



**BIM423 – SOFTWARE ENGINEERING**

**2022-2023 FALL SEMESTER**



**Virtual Patent**

**Group 12**

**Week 9 / 28.11.2022 – 5.12.2022**

## Members

1. *Uğur Dindar*
2. *Emre Kayacık*
3. *İdris Can Böyükbaşı*
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>

## Works Done

- Created an authorization middleware and implemented in every authorization required endpoints.

```
3 // auth middleware
4 const auth = (req, res, next) => {
5     const token = req.header('auth-token');
6     if (!token) return res.status(401).send('Access Denied');
7     try {
8         const verified = jwt.verify(token, process.env.SECRET);
9         req.user = verified;
10        next();
11    }
12    catch (err) {
13        res.status(400).send('Invalid token')
14    }
15}
16
17 // verify token and authorization middleware
18 const verifyTokenAndAuthorization = (req, res, next) => {
19     auth(req, res, () => {
20         if (req.user.id === req.params.id || req.user.isAdmin) {
21             next();
22         }
23         else {
24             res.status(403).json("You are not authorized to do that!");
25         }
26     });
27 }
28
29 // verify token and admin middleware
30 const verifyTokenAndAdmin = (req, res, next) => [
31     auth(req, res, () => {
32         if (req.user.isAdmin) {
33             next();
34         }
35         else {
36             res.status(403).json("You are not authorized to do that!");
37         }
38     });
39 ]
```

- Created authorize endpoint which is going to be used for authorize logged user.

```

77 // authorize route
78 router.post('/authorize', auth, async (req, res) => {
79   // get auth secret from dotenv
80   let authSecret = req.body.secret;
81
82   // if auth secret is not valid then return
83   if (process.env.AUTH_SECRET !== authSecret) {
84     return res.status(400).send('Wrong parameters.');
85   }
86
87   // get userID then get user by id
88   let userID = req.user._id;
89   const user = await User.findById(userID);
90
91   // check whether the user is already authorized
92   if (user.isAdmin) {
93     return res.status(400).send('User is already authorized!');
94   }
95
96   // update user
97   try {
98     await User.findByIdAndUpdate(userID, {
99       $set: {
100         id: user.id,
101         email: user.email,
102         name: user.name,
103         studentNumber: user.studentNumber,
104         password: user.password,
105         date: user.date,
106         isAdmin: true
107       }
108     }, { new: true }).then(result => {
109       return res.status(200).json(result);
110     });
111   } catch (err) {
112     return res.status(500).json(err);
113   }
114 }
115 );

```

- Code refactoring has been done. A middleware named verifyPatient has been created to reduce redundant codes where patient validation is needed.

<pre> // create patient norton pressure ulcer - router.post("/create", verifyTokenAndAdmin, async (req, res) =&gt; { -   // get patient id -   let patientId = req.body.owner; - -   // validation -   const { error } = idValidation(patientId); -   if (error) return res.status(400) -     .send(error.details[0].message); - -   // get patient -   const patient = await PatientModel.findOne({ _id: patientId }); - -   // return if there is not such that patient -   if (!patient) res.status(500).json(`Cannot find patient with id \${patientId}`); </pre>	<pre> 8 // create patient norton pressure ulcer 9 + router.post("/create", verifyTokenAndAdmin, verifyPatient, async (req, res) =&gt; { 10   // get patient 11   +   let patient = req.patient; </pre>
---	--

- Created a grade endpoint that using a grade secret to provide a robust authorization due to all grading requests are going to be sent from users.

```

// create grade model
router.post("/create", auth, verifyPatient, async (req, res) => {
  // validation
  const { error } = gradeValidation(req.body);
  if (error) return res.status(400)
    .send(error.details[0].message);

  // get patient
  let patient = req.patient;

  // get secret
  let secret = req.body.secret;

  // control secret
  if (process.env.GRADE_SECRET !== secret) return res.status(500).json('Invalid parameters');

  // get user
  const user = await UserModel.findOne({ _id: req.user._id });

  // control if there is one grade given paired with user and patient
  const existingModel = await GradeModel.findOne({
    user: user,
    patient: patient
  });

  // if there is one existing model then inform user
  if (existingModel) return res.status(500).send('User is already graded from this patient');

  // get grade
  let grade = req.body.grade;

  // if not then create a patient grade model
  const gradeModel = new GradeModel({
    user: user,
    patient: patient,
    grade: grade
  });

  // save grade model
  gradeModel.save()
    .then(grade => res.status(200).json(grade))
    .catch(err => res.status(500).json({ message: err }));
});

// list grades of patient by id
router.get("/list", auth, verifyPatient, async (req, res) => {
  // get patient
  let patient = req.patient;

  // get grade model
  const gradeModel = await GradeModel.find({ patient: patient }).populate({
    path: 'user',
    model: UserModel
  });

  // return grade model
  res.status(200).json(gradeModel);
});

// get request of patient by id
router.get("/get", auth, verifyPatient, async (req, res) => {
  // query validation
  const { error } = gradeQueryValidation(req.query);

  if (error) return res.status(400)
    .send(error.details[0].message);

  // get patient
  let patient = req.patient;

  // get user id
  let userId = req.query.userId;

  // get user
  const user = await UserModel.findOne({ _id: userId });

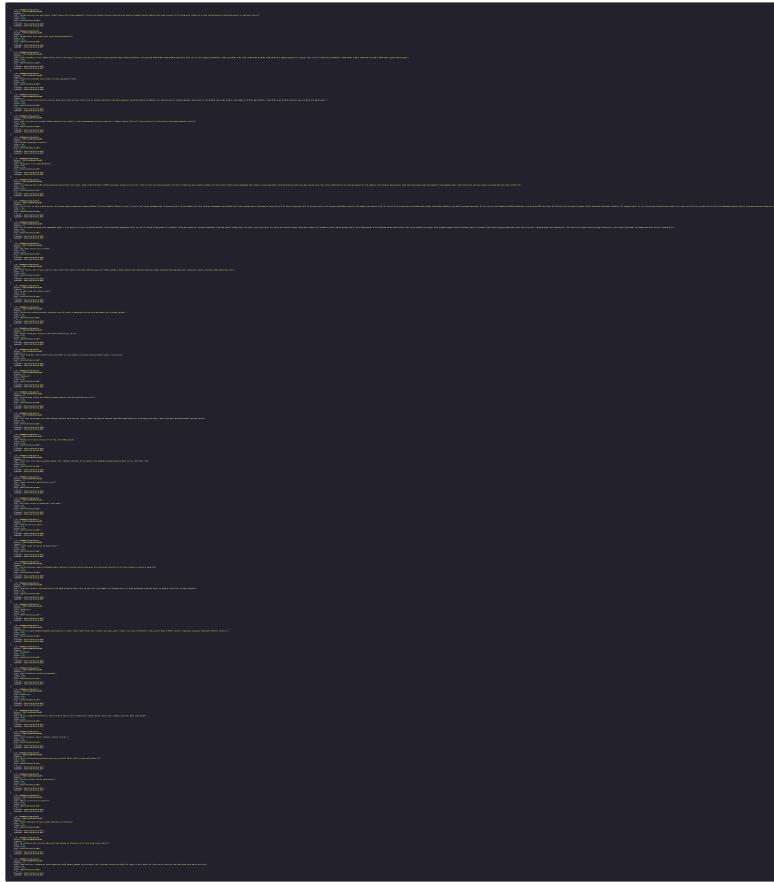
  // return if there is not such that patient
  if (!user) return res.status(500).json(`Cannot find user with id ${userId}`);

  // create grade model
  const gradeModel = await GradeModel.findOne({ patient: patient, user: user });

  // return grade model
  res.status(200).json(gradeModel);
});

```

- Created a comprehensive scenario then implemented it into the system.



- Created readme.md in VPatientAPI.

**README.md**

## 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.

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

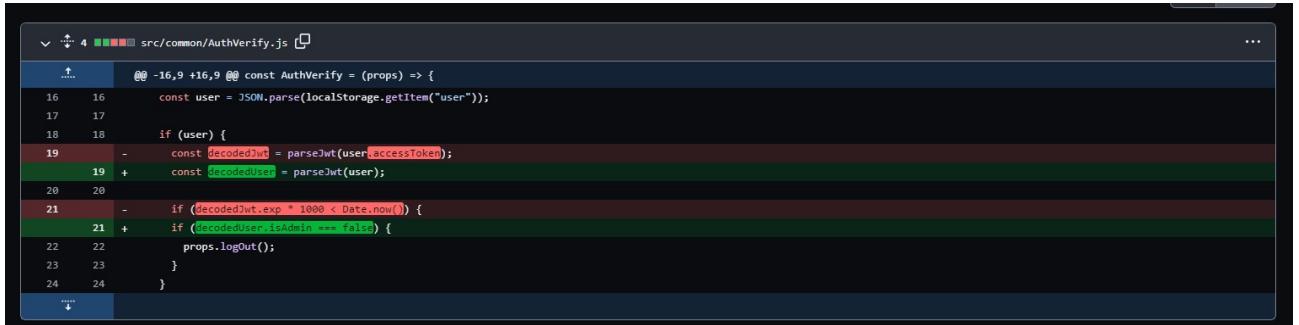
```
MONGO_DB_URL= # mongodb connect url
PORT=5000 # port that is going to be used
SECRET=ef28ed74551a40deba54b81df485c83a # password hashing secret
AUTH_SECRET=ef28ed74551a40deba54b81df485c83b # authorizing secret -> to get a user to be an autho
GRADE_SECRET=ef28ed74551a40deba54b81df485c83c # grade secret -> to use grade/create route and sav
```

4. You have to run:

```
npm start
```

After all these steps your API should be running on <http://localhost:5000>

- Role-based authentication and authorization system implementation has been done simultaneously with back-end development:



```

@@ -16,9 +16,9 @@ const AuthVerify = (props) => {
 16   16     const user = JSON.parse(localStorage.getItem("user"));
 17   17
 18   18     if (user) {
 19 -     const decodedJwt = parseJwt(user.accessToken);
 19 +     const decodedUser = parseJwt(user);
 20   20
 21 -     if (decodedJwt.exp * 1000 < Date.now()) {
 21 +     if (decodedUser.isAdmin === false) {
 22       22       props.logOut();
 23     23   }
 24   24 }

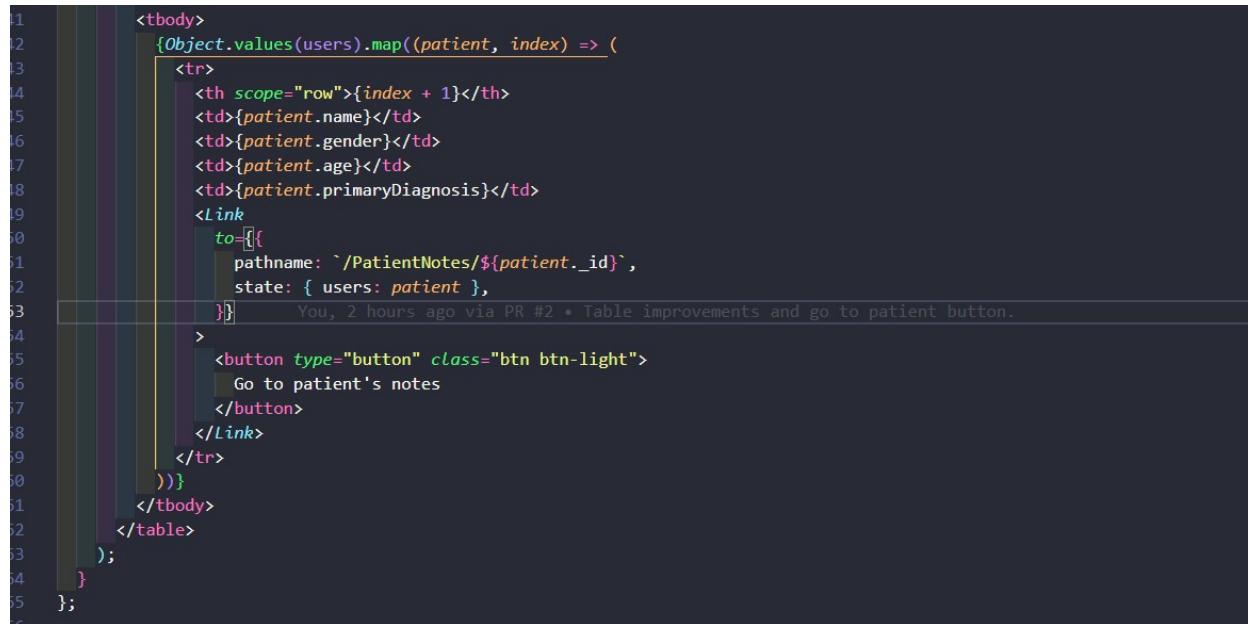
```

- Patient Table Implementation:



#	Name	Gender	Age	Primary Diagnosis	Patient Notes
1	Bay X	Erkek	75	Kolorektal Kanser (Kalin Bağırsak Kanseri)	<a href="#">Go to patient's notes</a>
2	Bayan Z	Kadin	35	Kolorektal Kanser (Kalin Bağırsak Kanseri)	<a href="#">Go to patient's notes</a>
3	Bayan X	Kadin	35	Kolorektal Kanser (Kalin Bağırsak Kanseri)	<a href="#">Go to patient's notes</a>
4	Bayan C	Kadin	35	Kolorektal Kanser (Kalin Bağırsak Kanseri)	<a href="#">Go to patient's notes</a>

- Also “Go to patient’s notes” button has been added.



```

11   <tbody>
12     {Object.values(users).map((patient, index) =>
13       <tr>
14         <th scope="row">{index + 1}</th>
15         <td>{patient.name}</td>
16         <td>{patient.gender}</td>
17         <td>{patient.age}</td>
18         <td>{patient.primaryDiagnosis}</td>
19         <Link
20           to={{
21             pathname: `/PatientNotes/${patient._id}`,
22             state: { users: patient },
23           }}
24           You, 2 hours ago via PR #2 • Table improvements and go to patient button.
25         >
26           <button type="button" class="btn btn-light">
27             Go to patient's notes
28           </button>
29         </Link>
30       </tr>
31     ))}
32   </tbody>
33 </table>
34 );
35 };

```

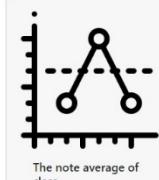
- Go to patient button links to a new page with given url: “/PatientNotes/\$patientId”. According to this, every patient has a unique page with his/her id’s. On this part we worked simultaneously with back-end development team and this helps us to get necessary data easily.

VPatient Home User LogOut

Patient Id: 636577a1a9304eb1637daa00

#	Name	Student Number	Date	Grade
1	Uğur Dindar	33127209018	2022-11-30T19:36:45.738Z	100
2	Uğur Dindar	33127209018	2022-11-30T19:36:52.974Z	100
3	Uğur Dindar	33127209018	2022-12-01T08:47:58.916Z	100
4	Uğur Dindar	33127209018	2022-12-01T08:49:16.292Z	100

[Go back](#)



The note average of class  
100



The highest grade of class  
Uğur Dindar  
100

```

9
10 const PatientNotes = () => {
11   const { state } = useLocation();
12
13   var patientId = window.location.pathname.split("/")[2];
14
15   const [grades, setGrades] = useState("");
16
17   useEffect(() => {
18     UserService.getListOfGradesByPatientId(patientId).then(

```

user.service.js C:\Users\dolph\Desktop\VPatient-Frontend\VPatientDashboard\src\services - Definitions (2)

```

9
10
11   const getPatientList = () => {
12     return axios.get(API_URL + "patient/list", { headers: authHeader() });
13   };
14
15   const getListOfGradesByPatientId = (id) => {
16     return axios.get(API_URL + "grade/list?id=" + id, { headers: authHeader() });
17   };
18
19   const getUserBoard = () => {
20     return axios.get(API_URL + "user", { headers: authHeader() });
21   };

```

Also note average of class and the highest grade of class can be observed from this page.

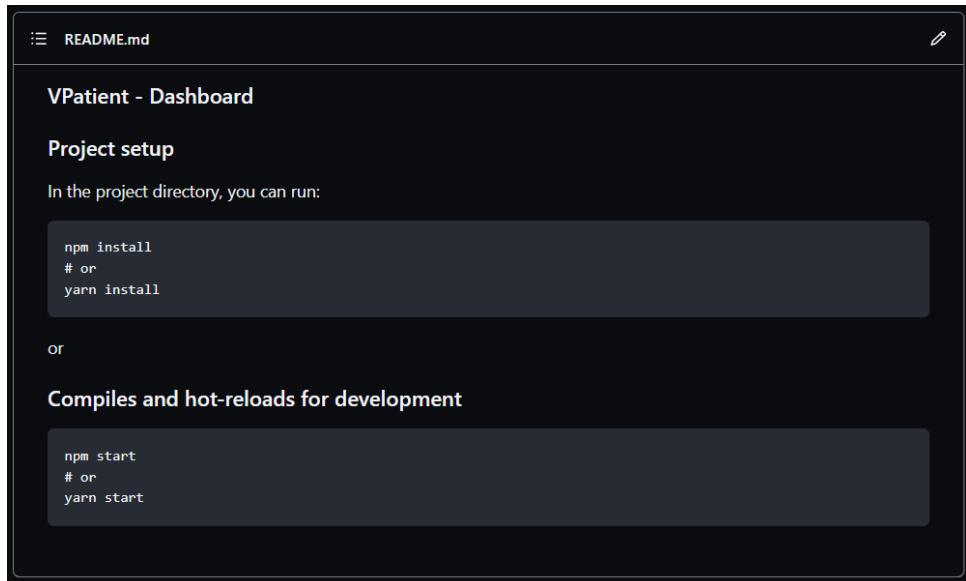
- If there are no grades for a patient:

VPatient Home User LogOut

There are no grades for this patient:  
6366ef3d48009a81ef261ead

[Go back](#)

- README.md created in VPatientDashboard



README.md

## VPatient - Dashboard

### Project setup

In the project directory, you can run:

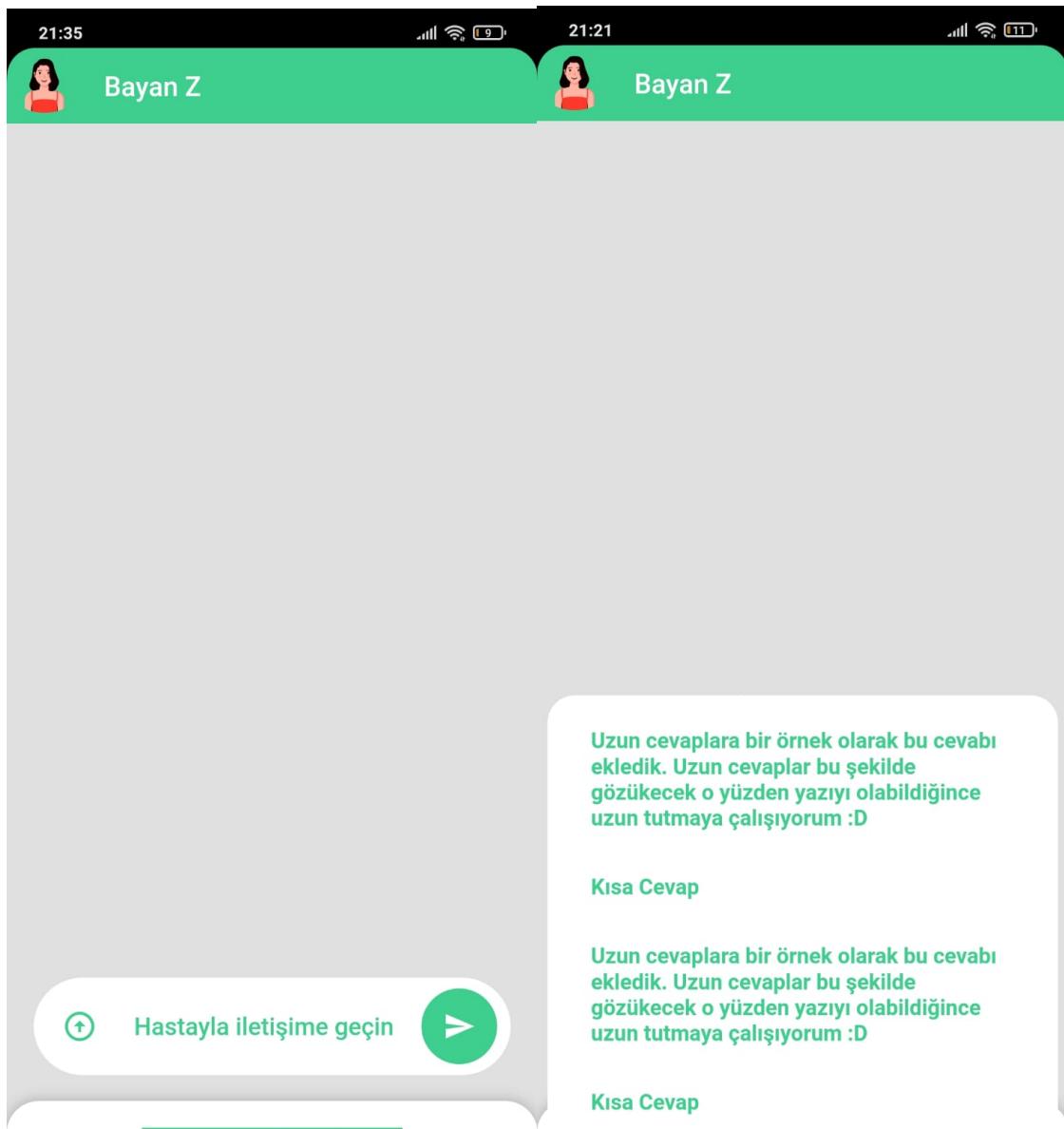
```
npm install
# or
yarn install
```

or

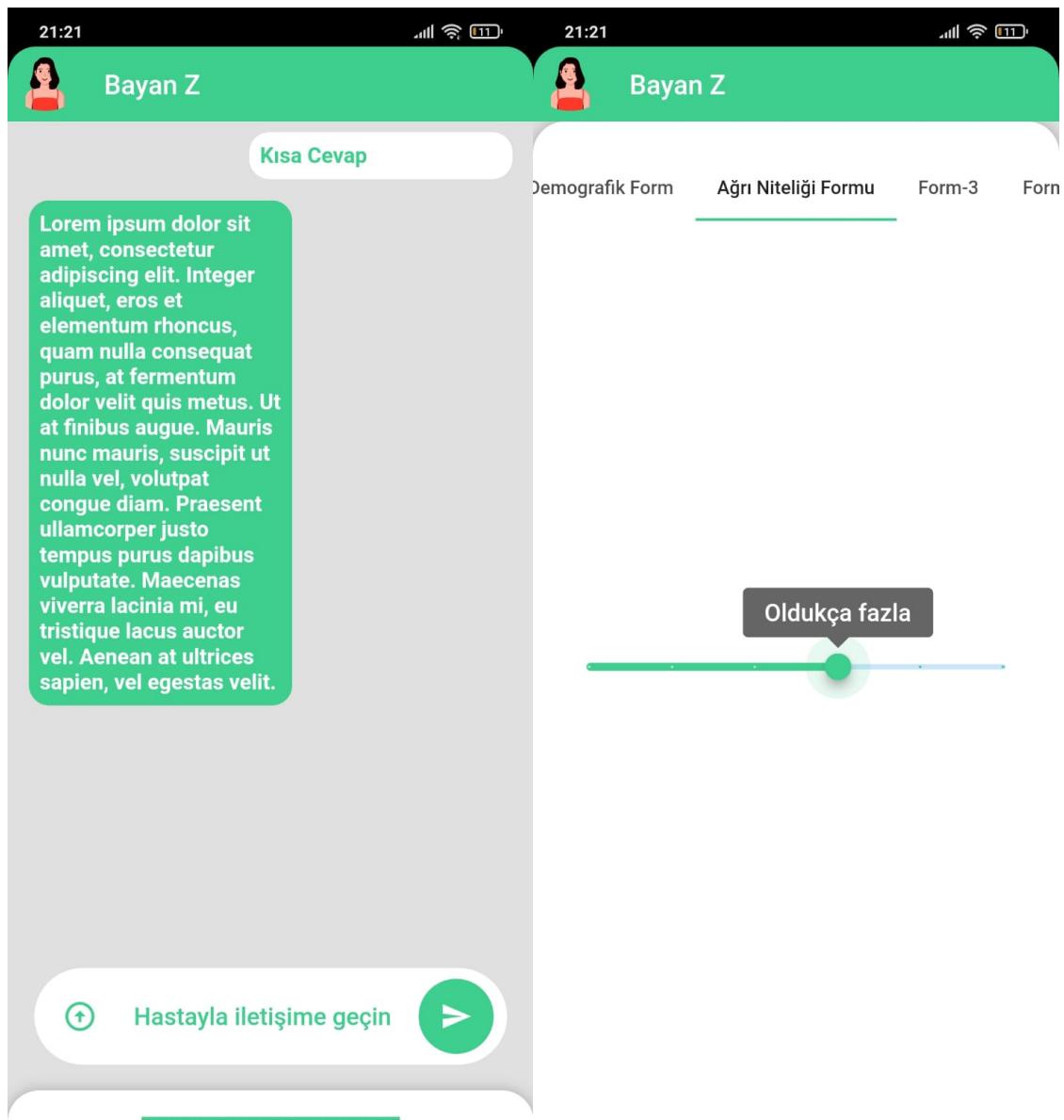
### Compiles and hot-reloads for development

```
npm start
# or
yarn start
```

- UI improvements in the whole project. Added combobox to provide communication that suits for given scenario.



- We are continuing to implement necessary forms to be ready to be fulfilled.



- We redesigned our CI/CD process and management in GitHub Actions to minimize storage usage.

You've used 100% of included services for the VPatient account



### Shared storage usage

Hello! We wanted to provide you with an update on your usage and spending.

Storage used	692MB of 512MB included
--------------	-------------------------

**You've used 100% of included services for GitHub Storage (GitHub Actions and Packages)**

To continue using Actions & Packages uninterrupted, update your spending limit.

Your usage will reset on January 01, 2023.

[Update spending limit](#)

## Artifact and log retention

Choose the repository settings for artifacts and logs. [Learn more.](#)

### Artifact and log retention

3 days

Save

## Task Distribution

- *Emre Kayacık: User endpoint has been created which is going to be used to get all users registered to the system whether admin or nurse. Default user definition has been updated to control authorization. Implemented authorization middleware into every single endpoint that must be authorized. Code refactoring has been done.*
- *Uğur Dindar: Created a grade endpoint that uses a grade secret to provide a robust authorization due to all grading requests are going to be sent from users. Also created an authorize endpoint works similarly to grade endpoint to save logged user 'authorized'. Created a comprehensive scenario then implemented it into the system. Created readme in public repository.*
- *İdris Can Bölükbaşı: Chat simulation screen UI improvements, added predefined messages combobox to provide a better communication between nurse and patient according to scenario. Shared storage usage error has been fixed in GitHub Actions.*
- *Oğuzhan Çetin: Working on implementing necessary forms into chat simulation screen. Code refactoring to provide a more robust and maintainable structure.*
- *Umutcan Yavuz: Role-based authorization and authentication implementation. Preparation of README.md. Design of pages and general structures.*
- *Yunus Emre Korkmaz: Patient table and Patient notes page implementations. User Service implementations for getting students' grades.*

**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:

- More code refactoring.
- Expanding scenario if needed.

⊕ Back-end development of VPatient

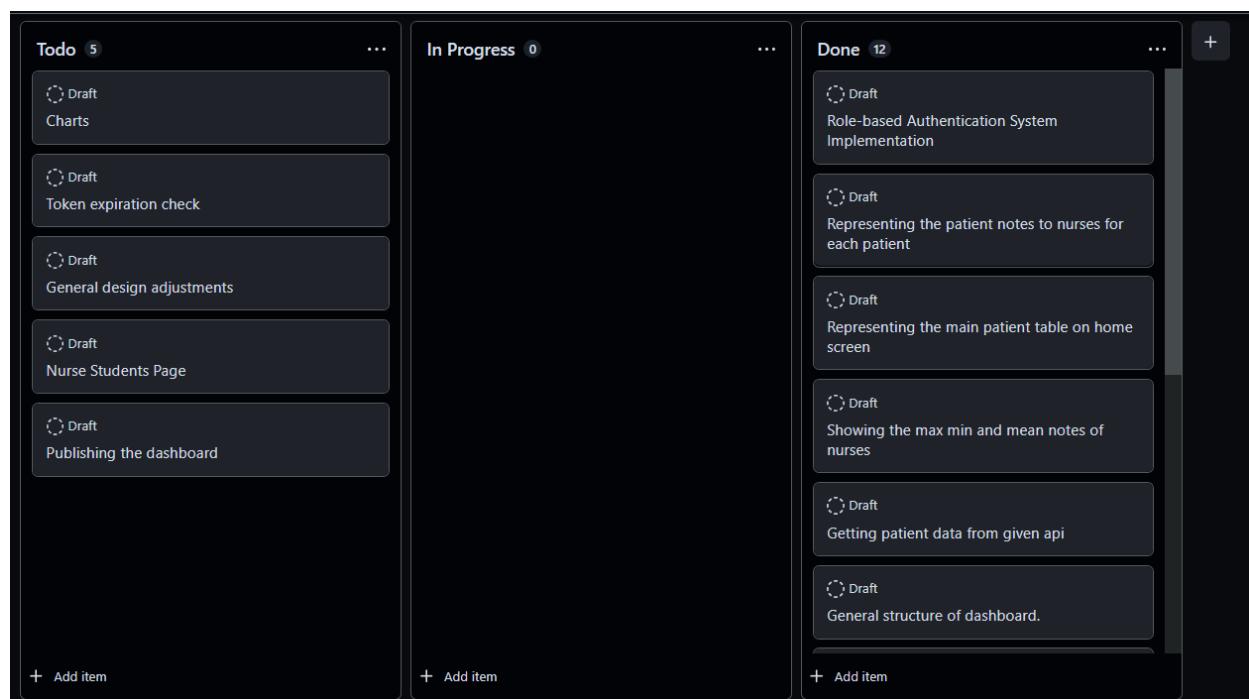
Board + New view Filter by keyword or by field

Todo (2)	In Progress (0)	Done (24)
Draft Expanding scenario if needed		Draft Create grade endpoint
Draft More code refactoring		Draft Create user endpoint
		Draft Add readme.md consists instructions of how to run project
		Draft Provide an authorization endpoint to add user as 'authorized' -a.k.a admin-
		Draft Add authorization middleware into every single endpoint that must be authorized
+ Add item	+ Add item	+ Add item

## Front-end Development Team

Plans for the following week:

- Token expiration check.
- Charts.
- General design adjustments.
- Students Page.
- Publishing the dashboard



## Mobile Development Team

Plans for the following week:

- Integration with back-end in chat simulation screen.
- Implementing all necessary forms at least in UI.
- Implementing landing page for chat simulation screen to inform nurse about the processes.

Board: Mobile development of VPatient

Filter by keyword or by field

Column	Items
Todo	2
In Progress	1
Done	11

**Todo (2)**

- Draft: Chat simulation screen integration with backend
- Draft: Landing page for chat simulation screen

**In Progress (1)**

- Draft: Develop necessary forms to use at Chat Simulation Screen

**Done (11)**

- Draft: Develop combo box menu for nurse to use in Chat Simulation Screen
- Draft: Code refactoring
- Draft: Resolve storage in GitHub Actions
- Draft: UI overlay bug fix
- Draft: Login/Register screen validation revisited
- Draft: Chat Screen development for nurse's get patient info.
- Draft: VPatient Mobile App Login Screen Implementation

+ Add item