UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

Architecture Development Document - Tutor Me

Authors: CapsOn

1. Architectural design strategy

We will be making use of the Decomposition design strategy
which has to do with breaking down a complex system into smaller and much more manageable chunks.

This method requires a ‘big-picture’ understanding of the system as a whole in the beginning stages. The type of decomposition typically performed in software development and engineering is Structural Decomposition and involves the decomposition of a software system into physical and logical components. The arrangement of these components is discovered during the decomposition process.

We will be picking this design strategy as the project we are working on is well-defined and it is tightly-coupled, which will allow us to break it down into granular subsystems, develop them and then integrate the various subsystems back into the whole system. We broke down the development of the TutorMe system into Tutor Subsystem and a Tutee subsystem, then further broke down each of those into video call, chat functionality, profile management and connecting Tutors with Tutees . This design strategy can also be adapted to work with our Agile workflow as we can continue refining the decomposition process as we understand more about the application.

2. Architectural styles

We will be making use of a microservices architecture. This is because microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility. It requires less scaling and maintenance cost. It is language independent, allows for separation of concerns and is relatively easy to deploy compared to most architectures.

All these advantages of this architecture provides an added benefit of not relying on every component to be fully functional in order to run the app (failure of one component should not drastically impact the project). For example, failure of our videoSDK would not impact other parts of the app, like chat function. Another benefit is that the entire system does not need to be scaled, but just the relevant subsystems when required

3. Architectural quality requirements

- Availability

The app needs to be available at least 99% of the time. For example we can count on the API and database to almost always be available to take requests from Tutors/Tutees as long as they have internet access.

- Security

The app Should keep Tutors and Tutees private documents confidential and have only the people that the user has approved or is intending to send the documents to. We also do client side and server side validation of the users details upon registration and they will be given a unique id upon successful registration. We use encapsulation principles to prevent SQL injection attacks. We would also hash and salt passwords before we store them in the database for all users.

- Maintainability

This is the ability to repair, correct, and and maximize a products lifespan while coping with changing environments. We can do this by making sure that subcomponents of the system work independently of each other so that failure or deprecation of one component does not impact other components. An example is how we plan to implement the chat functionality of the app independently to the video call functionality and Tutor and Tutee services.

- Scalability

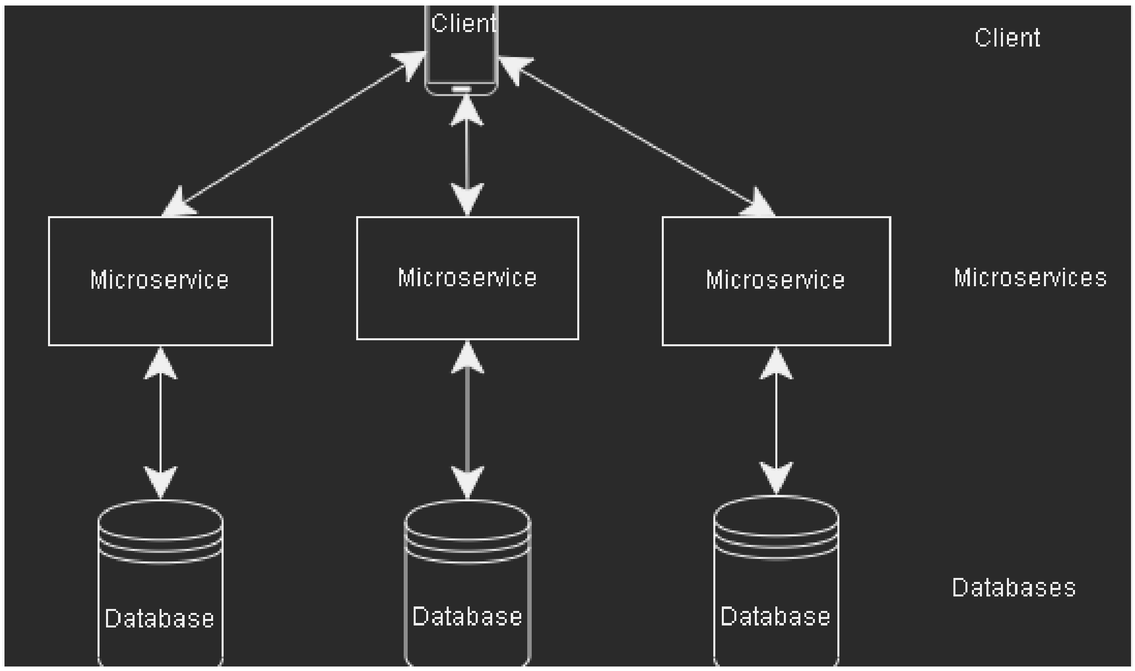
The application will be used mainly by students, thus it is expected that the system should dynamically scale to meet the demands of all the students using the system.

South Africa has 26 public universities with nearly one million students while 700 000 students are registered at the more than 50 higher education training colleges (TVET colleges – Technical vocational education training). An additional 90 000 students can be found at various private institutions. According to https://www.cmi.no.

The system should be able to scale to meet the demands of such growth, without requiring too much overhead and keeping costs down as much as possible.

- Serviceability
  This refers to the ability of technical support personnel to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis. We should be able to update the source code on github when we find errors or defects.

4. Architectural design & pattern



The microservices architecture was set up such that the flutter app (client) can interact with .net core backend (Microservice) in order to store, modify and retrieve tutor and tutee information from the Microsoft SQL Server Database (Database). We also made it possible for the app to interact independently with signalR (Microservice) in order to simulate a realtime chat functionality of the app. SignalR helps in this regard because it gives a layer of abstraction when working with web sockets making it easier to add "real-time" web functionality to our ASP.NET application. Similarly, we made it possible for the app to interact independently with VideoSDK (Microservice) in order to simulate a video call functionality of the app.

5. Architectural constraints

- The app should be able to work on all android devices running Android 4.1 and newer. Iphone users will need to update to iOS 12. The technologies that we use should be compatible with these devices (which they are).
- The app should be as fault-tolerant as possible. The nature of the microservice architecture helps in this regard by making each service as independent to the others as possible.
- The system should make use of open source solutions for accomplishing most (if not all) tasks.

6. Technology choices

6.1 Client

6.1.1 Overview

We have chosen to make use of Flutter for our mobile application framework. Flutter is a cross-platform, open-source toolkit, framework and software development kit built by Google that uses the Dart programming language (also developed by Google) as its language of choice for its environment. It can be used to build applications for Android, iOS, Windows, macOS and the Web using a single codebase. Since its release in 2015, Flutter has received widespread acclaim and support from developers. Flutter also has the ability to compile to native machine code, instead of running the whole application in a web view like other competing alternatives.

6.1.2 Pros

- Supports creating applications in all the major platforms
- Can render and compile applications into native platform machine code, resulting in increased performance
- Has access to many native API's
- Can develop an application for multiple platforms using a single codebase
- Provides optimized pre-built components suitable for development in platforms such as Android and iOS
- Reduced code development time
- Well documented and supported by community

6.1.3 Cons

- Design guidelines are segmented on a per platform basis, which means that creating applications that ‘feel’ like applications for a specific platform could require creating separate applications for each platform.
- The SDK is still relatively new and has not matured yet, meaning that there still exist some components of it that are unstable or still being developed.
- App sizes are relatively large due to overhead introduced when compiling the application in an environment which is not native to the platform.
- The app might not be as optimized for the target device as compared to if it was made using a native platform.

6.1.4 Justification

Flutter will fit in well with our architectural pattern of choice as it is ready-built for using the Microservices architecture when developing applications. It allows code to be segmented into packages that can be imported and used by other packages, meaning that code can easily be separated and grouped accordingly. Flutter will also function entirely as the Client in the Microservices Model chosen as our architectural style as it is suitable for the Mobile Application Framework in the Client Subsystem. The package structure of the application also means that the code can be grouped by the subsystems that were identified during the decomposition phases. Flutter is also created and managed by Google, which essentially guarantees its longevity in the industry.

6.2 Microservices

6.2.1 ASP.NET Core

6.2.1.1 Overview

We have chosen to make use of ASP.NET core as our main server side framework. ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-enabled, Internet-connected apps. With ASP.NET Core, you can: Build web apps and services, Internet of Things (IoT) apps, and mobile backends on windows, MacOS or linux.

6.2.1.2 Pros

- Has a lightweight, high performance, and modular HTTP request pipeline
- Architected for testability.
- Open source and community focused
- Integration of modern, client-side frameworks and development workflows.
- Built-in dependency injection
- Side-by-side versioning.

- Scaling and Dockerization

6.2.1.3 Cons
- Legacy
- Learning Curve
- Windows-based Development Tools

6.2.1.4 Justification
ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-enabled, Internet-connected apps. With ASP.NET Core, you can: Build web apps and services, Internet of Things (IoT) apps, and mobile backends on windows, MacOS or linux. These attributes of ASP.NET Core will definitely aid us with the creation of our Tutor me app on android and

6.2.2 SignalR

6.2.2.1 Overview
SignalR is a free and open-source software library for Microsoft ASP.NET that can be used to add any sort of "real-time" web functionality to your ASP.NET application. While chat is often used as an example, you can do a whole lot more. Any time a user refreshes a web page to see new data, or the page implements long polling to retrieve new data, it is a candidate for using SignalR. It gives a layer of abstraction when working with web sockets and falls back to other compatible techniques for older browsers making it easier to add "real-time" web functionality to our ASP.NET application.Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.

6.2.2.2 Pros
- Not restricted to web clients
- Selection of the best available communication mechanism automatically
- Binding complex objects (JSON)
- Open-source solution available on GitHub that can be installed via NuGet.
- Bidirectional communication for (web) clients.
- Easy to setup.
- Install by issuing NuGet command: “install-package Microsoft.AspNet.SignalR”.
- SignalR can run inside a web application or self-hosted.
- Easy to debug. It can run in IIS Express. All the powers of Visual Studio just wait to be utilized.
- Very good documentation and tutorials

6.2.2.3 Cons

- Not a reliable communication
- ASP.Net Development companies should not use it if a guaranteed delivery of messages is mandate. Bank Transaction are a negative example.
- It uses dynamics. Some syntactical errors can only be detected at runtime. However, only a few lines of code are affected. So, it’s not a real problem detected all errors related to the usage of dynamics.
- At times, it’s difficult to attach a debugger to the running process, especially under heavy load.
- Allows only 10 connections when running in IIS in desktop systems.
- Web clients and the server use different technologies so there is not much code reuse between them.

6.2.2.4 Justification

SignalR is looking to be a good library to use when implementing our chat and group chat functionality for our Tutor Me app. We can also use it to check whether

6.2.3 VideoSDK

6.2.3.1 Overview

VideoSDK.live allows for easier addition of Video chat - Voice chat, Live streaming, or host video in your app. It is free and open source and is compatible with flutter.

6.2.3.2 Pros

- Supports multiple users at once
- Can work with unstable internet connection
- Supports recording HD quality
- Supports streaming over multiple platforms at once
- Supports screen sharing

6.2.2.3 Cons

- A limited number of participants can join
- Poor video quality in case of unstable internet connection

6.2.3.4 Justification

VideoSDK.live will make it easier for us to add Video chat - Voice chat functionallity in our app. It is free and open source and is compatible with flutter.

6.3 Database

6.3.1 Overview

We have chosen to make use Microsoft SQL Server as our database. It is a relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments.

6.3.2 Pros

- Easy Installation
- Faster Query Processing
- Security
- Multiple Editions and Price Variations
- Excellent Data Restoration and Recovery Mechanism
- Standardized Language
- Portable
- Excellent Data Recovery Support.

6.3.3 Cons

- Partial Control

6.3.4 Justification

We will use Microsoft SQL Server for the storage, modification and retrieval of the data that we will use in our Tutor_Me app.