

Tutor

[Document subtitle]



[Date]

[Company name]

[Company address]

# Abstract

This tutor app allows users to access educational videos from YouTube, whose links are downloaded from Firebase. The app features a simple user interface and navigation, with video playback controls and the ability to choose courses manually or automatically by giving answers to some questions. For the users who have internet problems, we save the video in the cache memory that users can see in case of internet fluctuation. The app aims to provide a convenient and accessible way for students to access educational content at their own pace and on their own schedule. This app is free of cost and anyone can use this app.

Table of Contents

[Abstract 1](#_Toc128395238)

[Introduction 3](#_Toc128395239)

[Road Map 3](#_Toc128395240)

[Agile Method 5](#_Toc128395241)

[Scenarios 10](#_Toc128395242)

[Key Features of our application 11](#_Toc128395243)

[User stories 12](#_Toc128395244)

[Development Stack 13](#_Toc128395245)

[Application Infrastructure 14](#_Toc128395246)

[App Architecture 15](#_Toc128395247)

[Sequence Diagram 17](#_Toc128395248)

[Data model 18](#_Toc128395249)

[Use case 18](#_Toc128395250)

[Process diagram 21](#_Toc128395251)

[Application diagram 22](#_Toc128395252)

[Privacy 24](#_Toc128395253)

[Branching Strategy 27](#_Toc128395254)

[Deployment Strategy 28](#_Toc128395255)

[Requirements 29](#_Toc128395256)

[Methodology 30](#_Toc128395257)

[Feedback: 46](#_Toc128395258)

[Reference: 46](#_Toc128395259)

[Team Members: 47](#_Toc128395260)

# Introduction

In this tutoring app I am going to help students to learn emerging technologies in an efficient and easy way. For students that do not know what to study or have trouble searching the study materials, this app is best for them. In this app, we ask students what they want to study or we tell them they have to study this on the basis of some question answers and then provide the best material from all over the world to them. In the first launch, we may only target one or two courses only, and then in the next launch, we target different courses that help the student to make their future better.

# Road Map

Phase 1: MVP Release

* Develop and test basic app functionality, including playing videos from YouTube and downloading video links from Firebase
* Implement course selection feature
* Design and implement basic user interface
* Conduct initial user testing and feedback gathering

Phase 2: Content Expansion

* Create additional video content for courses
* Allow teachers to submit their own video content for review and inclusion in the app through email not directly in the app.
* Enhance course selection functionality to allow users to automatically choose a course by giving answers to some question

Phase 3: Platform Enhancements

* Add the ability to save the video in cache memory in case of internet fluctuation
* Improve video recommendation engine to suggest new content based on user history and preferences
* Implement analytics to track usage and identify areas for improvement

Phase 4: Mobile App Release

* Develop and release mobile versions of the app for iOS and Android
* Optimize user interface for mobile devices
* Integrate with mobile-specific features

Phase 5: Community Building

* Develop social features to allow users to connect with each other and share content
* Implement a ratings and review system for app

The specifics of the roadmap will depend on the priorities and resources of the development team and company. It's important to continually assess and adjust the roadmap based on feedback and changing market conditions.

# Agile Method

1. Planning:

* Create a product backlog with all the features required for the app

1. As a user, I want to be able to view a list of all available courses, so that I can easily find the course I want to take.
2. As a user, I want to be able to select a course and view a list of all videos associated with that course, so that I can watch them in order.
3. As an administrator, I want to be able to add new courses to the app, so that users can have access to new content.
4. As an administrator, I want to be able to remove courses that are no longer relevant, so that users don't waste time looking for outdated content.
5. As a developer, Implement the feature to save the video in the cache so that when the internet fluctuates video does not stop.
6. As a developer, I want to implement a system for tracking user interactions, so that we can analyze how users are engaging with the app and make improvements accordingly.
7. As a developer, I want to optimize the app's performance to ensure that videos load quickly and without interruptions.

* Break down the features into smaller user stories
* Prioritize the user stories in the backlog

1. Sprint 1:

* Conduct a sprint planning meeting and select the 1st user stories to be worked on in this sprint
* Develop a simple user interface to display the list of available courses
* Implement the feature to choose course or chosen automatically by the app by giving answers to some questions
* Implement the feature to retrieve the list of courses from Firebase
* Implement the feature to play videos from YouTube

1. Sprint 2:

* Conduct a sprint planning meeting and select the 2nd user stories to be worked on in this sprint
* Develop a user interface to allow users to choose a course
* Implement the feature to retrieve the list of videos for a selected course from Firebase
* Implement the feature to play selected videos from YouTube

1. Sprint 3:

* Conduct a sprint planning meeting and select the 3rd user stories to be worked on in this sprint
* Implement the feature to save the video in the cache so that when the internet fluctuates video does not stop.
* Implement the feature to stop, rewind, play, fast forward and change speed of the video.
* Add testing to all implemented features

1. Release:

* Release the app to the app store
* Continue to gather feedback and make improvements for future updates

***Personas***

1. A student is a busy graduate student pursuing master's degree in computer science and doing a part-time job and wants to supplement studies with online courses in free time and looking for a simple and convenient tutoring app that can help her learn new skills quickly and easily.

Goals:

* Find and access quality courses on the app
* Easily navigate the app and search for courses
* Save courses for later viewing
* Feel confident that the app is secure and reliable

Challenges:

* Limited time due to her busy schedule
* Limited budget for paid courses
* Slow internet connection at times
* Concerned about the security of her personal information

1. A software engineer working for a tech company. He wants to improve his skills and learn new technologies that can help him in his career growth. He likes to learn through online courses and video tutorials in his free time.

Goals:

* wants to find a simple and easy-to-use app that can provide him with a wide range of video tutorials on different topics related to software engineering.

Challenges:

* has a busy schedule with work and other commitments, so he needs an app that can provide him with a flexible learning experience. He also wants to make sure that the app is secure and that his personal data is protected.

1. Some one who is a High school teacher and has a Bachelor's degree in Education and 10 years of teaching experience

Goals:

* To easily find and share educational videos with his students.
* To have a tool that can help him easily organize course materials.

Challenges:

* Finding quality educational videos that align with his curriculum.
* Keeping track of which videos he has shared with his students.

# Scenarios

1. Students have a few minutes between classes and want to use that time to study. He opens the tutoring app on his phone and browses the available courses. He selects a course on calculus and begins watching a video lecture. The video is stored locally on her phone, so He can continue watching even when he loses the Wi-Fi connection. After the lecture, he takes a quick quiz to test her understanding of the material.
2. downloads the tutoring app from the app store. He opens the app and is presented with a list of different courses to choose from. He selects a course on "Introduction to Machine Learning." The app presents him with a list of videos related to the course, along with a brief description of each video. He selects a video on "Linear Regression" and starts watching it. He can pause, rewind or fast-forward the video as per his convenience. After watching the video, he can mark it as completed and move on to the next one in the course. He can also browse through other courses and select the ones he is interested in. Alex feels secure using the app
3. a high school teacher who is looking for a tool to help him better manage his course materials. He decides to try out a simple tutoring app that he has heard about. He opens the app and is taken to the app's homepage, where he sees a list of courses that he can choose from.

He selects the course that he is teaching, and is taken to a screen that displays all of the videos that have been uploaded or shared for that course. He selects "Photosynthesis" and is presented with a list of results.

He finds a video that looks promising and clicks on it. The video starts playing within the app, and he can easily pause, rewind, or fast forward as needed. He decides to share this video with his students, so he copies the app link and sends it to the students.

Students can download the app and play that video. He feels satisfied that he was able to easily find and share high-quality educational videos with his students.

# Key Features of our application

* Course Catalog: A searchable and filterable course catalog that allows users to browse and choose courses based on their interests or needs.
* Video Player: A high-quality video player that can stream videos from YouTube or play downloaded videos from Firebase.
* Download functionality: Users should be able to watch the videos when the internet fluctuates video must be saved in cache memory.
* Course Recommendations: Personalized course recommendations based on the user's interests, past viewing history, and completed courses can be done by giving answers to some questions.
* Notifications: Optional push notifications to remind users about new course content or upcoming deadlines.
* Share: Users can share the app with other students so that they can also take advantage of our app.

# User stories

1. Student
   * As a busy college student, I want to be able to watch educational videos on my phone so that I can study on the go.
   * As a Student, I want to be able to go on a specific course topic within the app, so that I can easily watch the videos that I need for my studies.
   * As a Student, I want to be able to choose a personalized course or if I do not know what course I have to study app will help me to find that course that I can watch at my convenience so that I can stay organized and on track with my learning goals.
2. Software engineer
   * As a software engineer, I want to find an app that can provide him with a wide range of video tutorials on different topics related to software engineering so that he can improve his skills and stay up-to-date with the latest technologies.
   * As a software engineer, I want to integrate a feature that allows users to choose courses by keyword so that users can easily find relevant content.
   * As a software engineer, I want to improve the performance of the app by implementing caching for frequently accessed video links, so that users can experience faster load times.
3. Teachers

* As a teacher, I want to be able to add new videos to YouTube so the developer adds them to the app for my students.
* As a teacher, I want to be able to share the app with students so that they download the app and watch the videos of the course.
* As a teacher, I want to be able to add a short description or summary for each course or video link so that my students can get an overview of the content before watching the video.

# Development Stack

1. Flutter
2. dart
3. Firebase
4. Android Studio

# Application Infrastructure

1. **A client-side mobile app built using Flutter to provide a responsive and smooth user experience.**
2. **A server-side API built using a modern web framework like Node.js, which connects to the Firebase database to fetch and store video links and video thumbnails.**
3. **A cloud-based database like Firebase to store and manage course data, video thumbnails, and video links.**
4. **A content delivery network (CDN) like Amazon CloudFront or Google Cloud CDN to ensure fast and reliable video streaming to users.**
5. **Third-party services like Google Analytics and Crashlytics to monitor app performance and crashes.**

# App Architecture

For this tutor app that plays videos from YouTube and downloads video links from Firebase, we would use a Clean Architecture.

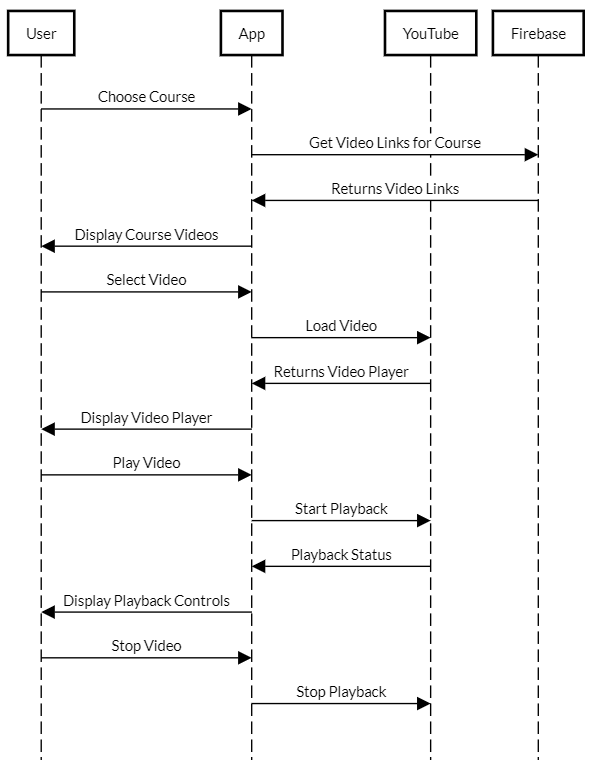
Using the Clean Architecture we would divide the app into several layers such as the presentation layer, domain layer, and data layer. This architecture promotes loose coupling between these layers, which makes it easy to test, maintain, and modify the app in the future.

In the case of this app, the presentation layer could be implemented using Flutter, which would handle the UI and user interactions. The domain layer would handle business logic and use cases, while the data layer would be responsible for accessing data from Firebase.

* Presentation Layer: This layer will contain the UI components that allow users to choose a course and play the video. It will also contain the code to handle the interactions between the UI and the domain layer.
* Domain Layer: This layer will contain the business logic of the app. It will be responsible for downloading the video links from Firebase, playing the video from YouTube, and handling any errors or exceptions that may occur.
* Data Layer: This layer will contain the code to interact with Firebase to download the video links. It will also contain the code to interact with YouTube to play the video. For this purpose, we would use the firebase api and youtube API.

By using this architecture, we can create a robust and scalable app that is easy to maintain and test.

# Sequence Diagram



# Data model

1. Course: represents a course that contains one or more videos, with attributes such as name, description, and image.
2. Video: represents a video that can be played within a course, with attributes such as title, description, URL (e.g., YouTube video ID), and duration.



# Use case

* Student



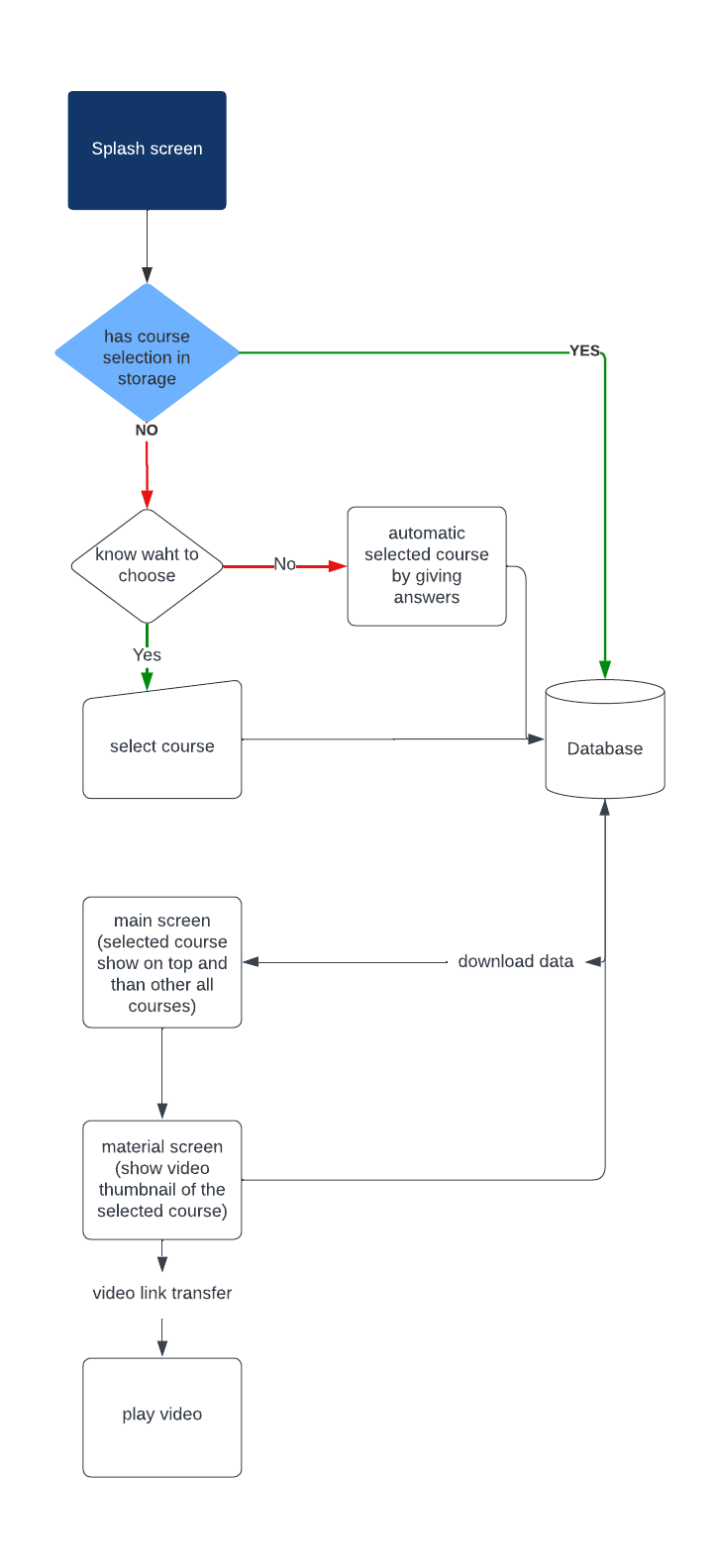
* Software Engineer



* Teacher



# Process diagram



# Application diagram

┌─────────────────------------------------------------------─┐

│ UI/Presentation │

│ Layer │

└─────────────────-------------------------------------------─┘

▲

│

┌─────────────────────------------------------------------─┐

│ Domain/Business │

│ Layer │

└─────────────────────------------------------------------─┘

▲

│

┌───────────────────-------------------------------------───┐

│ Data/Infrastructure │

│ Layer │

└───────────────────-------------------------------------───┘

The diagram shows three main layers: the UI/Presentation Layer, the Domain/Business Layer, and the Data/Infrastructure Layer.

The UI/Presentation Layer handles the user interface and user interaction of the app. It communicates with the Domain/Business Layer to retrieve or send data and business logic. This layer is responsible for rendering the user interface and handling user inputs, such as selecting a course or playing a video.

The Domain/Business Layer is where the app's business logic resides. It contains the use cases that the app needs to accomplish and the data models and repositories required to perform these tasks. This layer communicates with the Data/Infrastructure Layer to retrieve or send data from external sources.

The Data/Infrastructure Layer handles the data access and storage for the app. It includes data repositories, network clients, and any external services or APIs required to perform data operations. This layer communicates with the Domain/Business Layer to retrieve or send data from external sources.

This diagram follows the Clean Architecture principles, where each layer has a clear responsibility and is independent of the other layers. The layers are separated by boundaries that allow for testability, maintainability, and flexibility.

***Security***

* Secure data storage: Video links and user information should be stored securely using encryption or hashing techniques to prevent unauthorized access or data breaches.
* Secure communication: All data transmissions between the app and the server should be encrypted using SSL/TLS protocols to prevent man-in-the-middle attacks and eavesdropping. Moreover you view firebase policy.
* Regular updates and vulnerability scanning: The app should be regularly updated with the latest security patches and tested for vulnerabilities using tools such as penetration testing and vulnerability scanners.
* Error handling and logging: Proper error handling and logging should be implemented to detect and prevent attacks such as SQL injection and cross-site scripting.

# Privacy

Our app, Tutor, is designed to provide you with a simple and effective way to access educational video content. We take your privacy seriously and are committed to protecting your personal information.

Information We Collect

We collect information that you provide to us when you use our app, such as the courses you select and the videos you watch. We also collect device information, such as your device type and operating system, such as your IP address and app usage data.

How We Use Your Information

We use the information we collect to provide you with personalized content and to improve our app. We may also use your information for research and analytics purposes.

Sharing Your Information

We do not sell or share your personal information with third parties. We may share non-personally identifiable information, such as app usage data, with third-party partners for analytics and research purposes. Or you can view the privacy policy of firebase.

Security

We take appropriate technical and organizational measures to protect your personal information from unauthorized access, disclosure, or destruction.

Changes to Our Privacy Policy

We may update our privacy policy from time to time. If we make significant changes, we will notify you by email or by posting a notice on our app.

***Testing Strategy***

1. **Unit Testing**: Unit testing would be performed to test individual units of code for their functionality, accuracy, and stability. Unit tests would be written for the data model, video player, and other critical components of the app.
2. Integration Testing: Integration testing would be performed to ensure that the various components of the app are working together correctly. Integration tests would be written to test how the app interacts with Firebase and YouTube.
3. **Functional Testing:** Functional testing would be performed to ensure that the app is performing the functions that it is supposed to. This testing would include testing for the ability to choose a course, play videos, and download video links from Firebase.
4. **Performance Testing**: Performance testing would be performed to ensure that the app is performing well under different load conditions. Performance tests would be written to simulate different levels of user activity and measure the app's performance metrics such as response time, CPU utilization, and memory usage.
5. **User Acceptance Testing:** User acceptance testing (UAT) would be performed to ensure that the app meets the user's needs and is user-friendly. This testing would involve real users trying out the app and providing feedback on its usability, design, and functionality.
6. **Security Testing**: Security testing would be performed to ensure that the app is secure and the user's data is protected. This testing would include testing for vulnerabilities such as SQL injection, cross-site scripting, and other potential security issues.
7. **Regression Testing:** Regression testing would be performed after each release to ensure that the changes made have not negatively impacted the app's existing functionality. This testing would include running all existing test cases and making sure they still pass.

# Branching Strategy

1. **Main branch:** This branch represents the latest stable version of the app. Any new features or changes are merged into this branch only after thorough testing and review.
2. **Development branch:** This branch is used for ongoing development and testing of new features. Developers work on their individual feature branches and merge them into the development branch once they are tested and ready.
3. **Feature branches:** Developers create a new feature branch whenever they start working on a new feature or bug fix. These branches are created from the development branch and are merged back into the development branch once the feature is complete and tested.
4. **Hotfix branches:** These branches are created to fix critical bugs in the app. They are created from the main branch, and changes are merged back into the main branch once they are tested and reviewed.
5. **Release branches:** When a new version of the app is ready for release, a release branch is created from the development branch. The release branch is then used to test and fix any issues specific to that release. Once the release is ready, it is merged into the main branch.

# Deployment Strategy

1. ***accounts:*** *Create a Google Play Store account and an Apple Developer account to deploy the app on Android and iOS devices respectively.*
2. ***Prepare the app for release****: Make sure the app is properly tested and all bugs are fixed before preparing it for release. Update the app version and build numbers to ensure proper version control.*
3. ***Build the app:*** *Build the app in release mode to optimize its performance and minimize its size. Generate the APK file for Android devices and the IPA file for iOS devices.*
4. ***Submit the app****: Upload the APK and IPA files to the Google Play Store and the Apple App Store respectively. Provide all necessary information such as app name, description, screenshots, and privacy policy.*
5. ***Wait for review****: Wait for the app to undergo review from the respective app stores. This process can take several days to a few weeks depending on the app store's guidelines.*
6. ***Publish the app:*** *Once the app is approved, publish it on the respective app stores. Users can then download and install the app on their devices.*
7. ***Regular updates****: Continuously update the app with new features and bug fixes to keep users engaged and satisfied with the app.*
8. *Monitor app performance: Monitor the app's performance using analytics tools to gain insights into user behavior and improve the app's features accordingly.*
9. Testing: We are using a blue-green deployment strategy. The new version (the blue version) is brought up for testing and evaluation, while the users still use the stable version (the green version). When ready, the users are switched to the blue version. If a problem arises, you can switch back to the green version.

# Requirements

1. To use this app you need a mobile of any android version but greater than api 21 which is android version 5. You also must have internet connectivity.
2. To update or run in an emulator you must have the android studio and flutter and java installed on your computer then open the project in android studio and go to the pub.yml file and run pub get and then pub upgrade.
3. If you do not have flutter installed on the system then watch any YouTube video on how to install flutter on a computer.

# Methodology

1. Dependencies

* **shared preferences:**

we are using shared\_preferences: ^2.0.15 to store information of selected courses or chosen courses in the secure form on the mobile to make user expression heavy and to secure information from hackers.

* cached\_network\_image:

we are using cached\_network\_image: ^3.2.2 to get thumbnail images from the firebase storage and this store images in the cache so that next time it does not download the images again which reduce the network communication.

* firebase\_core:

We are using firebase\_core: ^2.1.1 to use the firebase functionality in our app and core is used to automatically update the firebase dependencies.

* firebase\_database:

we are using the firebase\_database: ^10.0.3 to use the firebase database functionality in our app to download the video thumbnails and video links.

* provider:

we are using the provider: ^6.0.3 for the management of the state such that when we want to update UI when it is running in such we are using this package. And the controller types widgets we are using the provider.

* internet\_connection\_checker: ^1.0.0+1

we are using internet\_connection\_checker: ^1.0.0+1 for checking whether the internet is coming or not if the internet not coming we display the message net not coming.

* google\_fonts:

we are using google\_fonts: ^3.0.1 to make a creative look to our app and to use the fonts from google.

* page\_transition:

we are using page\_transition: "^2.0.9" to make navigation more realistic that increase user experience. This can make a transition from one page to other in an animated way.

* Lottie:

we are using this Lottie: ^2.2.0 for getting some cool animation to our app as we did on the splash screen.

* youtube\_player\_flutter:

we are using youtube\_player\_flutter: ^8.1.2 to play youtube videos in our app. We provide the video id to the youtube player that is downloaded from the database to play video.

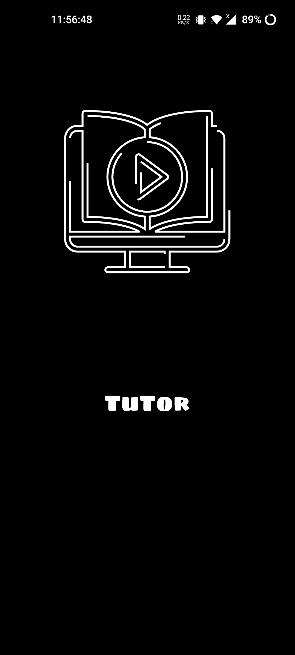
1. Dev Dependencies:

* flutter\_launcher\_icons:

we are using the flutter\_launcher\_icons: "^0.11.0" to change the icon of our app.

1. Working and code:

* Splash screen:

This is the first screen that the user sees when they open our app. To make it cool I am using the Lottie animation as described early. And use black color to protect the user eyesight.

return Scaffold(  
 body: Container(  
 width: width,  
 height: height,  
 decoration: BoxDecoration(  
 color: Colors.black  
 ),  
  
 child: Column(  
 children: [  
  
 Container(  
 height: 500,  
 width: width,  
 child: Lottie.asset('assets/anim.json'),  
 ),  
  
 Text("TuTor",style: GoogleFonts.sigmarOne(fontSize: 30,color: Colors.white,))  
  
  
 ],  
 ),  
 ),  
);

I am only discussing the main part of the code here I use the scaffold to get the UI control of the mobile and use the container to put items on the screen. Her height and width are coming from the app layout class

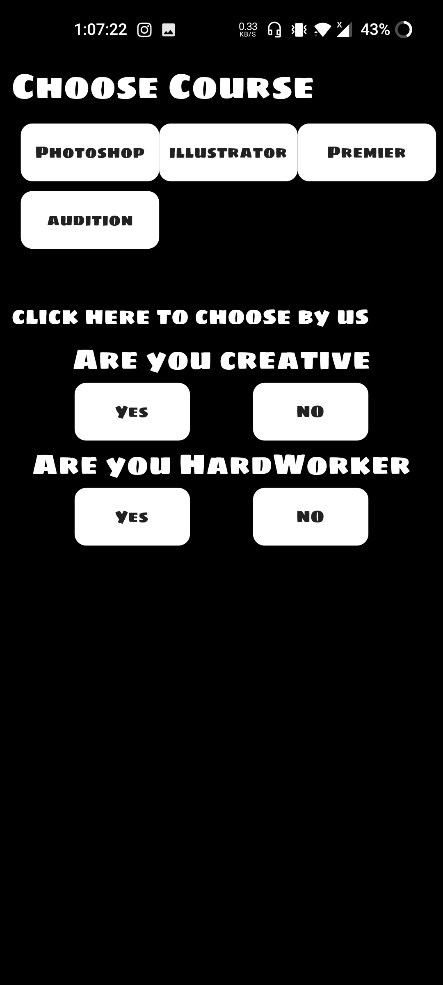
class AppLayout{  
 static *getwidth*(BuildContext context){  
 return MediaQuery.*of*(context).size.width;  
 }  
  
 static *getheight*(BuildContext context){  
 return MediaQuery.*of*(context).size.height;  
 }  
}

there are 2 static methods that give height and width to all other classes where we need Media Query as the built-in class that provides height and width. we navigate to the next class after the 3 seconds for the navigation we bring code from the app state class by the provider

Future<void> change\_screen(BuildContext context)async{  
 Future.delayed(const Duration(seconds: 4),(){  
 try{  
 if(prefs.containsKey("course")) {  
 Navigator.*pushReplacement*(context, PageTransition(  
 child: const HomeScreen(), type: PageTransitionType.fade));  
 } else{  
 Navigator.*pushReplacement*(context, PageTransition(  
 child: const qascreen(), type: PageTransitionType.fade));  
 }  
 } catch(e){}  
 });  
}

Here we are also checking if the course selection is done or not if the course selection is not done then we navigate to the course selection screen or if the course selection is done then we move to the home screen. We are also using page transition for soft navigation we are using fade-in fade-out transition.

* Course selection screen:

 here we are showing all the available courses to the user and if the user is not interested or does not know what to choose then he can give the answer to the given question then the course is chosen by our app. The question and answer only show when the user clicks on the “click here to choose by us” then only question show if he clicks on the no then the next question show if he clicks the yes then the course is selected and in the next question there are also 2 option if he clicks then the course is selected and same on the no selected.

return Scaffold(  
 body: Container(  
 width: width,  
 height: height,  
 color: Colors.*black*,  
  
 child: Padding(  
 padding: const EdgeInsets.only(left: 10 ,right: 10,bottom: 10,top: 50),  
 child: Column(  
 children: [  
  
 Align( alignment: Alignment.*centerLeft*,  
 child: Text("Choose Course",style: GoogleFonts.*sigmarOne*(color: Colors.*white*,fontSize: 30),)),  
  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Column(  
 children: [  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","photoshop");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 120,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("Photoshop",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","illustrator");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 120,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("illustrator",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","premier pro");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 120,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("Premier",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 )  
  
 ],  
 ),  
 SizedBox(height: height\*0.01,),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","audition");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 120,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("audition",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 )  
 ],  
 )  
 ],  
 ),  
 ),  
  
 SizedBox(height: height\*0.04,),  
  
 InkWell(  
 onTap: (){  
 if(provider.check == true){  
 provider.check = false;  
 provider.notifyListeners();  
 }else{  
 provider.check = true;  
 provider.notifyListeners();  
 }  
 },  
 child: Align( alignment: Alignment.*centerLeft*,  
 child: Text("click here to choose by us",style: GoogleFonts.*sigmarOne*(color: Colors.*white*,fontSize: 20),)),  
 ),  
  
 Visibility(  
 visible: provider.check,  
 child: Container(  
 child: Column(  
 children: [  
 Text("Are you creative",style: GoogleFonts.*sigmarOne*(color: Colors.*white*,fontSize: 25),),  
  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","photoshop");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 100,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("Yes",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
  
 InkWell(  
 onTap: (){  
 provider.newcheck = true;  
 provider.notifyListeners();  
 },  
 child: Container(  
 width: 100,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("NO",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
 ],  
 ),  
  
 Visibility(  
 visible: provider.newcheck,  
 child: Container(  
 child: Column(  
 children: [  
 Text("Are you HardWorker",style: GoogleFonts.*sigmarOne*(color: Colors.*white*,fontSize: 25),),  
  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","illustrator");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 100,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("Yes",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
  
 InkWell(  
 onTap: (){  
 provider.prefs.setString("course","audition");  
 Navigator.*pushReplacement*(context, PageTransition(child: const HomeScreen(), type: PageTransitionType.fade));  
 },  
 child: Container(  
 width: 100,  
 height: 50,  
 decoration: const BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.all(Radius.circular(10))  
 ),  
 child: Center(child: Text("NO",style: GoogleFonts.*sigmarOne*(),)),  
 ),  
 ),  
  
 ],  
 ),  
 ],  
 ),  
  
 ))  
  
 ],  
 )),  
 )  
  
 ],  
 ),  
 ),  
  
 ),  
  
  
);

here we are using the visibility widget to show and un shows the questions.

* Home Screen :

 this is our home screen here the selected course data are coming from the shared preferences

return Scaffold(  
 body: Container(  
 width: width,  
 height: height,  
 decoration: BoxDecoration(  
 color: Colors.*black* ),  
  
 child: Column(  
 children: [  
  
 Container(  
 decoration: const BoxDecoration(  
 borderRadius: BorderRadius.only(bottomLeft: Radius.circular(50),bottomRight: Radius.circular(50)),  
 color: Colors.*blue* ),  
 child: Padding(  
 padding: const EdgeInsets.only(top: 20,bottom: 40,left: 20,right: 10),  
 child: Column(  
 children: [  
  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text("Suggested course",style: GoogleFonts.*sigmarOne*(fontSize: 30,color: Colors.*white*),)),  
  
 InkWell(  
 onTap: (){  
 Navigator.*push*(context, PageTransition(  
 child: materialscreen(name:provider.prefs.getString("course").toString()), type: PageTransitionType.fade));  
 },  
 child: Align(  
 alignment: Alignment.*centerLeft*,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Container(  
 decoration: const BoxDecoration(  
 color: Colors.*white* ),  
 child:  
 Padding(padding: const EdgeInsets.only(left: 8,right: 8),  
 child: Text(provider.prefs.getString("course").toString(),style: GoogleFonts.*sigmarOne*(fontSize: 30,color: Colors.*black*),)),  
 ),  
 ),  
 ),  
 ),  
  
  
  
 ],  
 ),  
 ),  
 ),  
  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text("All Course",style: GoogleFonts.*sigmarOne*(fontSize: 30,color: Colors.*white*),)),  
  
  
 Expanded(  
 flex: 1,  
 child: FirebaseAnimatedList(  
 query: provider.dbref.child('playlist'),  
 defaultChild: const Center( child: CircularProgressIndicator()),  
 itemBuilder: (BuildContext context, DataSnapshot snapshot, Animation<double> animation, int index){  
  
 return InkWell(  
 onTap: (){  
 Navigator.*push*(context, PageTransition(  
 child: materialscreen(name:snapshot.key.toString()), type: PageTransitionType.fade));  
 },  
 child: Card(  
 color: Colors.*black*,  
  
 child: Container(  
 width: width,  
 child: Column(  
 children: [  
 CachedNetworkImage(  
 imageUrl: snapshot.value.toString(),  
 progressIndicatorBuilder: (context, url, downloadProgress) =>  
 CircularProgressIndicator(value: downloadProgress.progress),  
 errorWidget: (context, url, error) => Icon(Icons.*error*),  
 ),  
  
 Text(snapshot.key.toString(),style: GoogleFonts.*sigmarOne*(fontSize: 30,color: Colors.*white*),),  
  
 SizedBox(height: 1,child: Container(color: Colors.*white60*,),)  
 ],  
 )  
 ),  
 ),  
  
 );  
  
 },)  
 )  
  
  
 ],  
 ),  
 ),

here we are using the scaffold as described early and container we are using the Ink well widget to make a widget work like a button. We are using align to align the widget to the left right top bottom to make UI interesting. I am using the firebaseanimatedlist to download the list of data from the firebase database we are putting this widget into the expanded widget to make the full rest of the space in the firebase list to occupied. We are using cached network images as described early

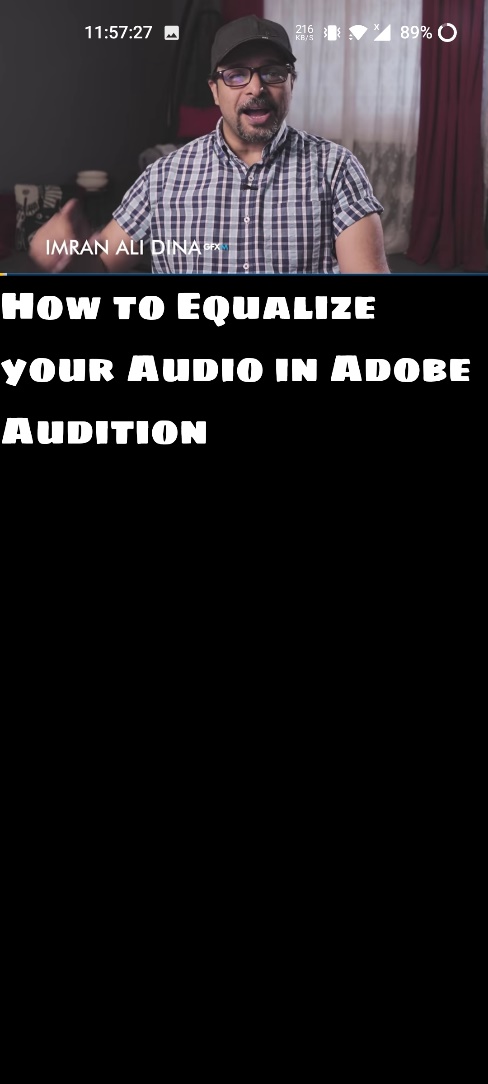
* Material Screen:

 here all the material of the selected course is shown to the user to select what video he wants to see and go to the next video screen to play the video.

return Scaffold(  
 body: Container(  
 width: width,  
 height: height,  
 color: Colors.*black*,  
  
 child: Padding(  
 padding: const EdgeInsets.only(top: 20),  
 child: Column(  
 children: [  
 Expanded(  
 flex: 1,  
 child: FirebaseAnimatedList(  
 query: provider.dbref.child('tutorial').child(name),  
 defaultChild: const Center( child: CircularProgressIndicator()),  
 itemBuilder: (BuildContext context, DataSnapshot snapshot, Animation<double> animation, int index){  
  
 Map data = snapshot.value as Map;  
  
 return InkWell(  
 onTap: (){  
 Navigator.*push*(context, PageTransition(  
 child: videoscreen(id:data["id"],des:data["text"]), type: PageTransitionType.fade));  
 },  
 child: Card(  
 color: Colors.*white10*,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Container(  
 width: width,  
 child: Column(  
 children: [  
 CachedNetworkImage(  
 imageUrl: data["link"],  
 progressIndicatorBuilder: (context, url, downloadProgress) =>  
 CircularProgressIndicator(value: downloadProgress.progress),  
 errorWidget: (context, url, error) => Icon(Icons.*error*),  
 ),  
  
 Text(data["text"],style: GoogleFonts.*sigmarOne*(fontSize: 20,color: Colors.*white*),),  
 ],  
 )  
 ),  
 ),  
 ),  
 );  
  
 },)  
 ),  
 ],  
 ),  
 ),  
  
 ),  
 );  
 }  
}

here we are using the same scaffold, container, firebase animated list, and expanded widgets as described early we are using a text widget to write text on the screen. We are using a sized box to make space between the widgets. And we are using the card widget to elevate the widget inside it from the base screen.

* Video screen

 here the user can play the desired video of the desired course. In the player, the user can play stop the video and move forward and backward and speed up the video and view the remaining time of the video.

Widget build(BuildContext context) {  
 double width = AppLayout.*getwidth*(context);  
 double height = AppLayout.*getheight*(context);  
 AppState provider = Provider.*of*<AppState>(context);  
  
 provider.controller = YoutubePlayerController(  
 initialVideoId: id,  
 flags: const YoutubePlayerFlags(  
 autoPlay: true,  
 hideControls: false,  
 mute: false,  
 loop: true,showLiveFullscreenButton: false  
 ),  
 );  
  
 return YoutubePlayerBuilder(  
 onExitFullScreen: () {  
 SystemChrome.*setPreferredOrientations*(DeviceOrientation.*values*);  
 },  
 player: YoutubePlayer(  
 controller: provider.controller,  
 showVideoProgressIndicator: true,  
 progressColors: const ProgressBarColors(  
 playedColor: Colors.*amber*,  
 handleColor: Colors.*amberAccent*,  
 ),  
 bottomActions: [  
 CurrentPosition(),  
 ProgressBar(isExpanded: true),  
 RemainingDuration(),  
 PlaybackSpeedButton(),  
 ],  
  
 ), builder: (BuildContext , Widget ) {  
 return Scaffold(  
 body: Container(  
 height: height,  
 width: width,  
 color: Colors.*black*,  
  
 child: Column(  
 children: [  
 Widget,  
 Text(des,style: GoogleFonts.*sigmarOne*(fontSize: 30,color: Colors.*white*),),  
  
 ],  
 ),  
  
  
 ),  
 );  
 },  
  
 );  
}

Here the YouTube video player is used to play the videos we give the id of the video to the player id that we got from the last material screen we pass the id to the class object in the navigation part and get the id in this class in the constructor and by using this keyword we assign the id to the variable and same as for the text.

# Feedback:

Overall, the simple tutoring app that only plays videos from YouTube and video links downloaded from Firebase with the ability to choose a course is easy to use and navigate. The ability to choose courses and access video links from Firebase adds a level of customization to the app, allowing users to select content that suits their needs.

One potential improvement would be to add more interactive features, such as quizzes or assignments, to enhance the learning experience. Additionally, the app could benefit from adding more detailed course descriptions and instructor profiles to help users make more informed decisions about which courses to take.

Overall, the app provides a convenient and accessible way to access educational content, but there is room for further improvement in terms of user engagement and course information.

# Reference:

1. "Top 10 E-Learning and Educational Apps in 2021" by Education Technology Insights
2. "Benefits of Using Video-Based Learning" by eLearning Industry
3. "Introduction to Firebase for Mobile Development" by Firebase Google Developers
4. "The Agile Methodology: A Beginner's Guide to Agile Project Management" by Wrike
5. "Mobile App Development: The Ultimate Guide for Building a Successful App" by BuildFire
6. "Flutter Documentation" by Flutter.dev
7. "Best Practices for Mobile App Security" by OWASP (Open Web Application Security Project)
8. "Privacy Policy Generator" by Shopify
9. "How to Improve User Experience in Mobile Apps" by UX Collective
10. "Mobile App Deployment Strategies" by Bitbar

# Team Members:

This is a solo Project