BIM423 - SOFTWARE ENGINEERING

2022-2023 FALL SEMESTER

Virtual

Patient

Group 12

Week 11 / 13.12.2022 – 19.12.2022

Members

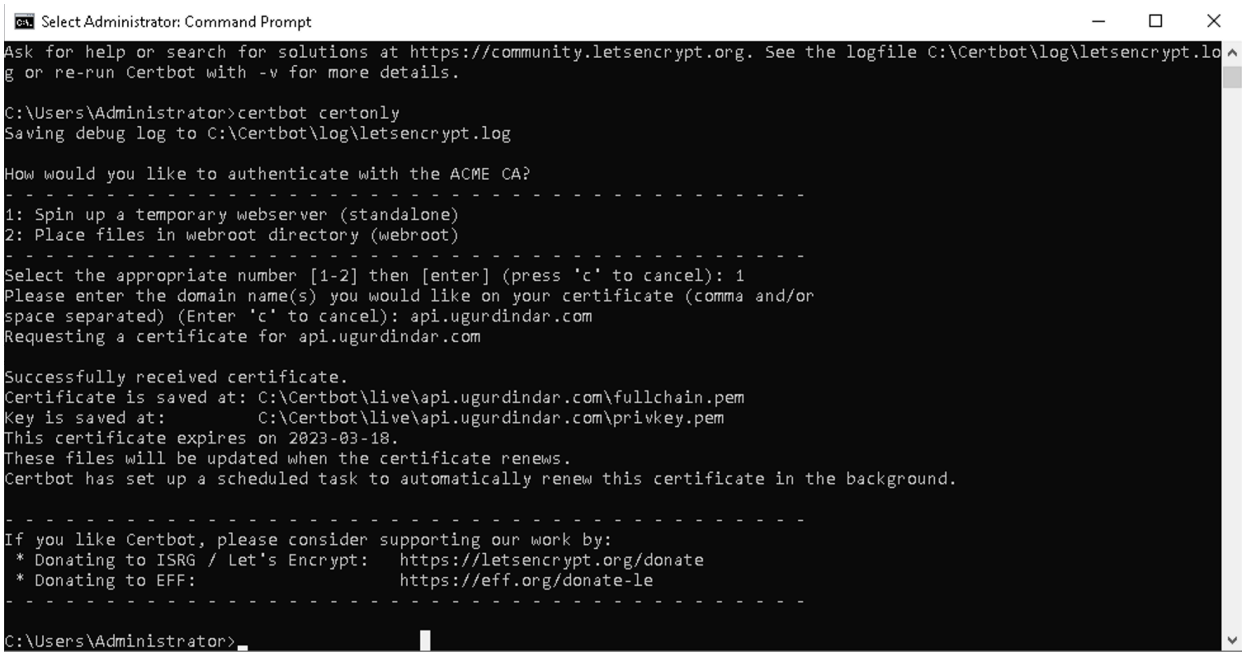
1. ***Uğur Dindar***
2. ***Emre Kayacık***
3. ***İdris Can Bölükbaşı***
4. ***Oğuzhan Çetin***
5. ***Umutcan Yavuz***
6. ***Yunus Emre Korkmaz***

Links

- ***GitHub Link: https://github.com/VPatient***
- ***Website Link: https://vpatient.github.io/***
- ***Repo Link: https://github.com/orgs/VPatient/repositories***
- ***Dashboard Link: https://vpatientdashboard.web.app/***

Works Done

- API URL has been changed from ‘server.ugurdindar.com:5000' -> ‘api.ugurdindar.com' (including CNAME A configurations, HTTP/HTTPS support)
- Our API used to work only with HTTP and with port 5000. HTTPS can be used only if we found a way to produce a SSL certificate and renew it periodically and most importantly systematically. And we have found a solution to that, introducing CertBot which is a certification bot as you may guess and used to generate certificates not just only self-signed and known all across the world. There are prerequisites to use CertBot, first and most important thing is a domain name to use it and a server to deploy it. In order to use CertBot, you have to install it to your server. After that you can use ‘certbot certonly’ command to create your certificate.



After creating your certificate your keys and certificates should appear in given location on the CLI. Then you can implement them to your application/website.

In Windows, given certificate locations consists of shortcuts to currently available certificates. Since we don't want to renew the locations and names of the certificates in every 3 months we added a variable to our dotEnv environment in our API. As you can see below we have created a structure to listen both http & https servers at the same time. Thus, we can use HTTPS and HTTP at the same time.

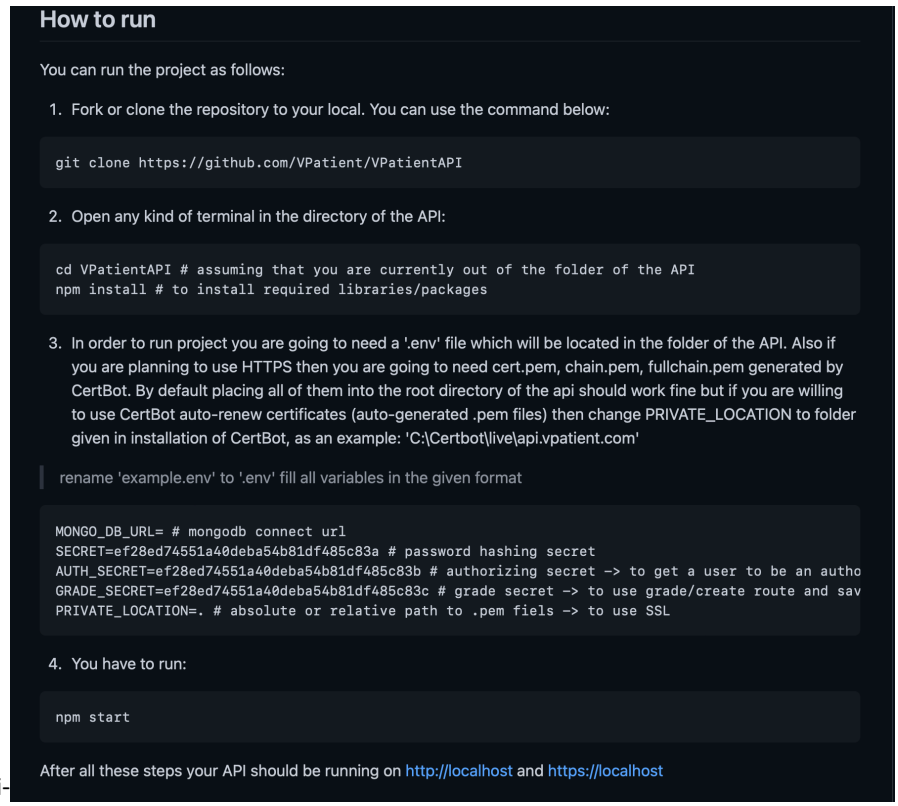
```
// get certificates
const privLocation = process.env.PRIVATE_LOCATION;
const privateKey = fs.readFileSync(`${privLocation}/privkey.pem`, 'utf8');
const certificate = fs.readFileSync(`${privLocation}/cert.pem`, 'utf8');
const ca = fs.readFileSync(`${privLocation}/chain.pem`, 'utf8');

// create credentials
const credentials = {
     key: privateKey,
     cert: certificate,
     ca: ca
};

// starting both http & https servers
const httpServer = http.createServer(app);
const httpsServer = https.createServer(credentials, app);

httpServer.listen(80, () => {
     console.log('HTTP Server running on port 80');
});

httpsServer.listen(443, () => {
     console.log('HTTPS Server running on port 443');
});
```



How to run

You can run the project as follows:

1. Fork or clone the repository to your local. You can use the command below:

```
git clone https://github.com/VPatient/VPatientAPI
```

2. Open any kind of terminal in the directory of the API:

```
cd VPatientAPI # assuming that you are currently out of the folder of the API
npm install # to install required libraries/packages
```

3. In order to run project you are going to need a '.env' file which will be located in the folder of the API. Also if you are planning to use HTTPS then you are going to need cert.pem, chain.pem, fullchain.pem generated by CertBot. By default placing all of them into the root directory of the api should work fine but if you are willing to use CertBot auto-renew certificates (auto-generated .pem files) then change PRIVATE_LOCATION to folder given in installation of CertBot, as an example: 'C:\Certbot\live\api.vpatient.com'

> rename 'example.env' to '.env' fill all variables in the given format

```
MONGO_DB_URL= # mongodb connect url
SECRET=ef28ed74551a40deba54b81df485c83a # password hashing secret
AUTH_SECRET=ef28ed74551a40deba54b81df485c83b # authorizing secret -> to get a user to be an autho
GRADE_SECRET=ef28ed74551a40deba54b81df485c83c # grade secret -> to use grade/create route and sav
PRIVATE_LOCATION=. # absolute or relative path to .pem fiels -> to use SSL
```

4. You have to run:

```
npm start
```

After all these steps your API should be running on http://localhost and https://localhost

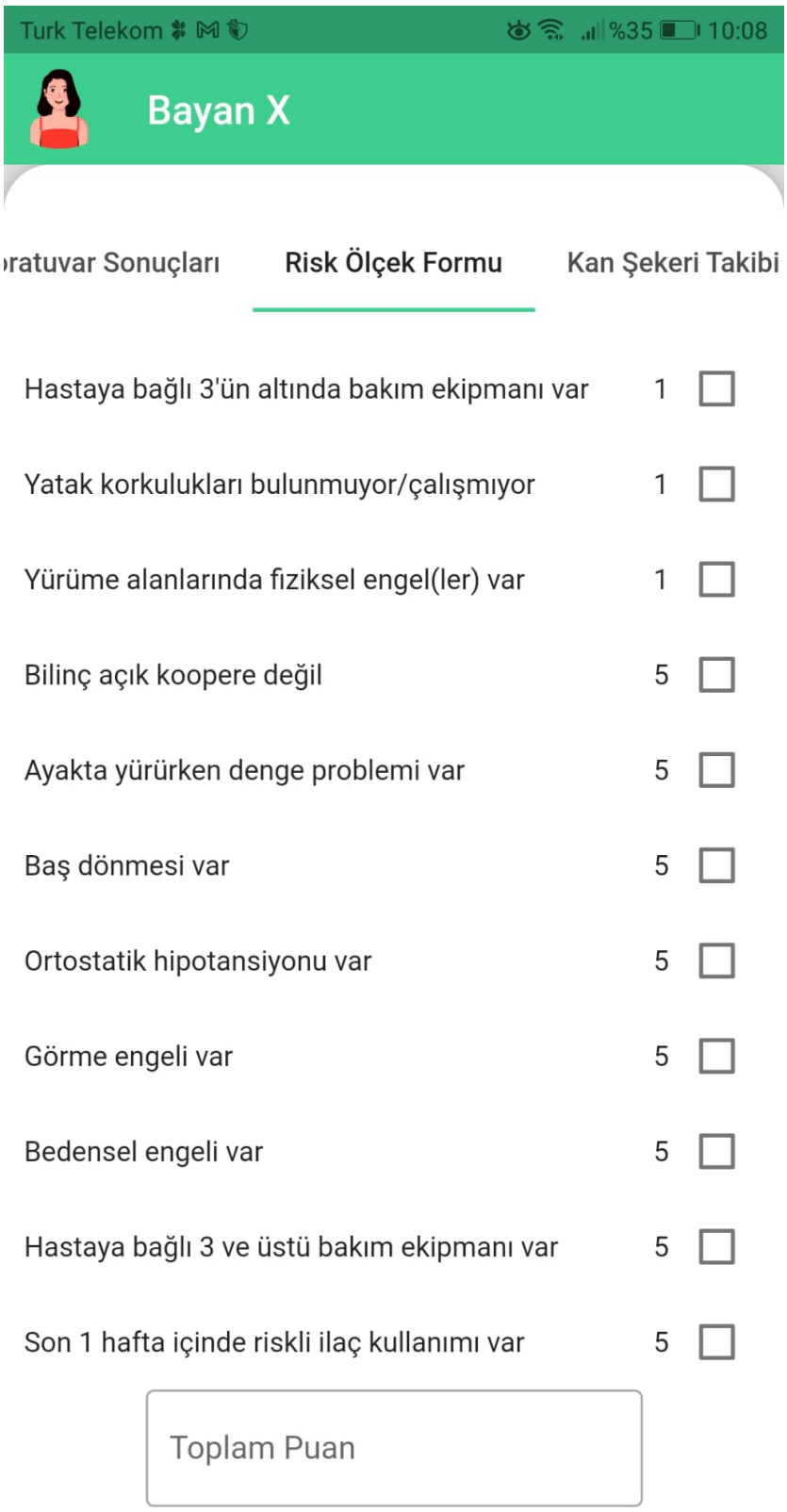
-Our scenario has been enlarged and almost finished.

-Structure and .env file have been changed so do readme.md and example.env in API.

-Mobile application's endpoints have been updated to support HTTPS protocol over api.ugurdindar.com

- IOS signature and configurations have been resolved. Now our application in IOS looks like:

- Fall risk form at mobile application has been done. “Total” input at the bottom of the screen automatically fill according to nurse’s selections. Still working on to get better user experience while using forms and chat screen at the same time.



Task Distribution

- ***Emre Kayacık: Changed API endpoints in Mobile Application to support HTTPS over new API address. Expanded scenario according to requirements of the workflow.***
- ***Uğur Dindar: Adding SSL certificate to both API and server. Creating a new CNAME record (api.ugurdindar.com). Providing a maintainable structure to listen both HTTP and HTTPS protocols. Configuration and signing done of mobile application in IOS.***
- ***İdris Can Bölükbaşı: Trying to get better user experience while using forms and chat screen at the same time. Working on forms (slide up panel) to mount on whole app.***
- ***Oğuzhan Çetin: Fall risk form has been developed. Code refactoring.***
- ***Umutcan Yavuz: UI improvements and refactoring.***
- ***Yunus Emre Korkmaz: Changing the api addresses with new ones to resolve mixed content error.***

Individual Contribution *(state as a percentage for each member, decide according to the contribution of the individual on that week. If a member does not have any contribution, please state)*

Every single team done work of themselves. We can just say that every team contributed evenly to in behalf of progress of the project.

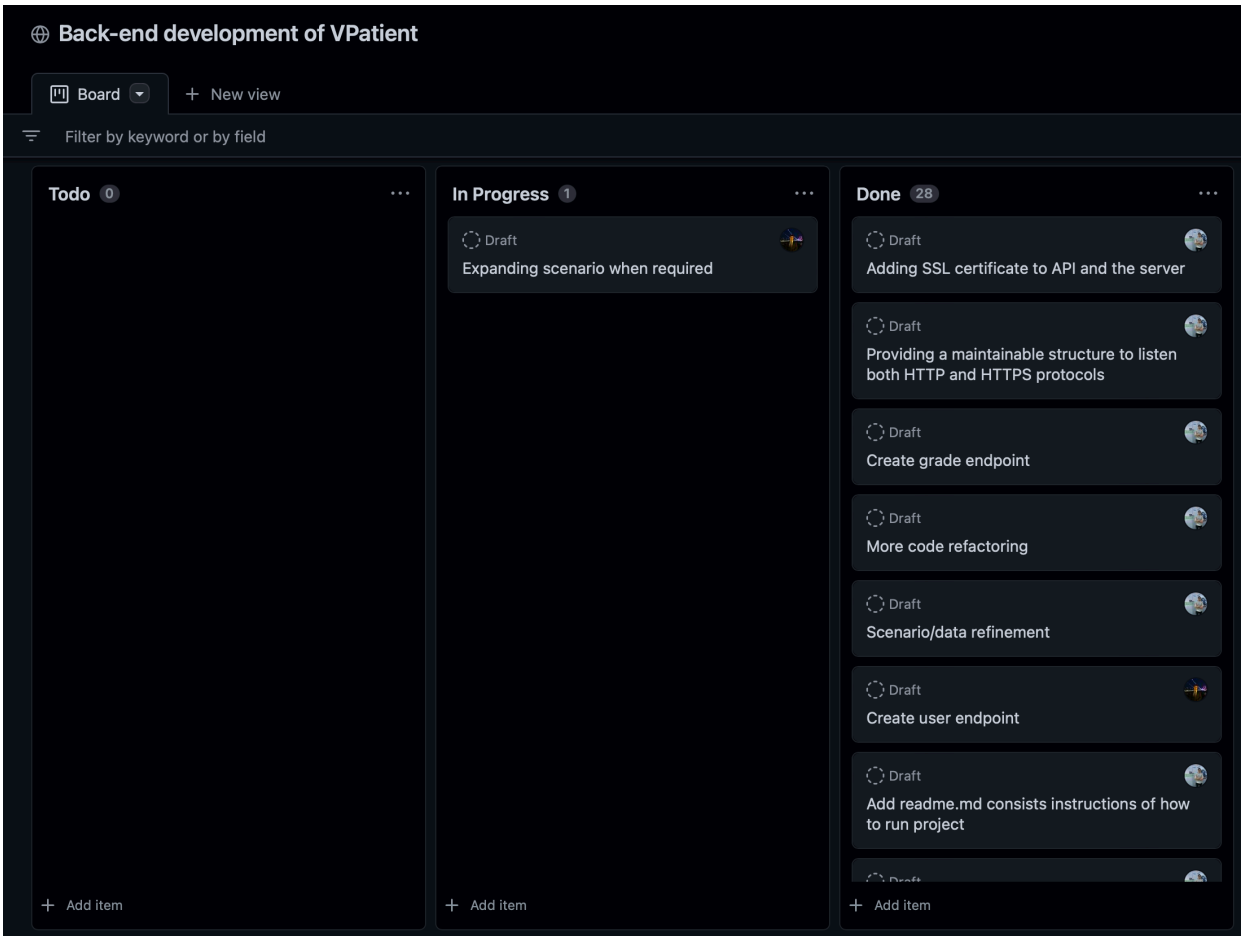
Plans for the Following Week

Every team has their own page to represent plans for the following week and things done in previously weeks.

Back-end Development Team

Plans for the following week:

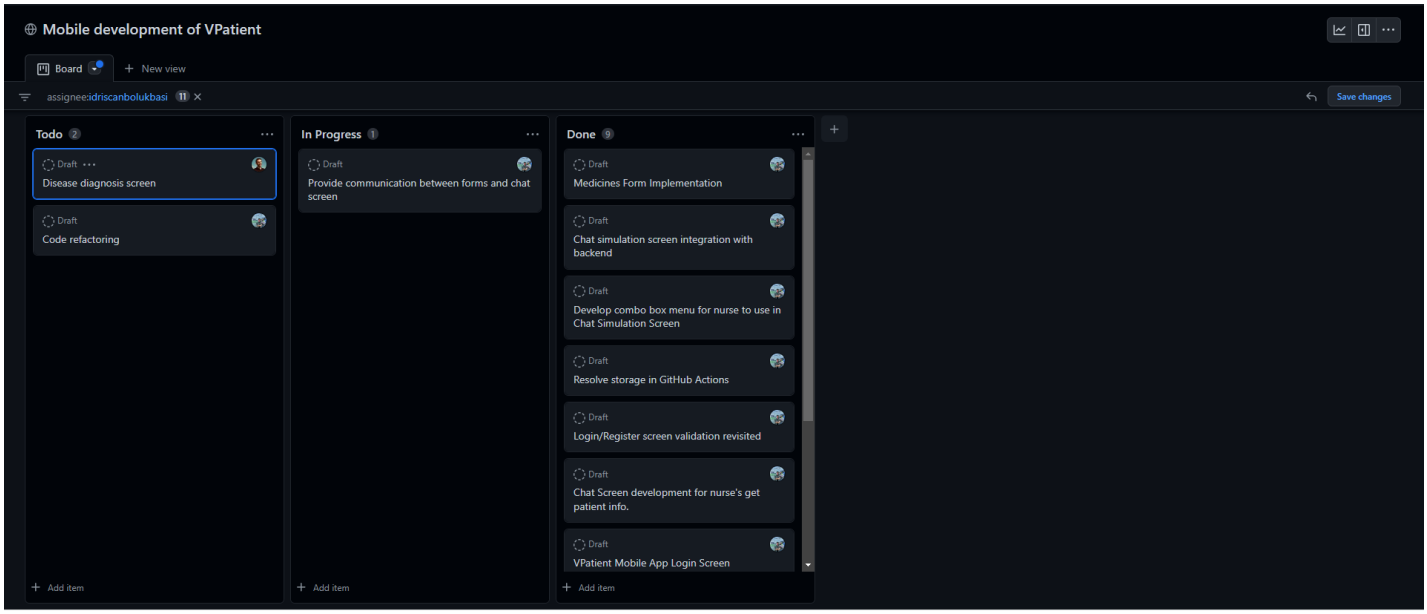
- Expanding scenario when required



Mobile Development Team

Plans for the following week:

- Provide communication between forms and chat screen,
- Disease diagnosis screen
- Code refactoring
- Improve user experience.



Front-end Development Team

Plans for the following week:

- Advanced Information
- Warning Messages