle state management, data sharing, data transformation and much more.

3.3.1 Authentication

The authentication in this project is done by the Firebase Authentication service. Firebase easily lets you choose sign-in providers from its console. Which can be seen in *Figure 3*.



Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Enabled
Play Games	Disabled
Game Center	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Apple	Disabled
Anonymous	Enabled

Figure 3. Firebase authentication service providers

As displayed in the image the supported providers are Google and anonymous.

The user does not need to submit any additional information except a temporary username to sign in using the anonymous provider. The Google sign in must be used if a user wants a permanent account that can be signed into again.

The user needs an interface to provide this information to Firebase, which is provided by the Angular frontend client. The frontend client must also prevent users from viewing and using the application without first logging in. This logic begins in the *app.component.ts* in the lifecycle hook method *ngOnInit*. Within that method a service called *AuthService.ts* is subscribing to its class variable user which is an Observable. By doing so it initiates a check to the Firebase user's database table to see if the user is signed in. If the user is signed in, the application will automatically reroute to the channel page otherwise the user is asked to sign in through the sign in page. Sign-in page can be seen in *Figure 4*.

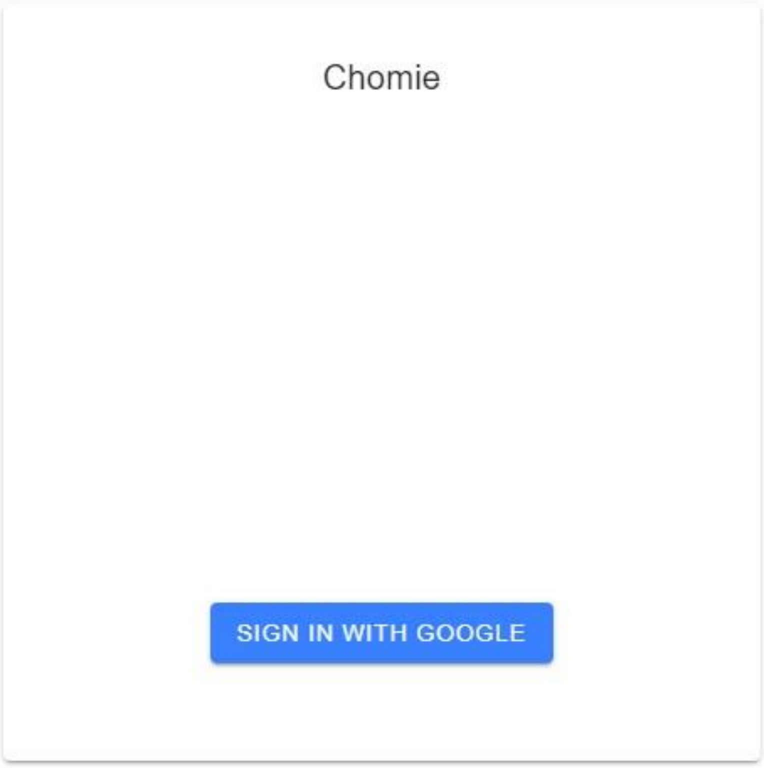
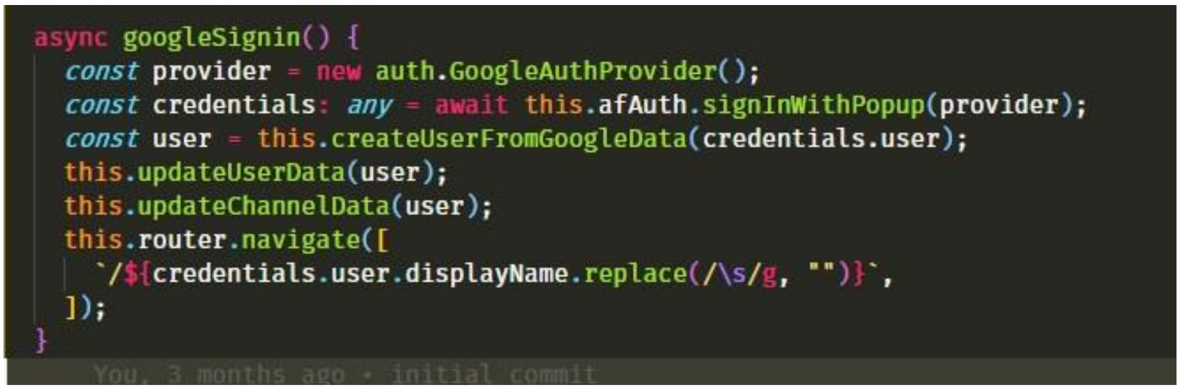


Figure 4. Application sign-in screen

This code snippet is executed when the user clicks the sign in button:

Code example 1. Google sign in with Firebase.



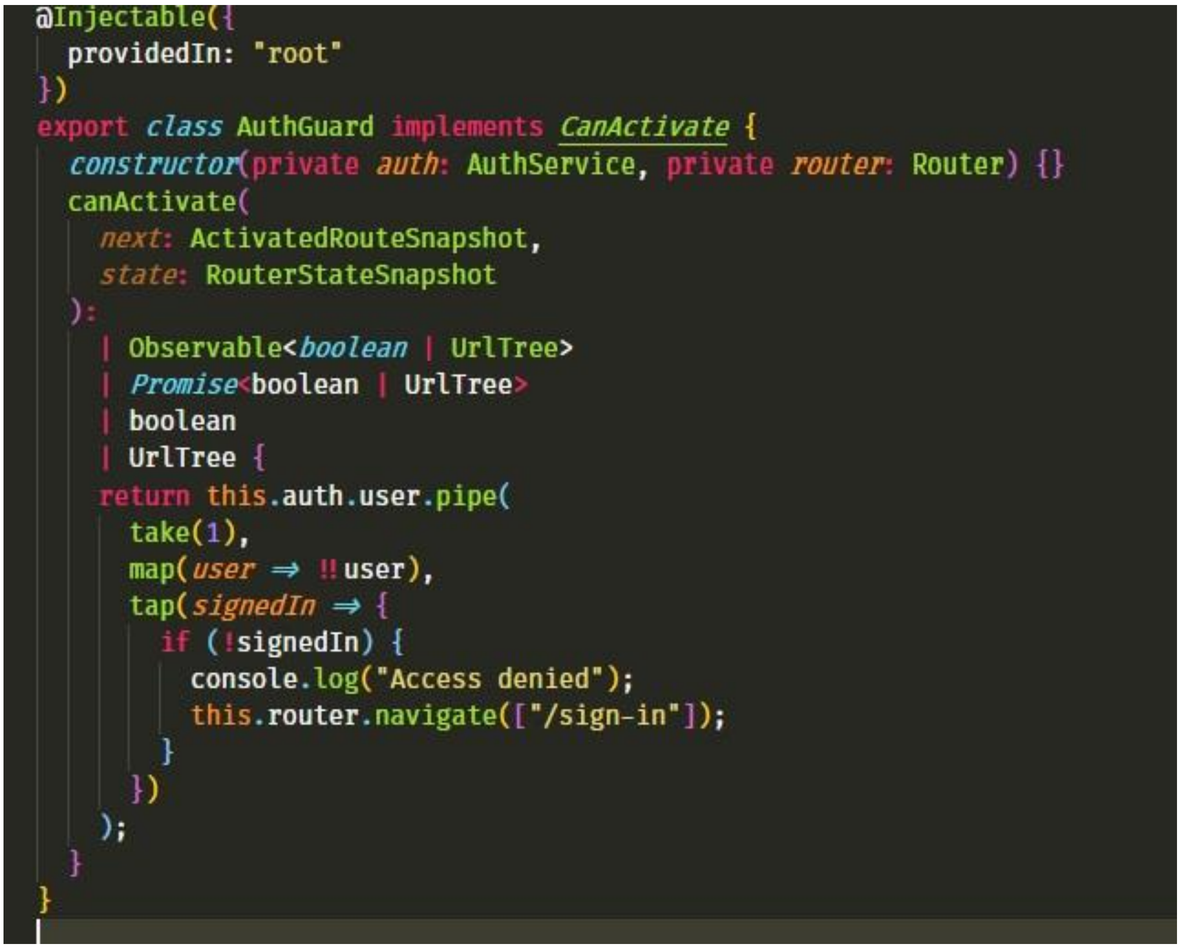
```
async googleSignin() {
  const provider = new auth.GoogleAuthProvider();
  const credentials: any = await this.afAuth.signInWithPopup(provider);
  const user = this.createUserFromGoogleData(credentials.user);
  this.updateUserData(user);
  this.updateChannelData(user);
  this.router.navigate([
    `/${credentials.user.displayName.replace(/\s/g, "")}`,
  ]);
}
    You, 3 months ago • initial commit
```

This method sets up a new Google authentication, then a popup window is displayed with the users currently signed Google accounts, if there is none it will ask the user to sign in to one. Otherwise, the user can just click on the desired account to sign in with it.

After the user has chosen the preferred account to sign in to, the User data table is updated in Firebase and then the application navigates to the users own channel. The username determines the URL for the user channel; effectively, the URL is just */username*.

All other routes except for the */sign-in* one are blocked by an Angular guard. The authentication guard in this project is called *auth.guard.ts*. Every time the application attempts to navigate to a protected route it verifies that the user is authenticated. If the user is not authenticated; it will not allow the navigation and the user will be redirected to the */sign-in* route. The following code is the *AuthGuard* implementation:

Code example 2. Angular guard protecting routes.



```
@Injectable({
  providedIn: "root"
})
export class AuthGuard implements CanActivate {
  constructor(private auth: AuthService, private router: Router) {}
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ):
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree>
    | boolean
    | UrlTree {
    return this.auth.user.pipe(
      take(1),
      map(user => !!user),
      tap(signedIn => {
        if (!signedIn) {
          console.log("Access denied");
          this.router.navigate(["/sign-in"]);
        }
      })
    );
  }
}
```

3.3.2 User presence

To make it easier for users to navigate to active channels this application has presence logic for the users. An *ion-menu* displays all users in a list with their presence indicated by both a text string and by colour. This menu can be seen in *Figure 5*.

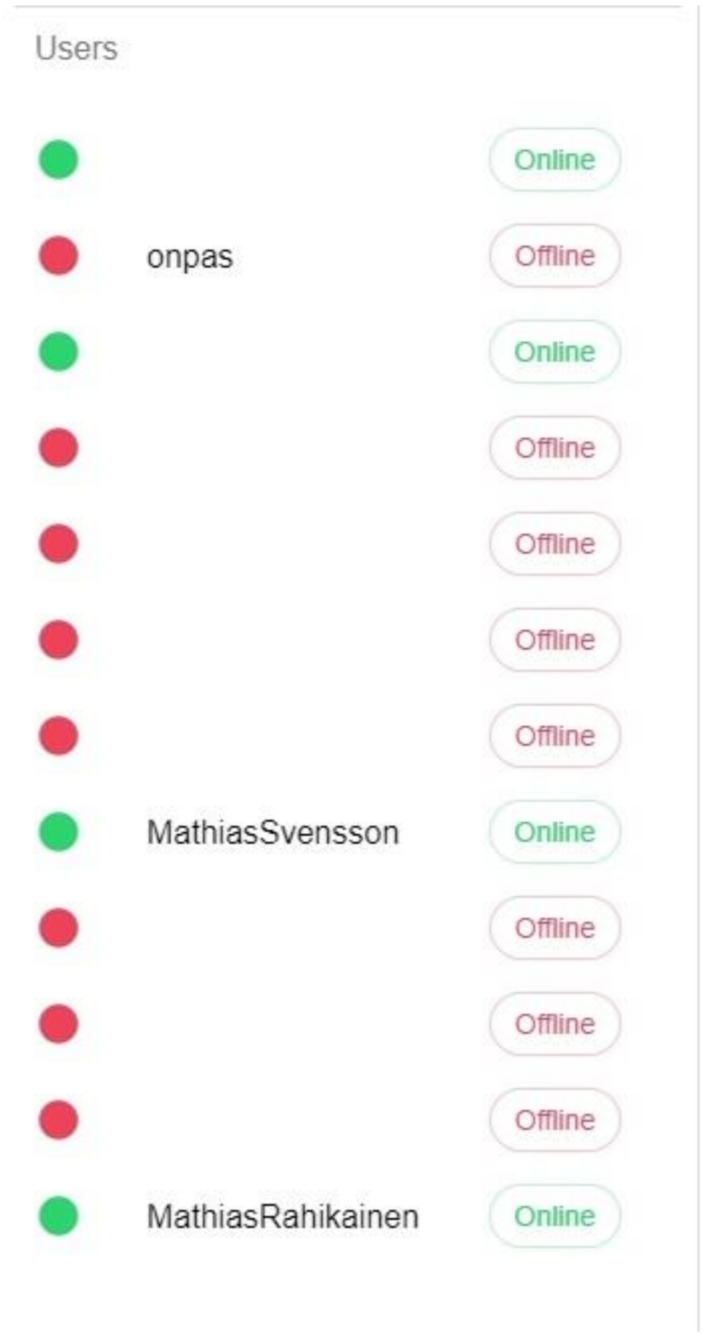
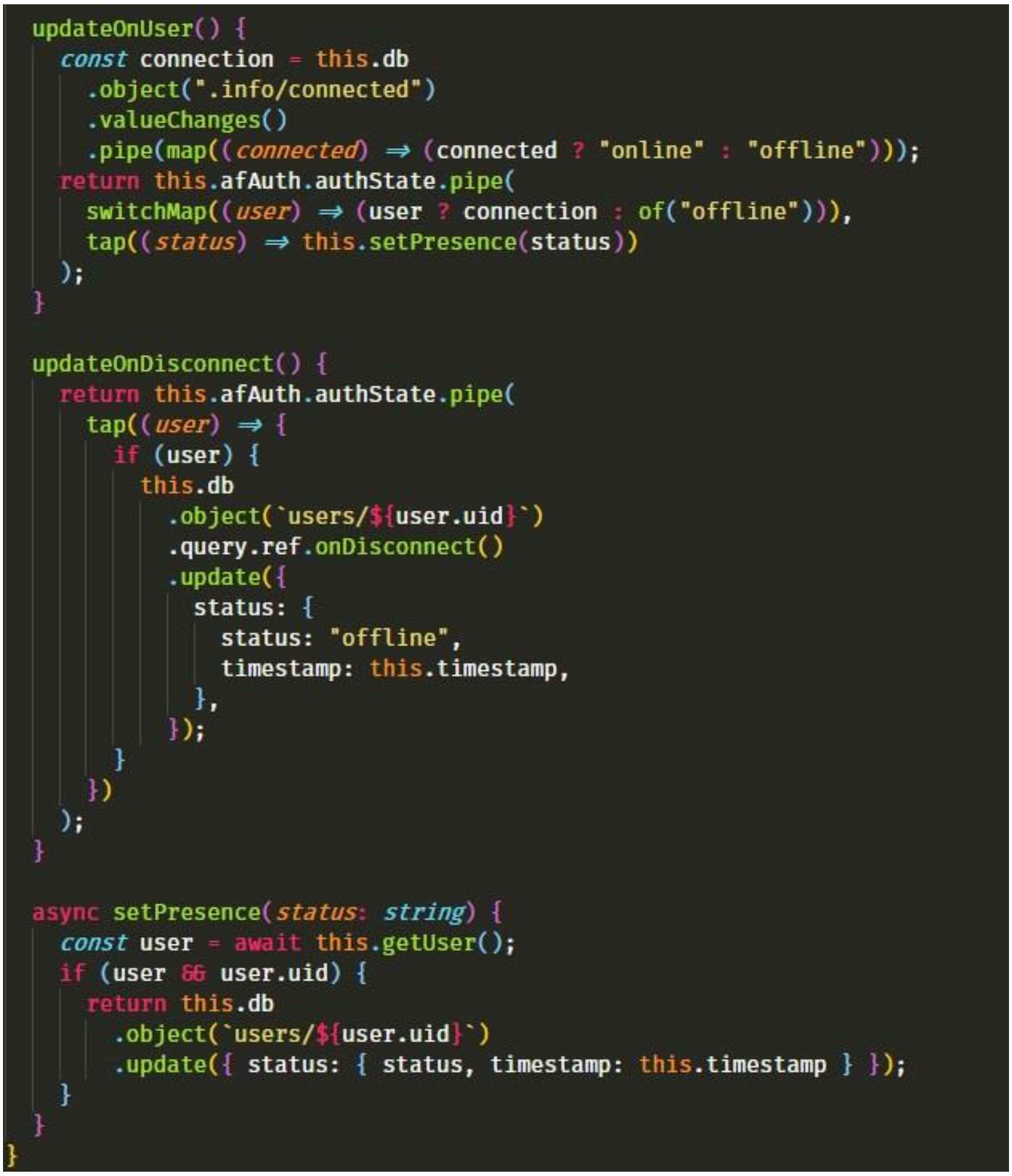


Figure 5. Application user list.

The Firebase Realtime database makes the user presence logic simple to implement. The database provides a special location -*.info/connected* where the presence information of the user is stored. Because of that information the presence logic could be implemented by the following three methods:

Code example 3. User activity state methods in Angular with RxJS.



```
  updateOnUser() {
    const connection = this.db
      .object(".info/connected")
      .valueChanges()
      .pipe(map((connected) => (connected ? "online" : "offline")));
    return this.afAuth.authState.pipe(
      switchMap((user) => (user ? connection : of("offline"))),
      tap((status) => this.setPresence(status))
    );
  }

  updateOnDisconnect() {
    return this.afAuth.authState.pipe(
      tap((user) => {
        if (user) {
          this.db
            .object(`users/${user.uid}`)
            .query.ref.onDisconnect()
            .update({
              status: {
                status: "offline",
                timestamp: this.timestamp,
              },
            });
        }
      })
    );
  }

  async setPresence(status: string) {
    const user = await this.getUser();
    if (user && user.uid) {
      return this.db
        .object(`users/${user.uid}`)
        .update({ status: { status, timestamp: this.timestamp } });
    }
  }
}
```

These three methods are all in the *presence.service.ts* service so that they can be injected to a root component in the application. This is essential so that the presence tracking starts as soon as possible. The main logic lies in the *updateOnUser()* method. It uses the Firebase

Realtime Database special location to check if a user is connected or not. If the user is online, it sets the status to online and vice versa.

3.3.3 Channel page

This application's main page is the Channel page. Users can watch videos together on this page. Each user has their own channel, that reflects their username.

The search bar can be found at the top of the page by its channel's host. It retrieves YouTube videos based on the provided keywords using the YouTube Data API.

Instead, if the user is a visitor, a text string notifies them that the host is looking for a video. Image of the search bar with items can be seen in *Figure 6*.

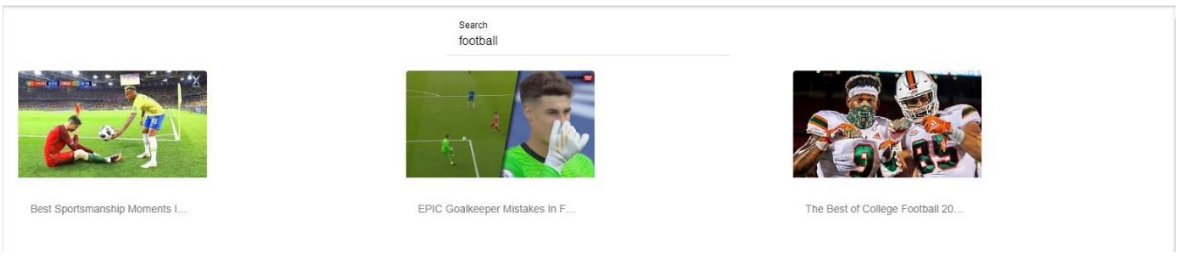
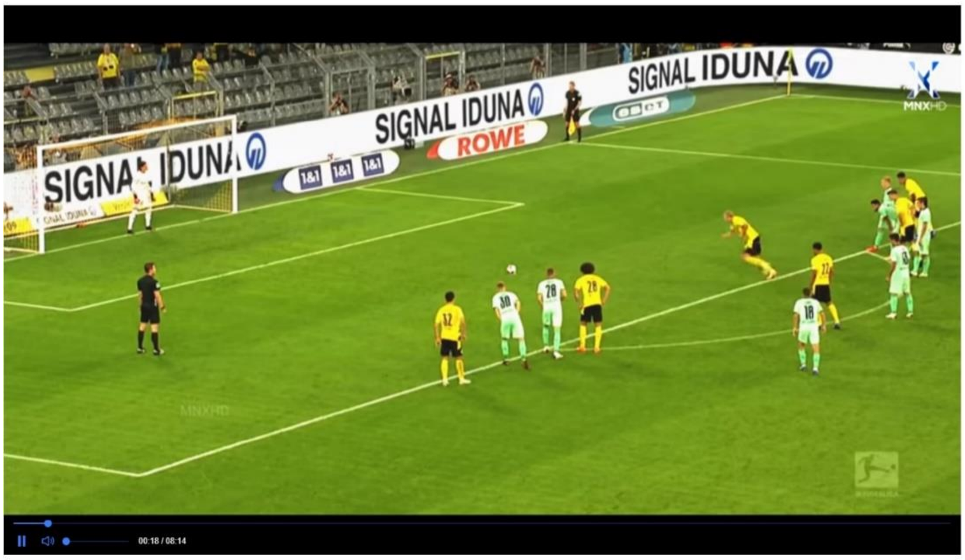

Search
football
Best Sportsmanship Moments I...
EPIC Goalkeeper Mistakes In F...
The Best of College Football 20...

Figure 6. Application channel page video list.

Below this the user will find the actual video player which consists of a YouTube Iframe with custom controls controlled by the Angular application. Image of the video player is presented in *Figure 7*.

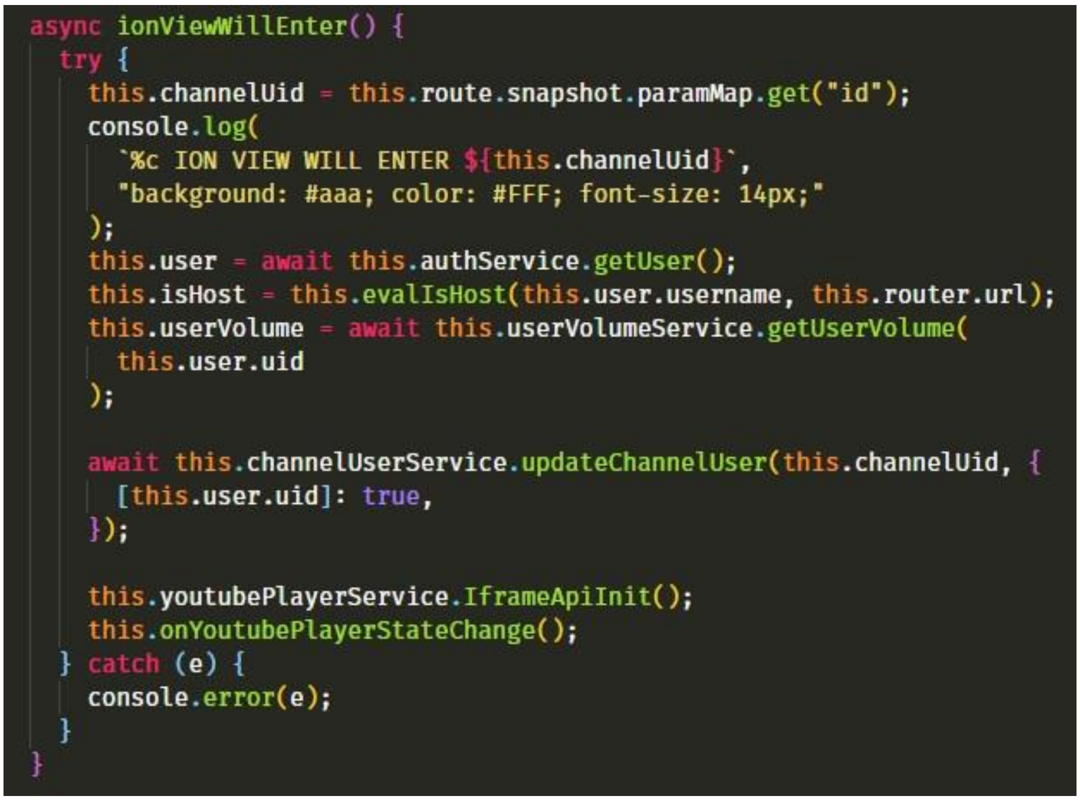
Figure 7. Application video player.



3.3.4 Initializing the channel component

When the application loads the channel component it sets up a lot of required class variables. In an Ionic application all component classes have access to a method called *ionViewWillEnter* which is a lifecycle method. When Ionic detects that the component is going to load, it calls this method.

Code example 4. Channel component initialization logic.



```
async ionViewWillEnter() {
  try {
    this.channelUid = this.route.snapshot.paramMap.get("id");
    console.log(
      `%c ION VIEW WILL ENTER ${this.channelUid}`,
      "background: #aaa; color: #FFF; font-size: 14px;"
    );
    this.user = await this.authService.getUser();
    this.isHost = this.evalIsHost(this.user.username, this.router.url);
    this.userVolume = await this.userVolumeService.getUserVolume(
      this.user.uid
    );

    await this.channelUserService.updateChannelUser(this.channelUid, {
      [this.user.uid]: true,
    });

    this.youtubePlayerService.IframeApiInit();
    this.onYoutubePlayerStateChange();
  } catch (e) {
    console.error(e);
  }
}
```

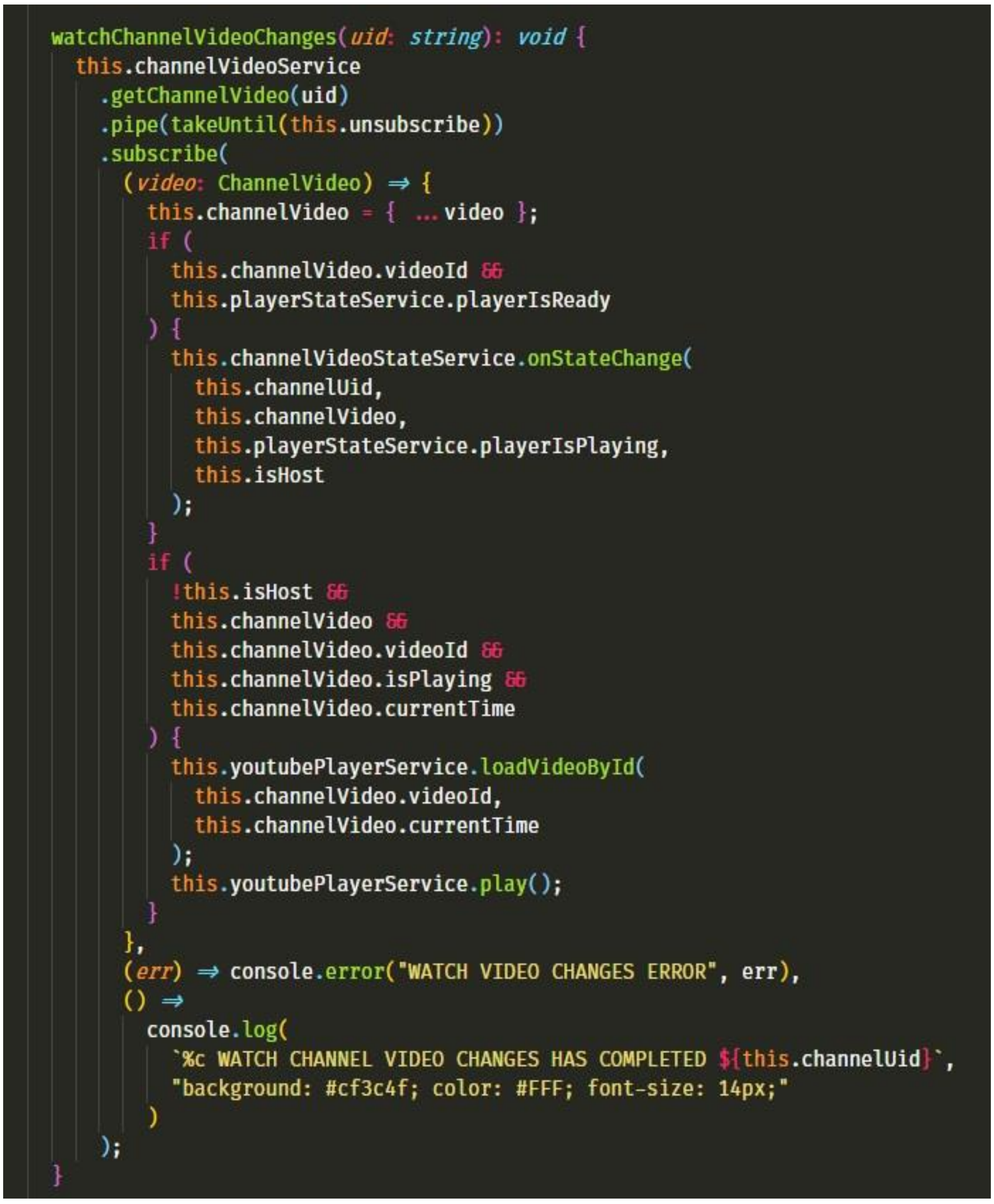
As seen in the figure, it begins by defining the class variable *channelUid*, which is simple to implement because the channel's unique id also serves as the route. It grabs the end of the URI to get the id. After that, it obtains the current user info. This information is needed to determine whether or not the user is the channel's host. Next, it obtains user volume, which is likewise saved in the database. This is an attempt to provide a better user experience by loading the same volume settings that the user had previously. Finally, it adds the new user to the channel in order to keep track of how many active users there are.

3.3.5 Syncing and handling videos

The project's major purpose was to make it possible to view YouTube videos with friends rather than alone. Because Firebase's live database responds so quickly to client requests, this is achievable.

When the application loads the channel component, it creates a number of listeners and determines if the current user is in his or her own channel, in which case he is the host, or not, in which case he is a guest.

Code example 5. Channel component start watching for video changes method.



```
watchChannelVideoChanges(uid: string): void {
  this.channelVideoService
    .getChannelVideo(uid)
    .pipe(takeUntil(this.unsubscribe))
    .subscribe(
      (video: ChannelVideo) => {
        this.channelVideo = { ...video };
        if (
          this.channelVideo.videoId &&
          this.playerStateService.playerIsReady
        ) {
          this.channelVideoStateService.onStateChange(
            this.channelUid,
            this.channelVideo,
            this.playerStateService.playerIsPlaying,
            this.isHost
          );
        }
        if (
          !this.isHost &&
          this.channelVideo &&
          this.channelVideo.videoId &&
          this.channelVideo.isPlaying &&
          this.channelVideo.currentTime
        ) {
          this.youtubePlayerService.loadVideoById(
            this.channelVideo.videoId,
            this.channelVideo.currentTime
          );
          this.youtubePlayerService.play();
        }
      },
      (err) => console.error("WATCH VIDEO CHANGES ERROR", err),
      () =>
        console.log(
          `%c WATCH CHANNEL VIDEO CHANGES HAS COMPLETED ${this.channelUid}`,
          "background: #cf3c4f; color: #FFF; font-size: 14px;"
        )
    );
}
```

In the above image the *channelVideo* is subscribed to. This subscription is key to this whole application. When the host starts playing a video this subscription updates all the users in the channel with the updated video status.

When the host, for example, clicks on one of the video items on the channel page, the program will automatically update all users by gathering the changes from the host first. The changes are then updated in the Firebase Realtime Database, and this listener is triggered to run as a result of that update. As a result, all users will receive the updated information, which in this case would be the right video with the video time set to 0 because the host had just started watching the video. If the host, for example, fast forwards to the middle of the video, updated video data will be delivered to the database, and all users will be provided with the new active time that the host has forwarded to.

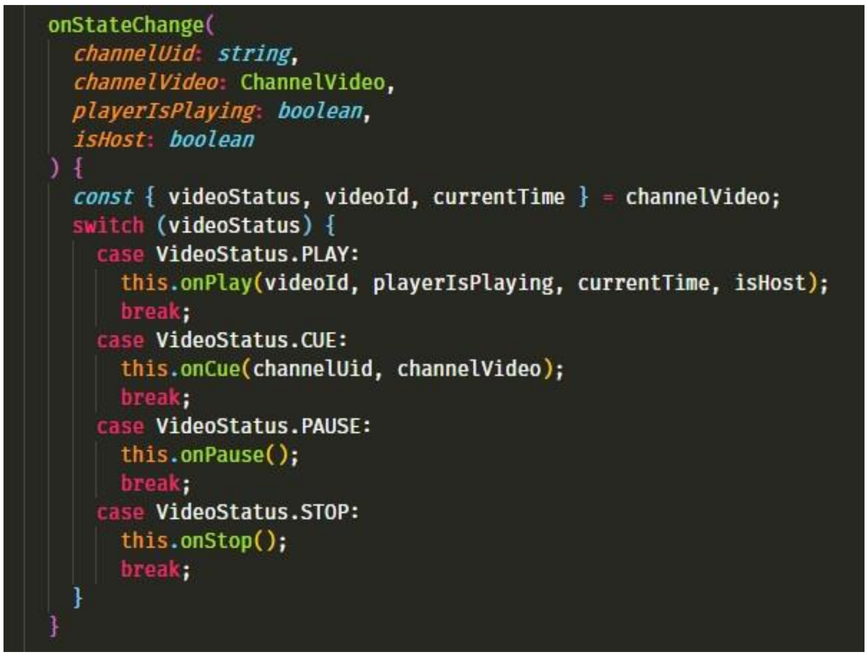
The video player makes use of the following events:

- IFRAME_READY: when the YouTube iframe has initialized
- READY: when the player is ready to receive videos and play them
- PLAYING: when the player is playing a video
- PAUSED: when the video is paused
- ENDED: when the video ended
- ERROR: when an error has occurred

An Angular service is listening to these events. The service is named *youtube-player.service.ts*. This service forwards this information to the *channel-video-state.service.ts*, which handles the change in player state and updates the database with the new information, on all event triggers. Some of these events, including IFRAME_READY and READY, are not kept in the database because they must always run and be true, therefore they are only stored locally.

These state changes are then handled by the *channel-video-state.service.ts*. The service handles these events by first determining what the new video status is in the following method:

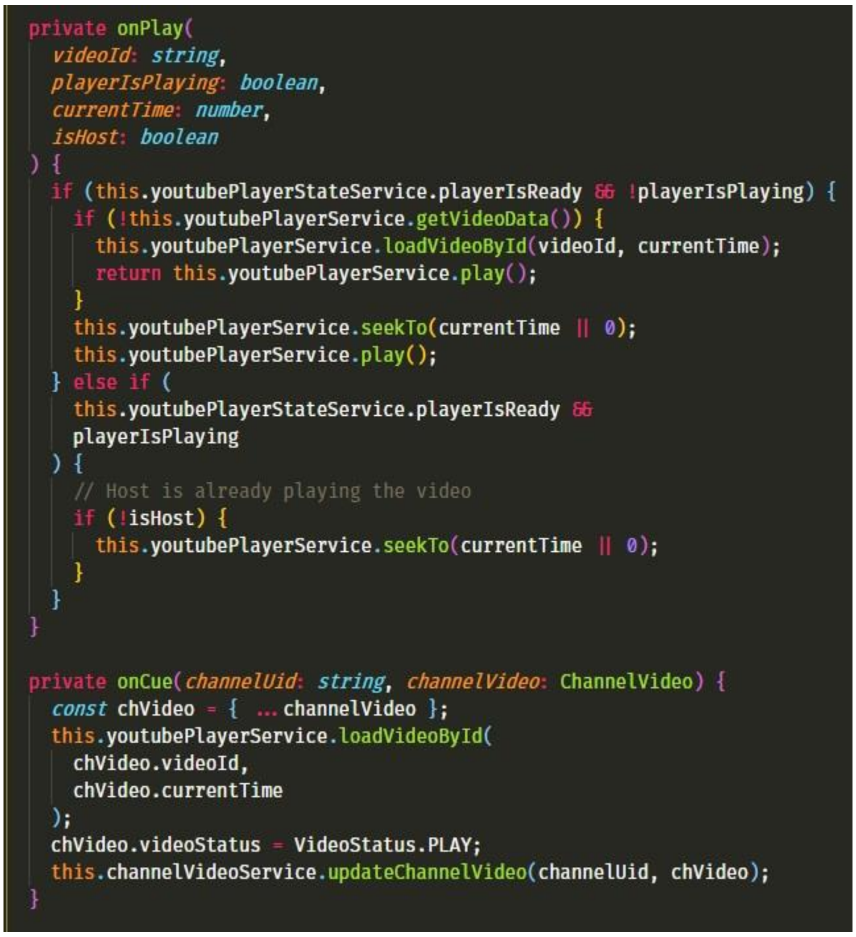
Code example 6. Video player service on state change method.



```
onStateChange(
  channelUid: string,
  channelVideo: ChannelVideo,
  playerIsPlaying: boolean,
  isHost: boolean
) {
  const { videoStatus, videoId, currentTime } = channelVideo;
  switch (videoStatus) {
    case VideoStatus.PLAY:
      this.onPlay(videoId, playerIsPlaying, currentTime, isHost);
      break;
    case VideoStatus.CUE:
      this.onCue(channelUid, channelVideo);
      break;
    case VideoStatus.PAUSE:
      this.onPause();
      break;
    case VideoStatus.STOP:
      this.onStop();
      break;
  }
}
```

And then after that the service handles the events accordingly, for example *onPlay* and *onCue* methods:

Code example 7. Channel component on play and on cue methods.



```
private onPlay(
  videoId: string,
  playerIsPlaying: boolean,
  currentTime: number,
  isHost: boolean
) {
  if (this.youtubePlayerStateService.playerIsReady && !playerIsPlaying) {
    if (!this.youtubePlayerService.getVideoData()) {
      this.youtubePlayerService.loadVideoById(videoId, currentTime);
      return this.youtubePlayerService.play();
    }
    this.youtubePlayerService.seekTo(currentTime || 0);
    this.youtubePlayerService.play();
  } else if (
    this.youtubePlayerStateService.playerIsReady &&
    playerIsPlaying
  ) {
    // Host is already playing the video
    if (!isHost) {
      this.youtubePlayerService.seekTo(currentTime || 0);
    }
  }
}

private onCue(channelUid: string, channelVideo: ChannelVideo) {
  const chVideo = { ...channelVideo };
  this.youtubePlayerService.loadVideoById(
    chVideo.videoId,
    chVideo.currentTime
  );
  chVideo.videoStatus = VideoStatus.PLAY;
  this.channelVideoService.updateChannelVideo(channelUid, chVideo);
}
```

In the *onPlay* method we are confirming that the player is ready and thereafter it attempts to play the video.

As we can see with the *onCue* method it receives the selected video and the channel unique id as parameters. Before the data is stored the *loadVideoById* method is invoked which will attempt to play the video. The data allows the users to receive the video id and the video position.

4 Conclusion

In this study I have developed a web application using TypeScript and the Angular framework. With Angular I used the Ionic-Angular API to achieve my goals. The backend is a Firebase project which handles all the data with the help of Firebase Realtime Database service. It also handles the sign-in methods with the Firebase Authentication service.

The goal of this application was to allow users to watch YouTube videos together online in a virtual environment. I think this goal has been met and this is possible with the current application. A user is able to fast forward, pause and rewind videos. The videos are fetched from YouTube API V3 data which means that it is the same videos that you can find on Youtube.com.

There is still a lot of functionality that could be added to this application to enhance the user experience. And there are also some bugs in the application however I still think this application was a success.

4.1 Discussion

This project was quite challenging but came with a lot of knowledge. It started out as an idea when COVID-19 restrictions where quite strict and similar functionalities where not common. A lot has happened since then and now platforms such as Spotify, Disney+, Facebook and more offer similar features.

The practical part of this project started with figuring out the architecture needed. Several problems occurred during the development of this application. There where problems with limited rights on styling the YouTube Iframe. This application needs its own custom handles for the player to work as intended. Then there were also problems with hard reloading the channel component. The Iframe would not re-initialize properly while switching between user channels.

In the end I managed to solve most problems, but the project was greatly delayed due to them. I would still conclude that the project was a success, and the application now works as intended.

References

Baumeister R. F., & Leary M. R. (1995). *The need to belong: desire for interpersonal attachments as a fundamental human motivation*. Psychological Bulletin, 117(3), 497 Retrieved from https://pubmed.ncbi.nlm.nih.gov/7777651/

Brendan, E. (2017). *A Brief History of JavaScript* Retrieved from https://www.youtube.com/watch?v=GxouWy-ZE80

Crunchbase. (2020). Retrieved from https://www.crunchbase.com/organization/firebase

React. (2020). Retrieved from https://reactjs.org/

Gao, Z., Bird, C., & Barr, E. (2017). *To type or not to type: quantifying detectable bugs in JavaScript. In Proceedings of the 39th International Conference on Software Engineering* (ICSE '17). IEEE Press, 758–769. DOI:https://doi.org/10.1109/ICSE.2017.75 Retrieved from http://earlbarr.com/publications/typestudy.pdf

Angular. (2020). Retrieved from https://angular.io/

Google Firebase. (2020). Retrieved from https://firebase.google.com/

Hejlsberg, A. MicroSoft (2020). Profile [Profile bio] GitHub Retrieved from https://github.com/ahejlsberg

Hejlsberg, A. (2012). *Introducing TypeScript* [Video online]. USA: MicroSoft Retrieved from https://channel9.msdn.com/posts/Anders-Hejlsberg-Introducing-TypeScript/player

Intelegain. (2019). *What are web frameworks and why you need them?* Retrieved from https://medium.com/@intelegain_inc/what-are-web-frameworks-and-why-you-need-them-c4e8806bd0fb

Ionic. (2020). Retrieved from https://ionicframework.com/

Jolly, E., Tamir, D. I., Burum, B., & Mitchell, J. P. (2019). *Wanting without enjoying: The social value of sharing experiences*. *PloS one*, *14*(4), e0215318. https://doi.org/10.1371/journal.pone.0215318, Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6472755/

Mozilla. (2020). Retrieved from https://developer.mozilla.org/

TypeScript. (2020). *Typed JavaScript at Any Scale.* [Online] https://www.typescriptlang.org/, [Retrieved: 24.08.2020]

ReactiveX. (2020). Retrieved from https://rxjs-dev.firebaseapp.com/

Reis, H. T., O'Keefe, S. D., & Lane, R. D. (2017). *Fun Is More Fun When Others Are Involved.* The journal of positive psychology, *12*(6), 547–557. https://doi.org/10.1080/17439760.2016.1221123, Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5597001/