Stack Overflow

Backend

Intro

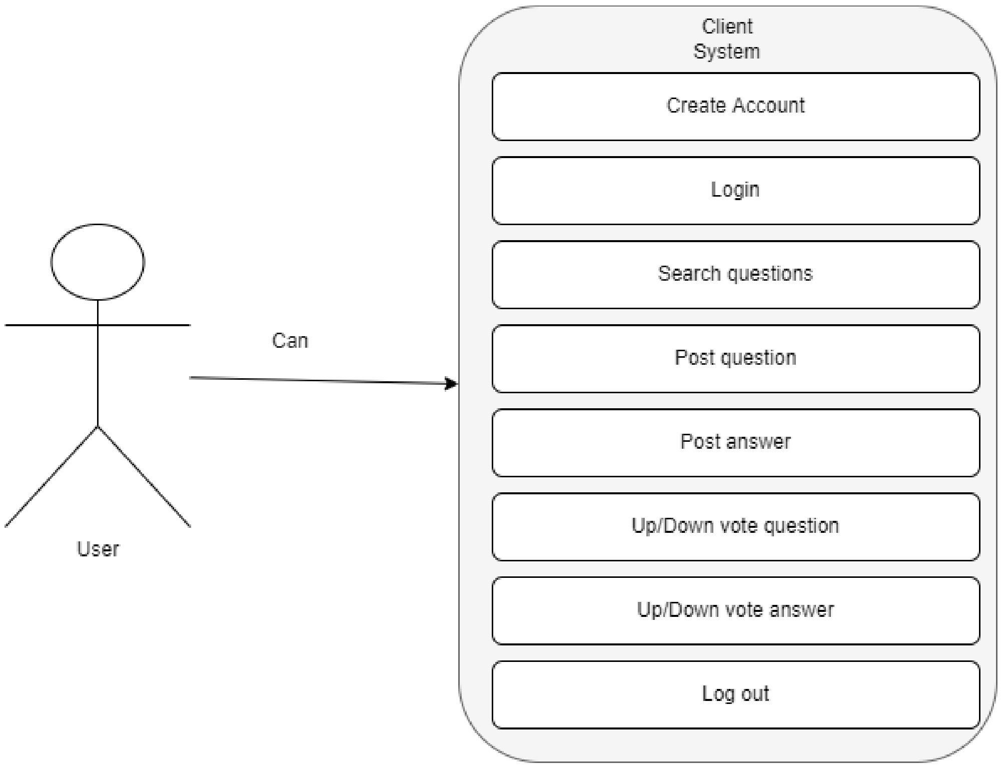
As we kept moving forward in the 21st century, more and more of our solutions to daily problems have been relying on modern software. The building blocks used in creating software has in turn also suffered major changes as to accommodate the many different type of applications. With this new approach to problem solving, we managed to create a new problem, and that is what happens when we don't know how to use or misuse these building blocks. To combat these 21st century problems, we will come up with a 21st century solution, the solution is to create a community where users can place questions and other users can answer them thus helping in building better and more reliable software.

Tech Stack

The technology that makes this app a reality is node.js + express.js for the backend and MySQL and TypeORM for the data base. Since this is a web app, this allows for easy cross platform development as most modern devices now have the needed requirement, a web browser, to run this application. For our security part of the application, we will use JSON web tokens to keep track of user sessions as well as two factor authentication using speakeasy for a more secure login experience.

Use Case Example

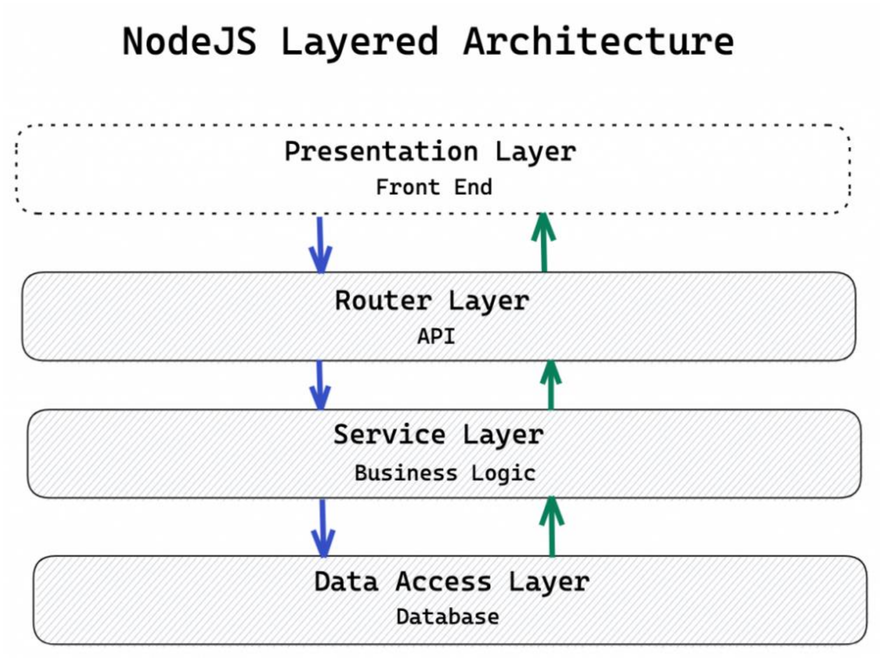
In the following diagram we will represent some use cases, followed by a description of the login use case:



I will describe the way that the login use case works. The client creates a post request containing his email and password in the body. The route handler will then pass the data along to the service layer. The service layer will then try to retrieve the user using the data access layer, if user is successfully retrieved, then the given password is compared to the password in the data base, if the passwords match, then the service layer will create a JWT token and store it using the data access layer. The created token is return to the route handler, and further on to the user who created the request. If any of the steps fail, the message “Login failed”, will be return to the user.

Architecture

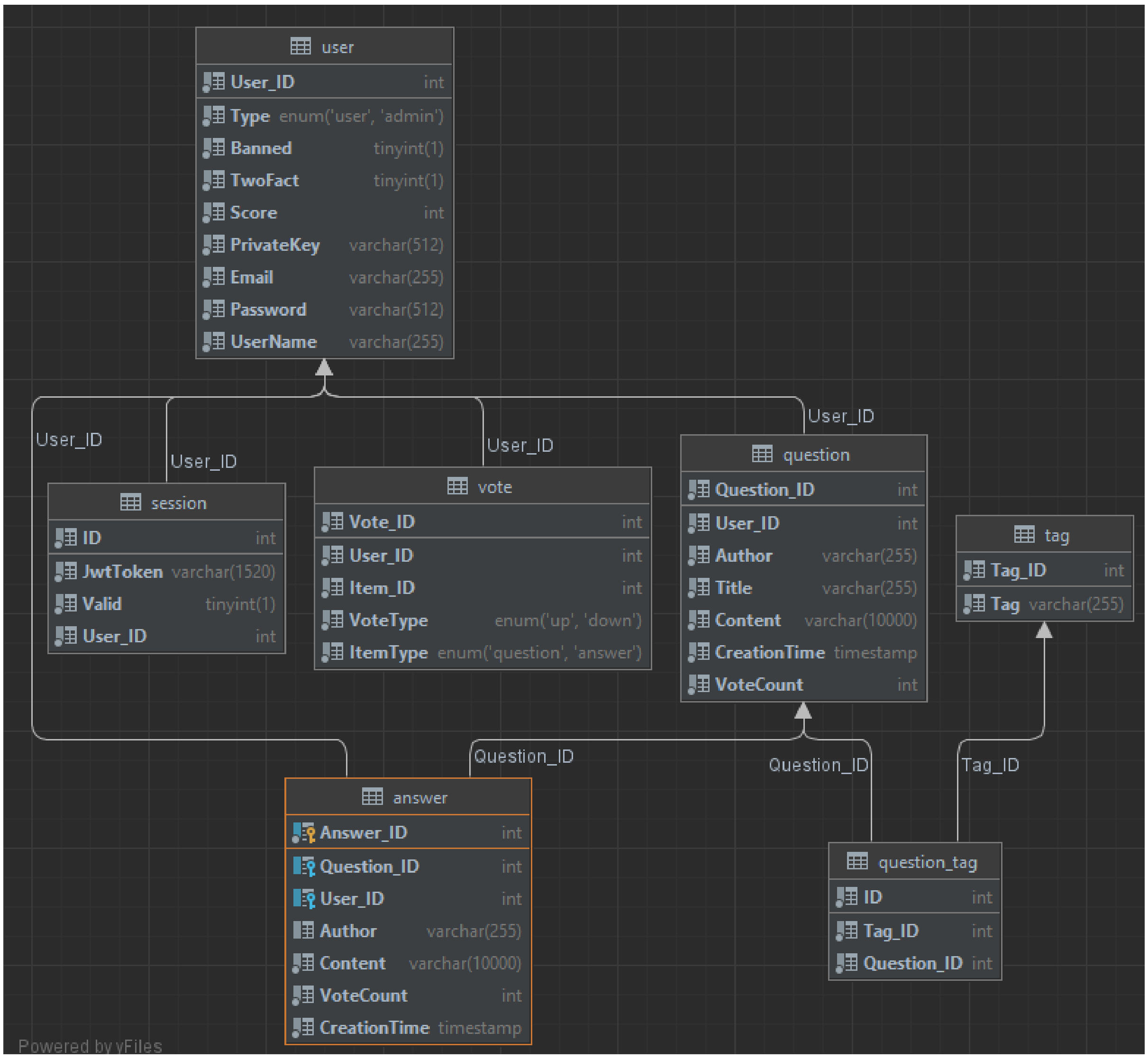
Behind all great apps, there are great architectures, and this app is no different. This app will feature the layered architecture.



As can be seen in the above diagram, this architecture is made up of 4 layers, 3 for the backend and 1 for the frontend. The router layer holds the endpoints and routes the data to the service layer which manipulates the data. The data access layer is then used to save the data or retrieve new data. The reworked items are then sent back up all the way to the presentation layer.

Database diagram

The following diagram represents our database structure



user	
User_ID	int
Type	enum('user', 'admin')
Banned	tinyint(1)
TwoFact	tinyint(1)
Score	int
PrivateKey	varchar(512)
Email	varchar(255)
Password	varchar(512)
UserName	varchar(255)

session	
ID	int
JwtToken	varchar(1520)
Valid	tinyint(1)
User_ID	int

vote	
Vote_ID	int
User_ID	int
Item_ID	int
VoteType	enum('up', 'down')
ItemType	enum('question', 'answer')

question	
Question_ID	int
User_ID	int
Author	varchar(255)
Title	varchar(255)
Content	varchar(10000)
CreationTime	timestamp
VoteCount	int

tag	
Tag_ID	int
Tag	varchar(255)

answer	
Answer_ID	int
Question_ID	int
User_ID	int
Author	varchar(255)
Content	varchar(10000)
VoteCount	int
CreationTime	timestamp

question_tag	
ID	int
Tag_ID	int
Question_ID	int

Bibliography

https://app.diagrams.net/

https://ctrly.blog/nodejs-layered-architecture/

https://dev.to/santypk4/bulletproof-node-js-project-architecture-4epf#architecture

http://expressjs.com/en/guide/using-middleware.html#middleware.application

https://www.section.io/engineering-education/speakeasy-two-factor-authentication-in-nodejs/

https://typeorm.io/#/

Frontend

Intro

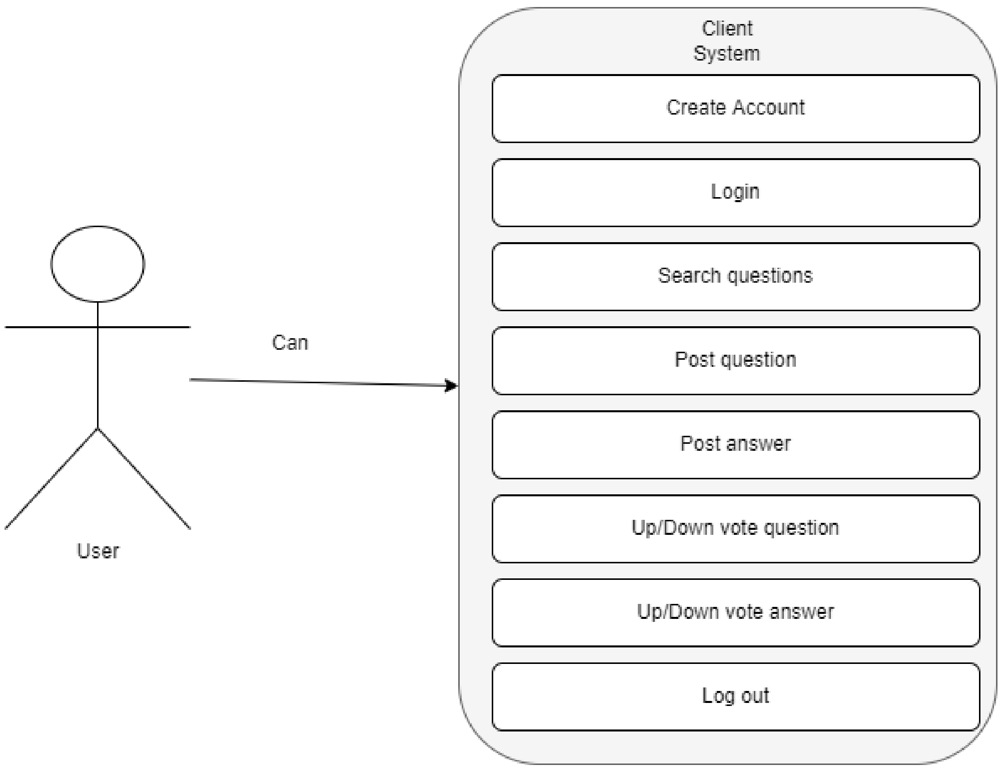
Now that we have a good and stable backend, it is time for the frontend to be our major focus. Since this is the part that people will interact with, it is essential to achieve a functional and clean result.

Tech Stack

The technology that makes this app a reality is angular. Since this is a web app, this allows for easy cross platform development as most modern devices now have the needed requirement, a web browser, to run this application.

Use Case Example

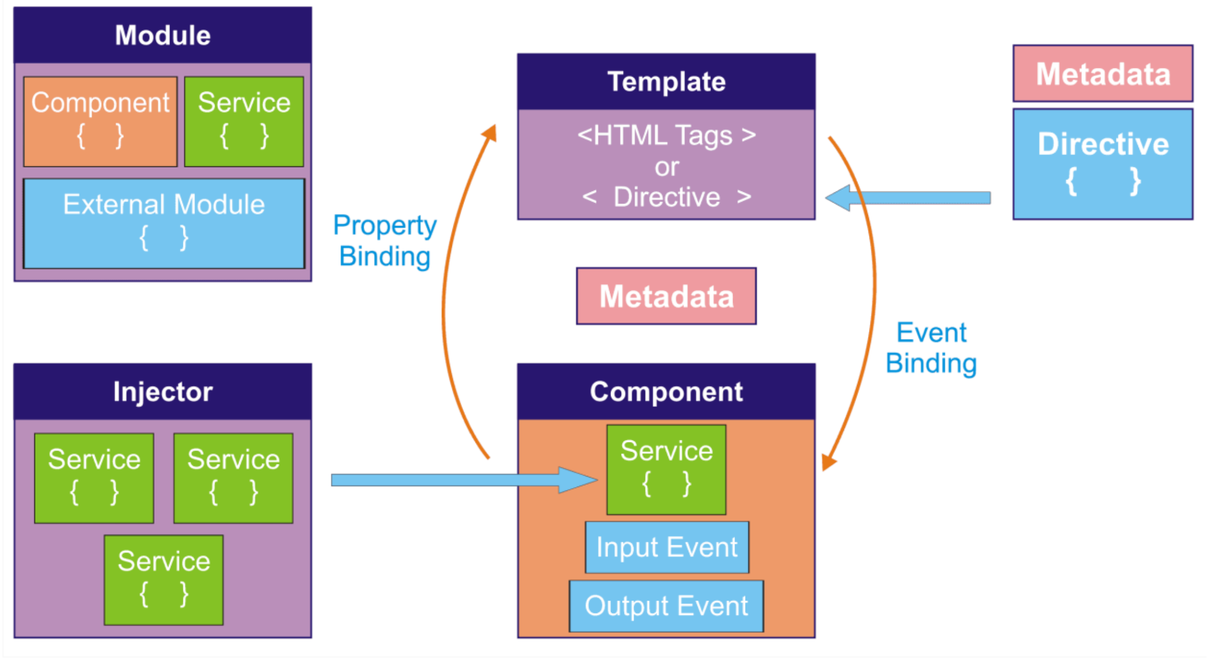
In the following diagram we will represent some use cases, followed by a description of the login use case:



I will describe the way that the login use case works. Once the client accesses the login page of our app, he will be presented with a form that requires an email and password as input. Once the form is completed the login button shall be pressed and the front end will make an API call to the backend to try and create a session for the user. If the user completed the form with incorrect information a message will be displayed detailing, why the login failed. If the login is successful, the user will receive a session token from the API call and he will be redirected to the home page.

Architecture

We focused on using a component-based architecture. This gives some major advantages especially in the reusability point of view. For example, our home page is put together using many components, such as a question, this component is reused quite a bit since there can be multiple questions on a home page.



Bibliography
https://www.ngdevelop.tech/wp-content/uploads/2017/12/Angular_Architecture.png

https://angular.io/api/common/NgIf

https://material.angular.io/components/chips/api#MatChipList

https://angular.io/api/common/AsyncPipe

https://stackoverflow.com/questions/68017199/cannot-get-http-headers-on-angular-interceptor

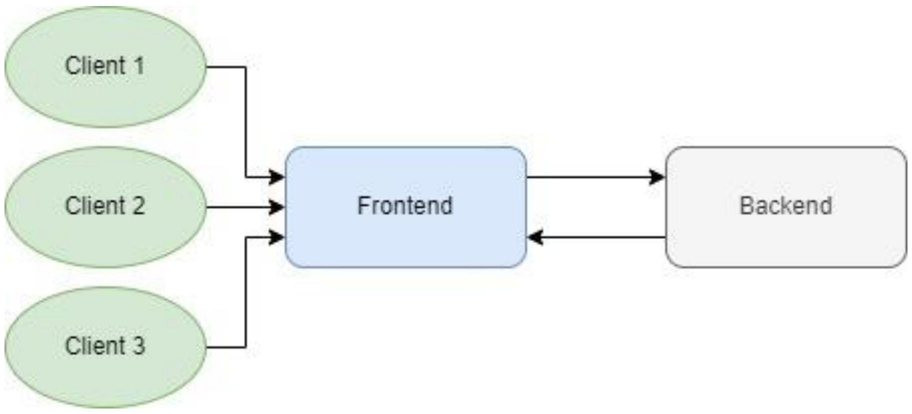
Link

Intro

Now that we have a good backend and frontend, it is time to link the two so that we may achieve a functional web app.

Creating the link

For creating the link, we will use angular to send requests to the API backend. This is done using the built in HttpClient class.


Client 1
Client 2
Client 3
Frontend
Backend