BIM423 – SOFTWARE ENGINEERING

2022-2023 FALL SEMESTER

Virtual Patent

Group 12

Week 9 / 28.11.2022 – 5.12.2022

Members

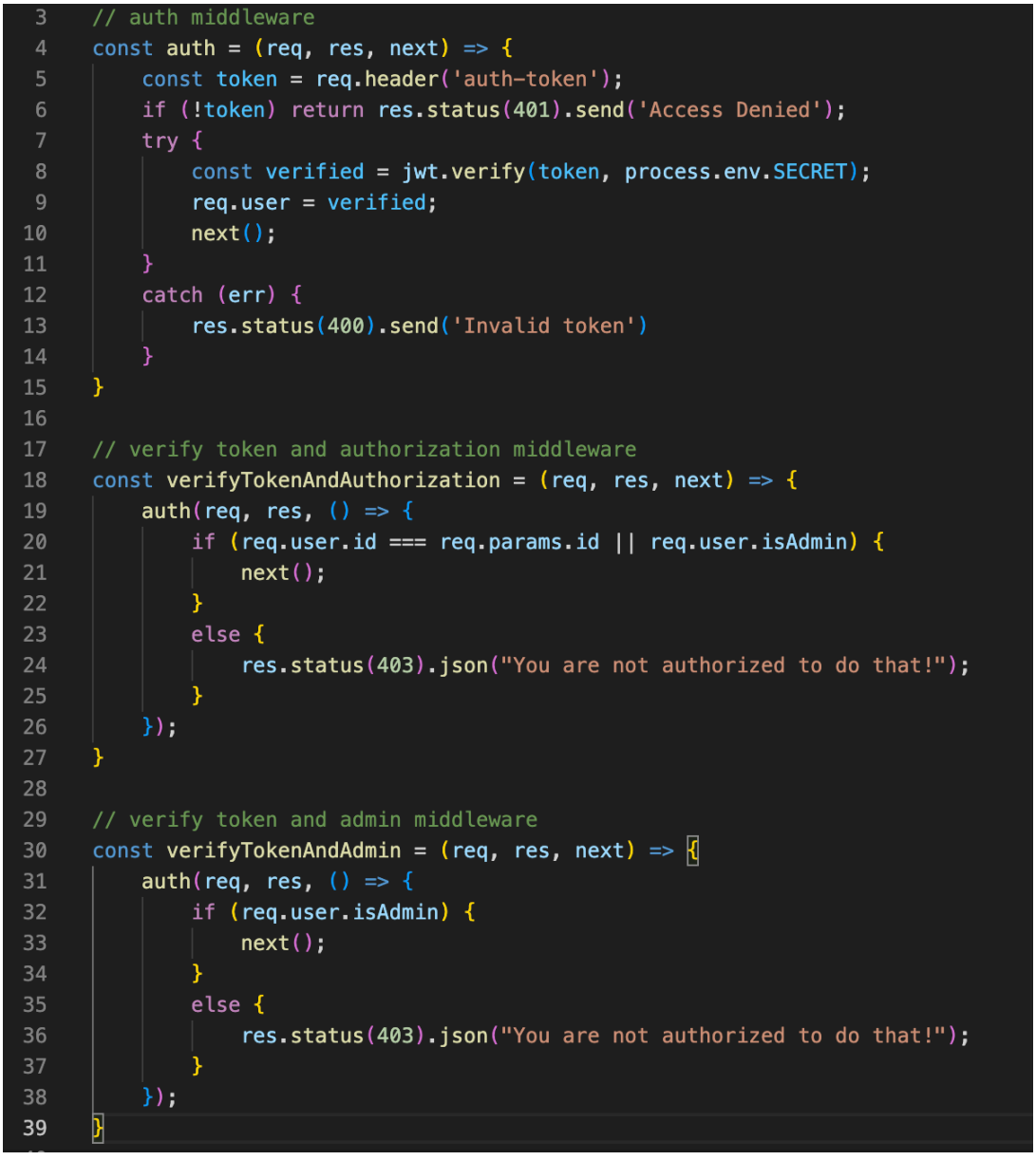
1. ***Uğur Dindar***
2. ***Emre Kayacık***
3. ***İdris Can Bölükbaşı***
4. ***Oğuzhan Çetin***
5. ***Umutcan Yavuz***
6. ***Yunus Emre Korkmaz***

Links

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

Works Done

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



```
// auth middleware
const auth = (req, res, next) => {
    const token = req.header('auth-token');
    if (!token) return res.status(401).send('Access Denied');
    try {
        const verified = jwt.verify(token, process.env.SECRET);
        req.user = verified;
        next();
    }
    catch (err) {
        res.status(400).send('Invalid token')
    }
}

// verify token and authorization middleware
const verifyTokenAndAuthorization = (req, res, next) => {
    auth(req, res, () => {
        if (req.user.id === req.params.id || req.user.isAdmin) {
            next();
        }
        else {
            res.status(403).json("You are not authorized to do that!");
        }
    });
}

// verify token and admin middleware
const verifyTokenAndAdmin = (req, res, next) => {
    auth(req, res, () => {
        if (req.user.isAdmin) {
            next();
        }
        else {
            res.status(403).json("You are not authorized to do that!");
        }
    });
}
```

- 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  });
```

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

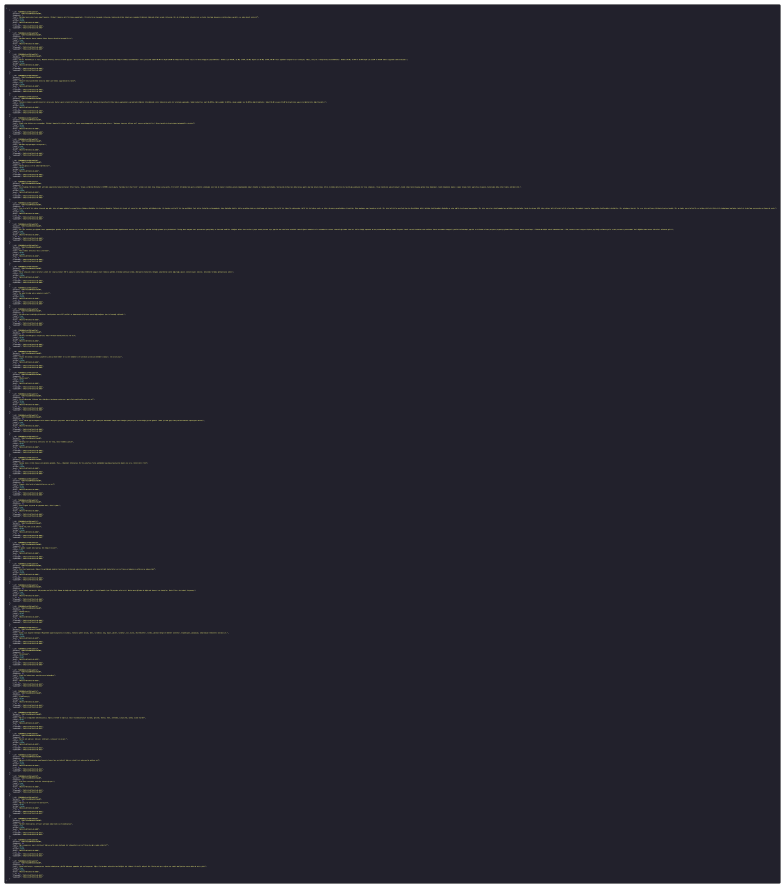
```
// create patient norton pressure ulcer
- router.post("/create", verifyTokenAndAdmin, async (req, res)
  => {
-     // get patient id
-     let patientId = req.body.owner;
-
-     // validation
-     const { error } = idValidation(patientId);
-     if (error) return res.status(400)
-         .send(error.details[0].message);
-
      // 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}`);
```

```
8    // create patient norton pressure ulcer
9  + router.post("/create", verifyTokenAndAdmin, verifyPatient,
     async (req, res) => {

10       // get patient
11 +     let patient = req.patient;
```

- 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

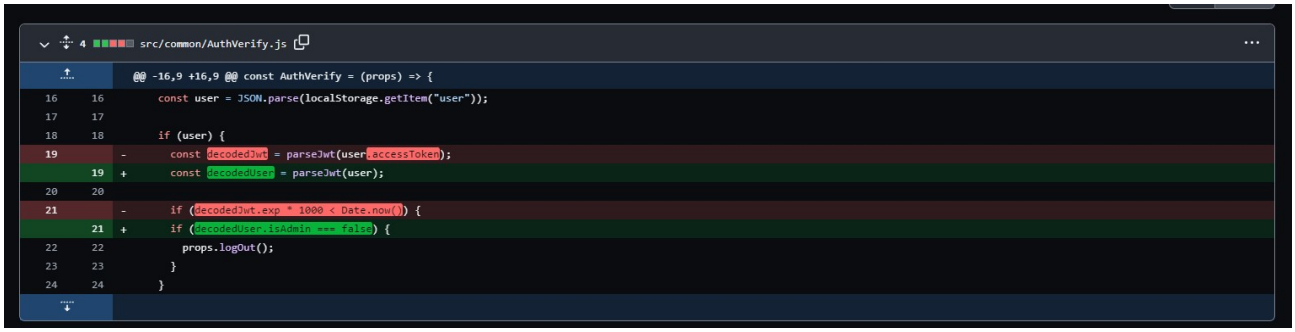
```
MONGO_DB_URL= # mongodb connect url
PORT=5000 # port that is going to be used
SECRET=ef28ed74551a40deba54b81df485c83a # password hashing secret
AUTH_SECRET=ef28ed74551a40deba54b81df485c83b # authorizing secret -> to get a user to be an autho
GRADE_SECRET=ef28ed74551a40deba54b81df485c83c # grade secret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

4. You have to run:

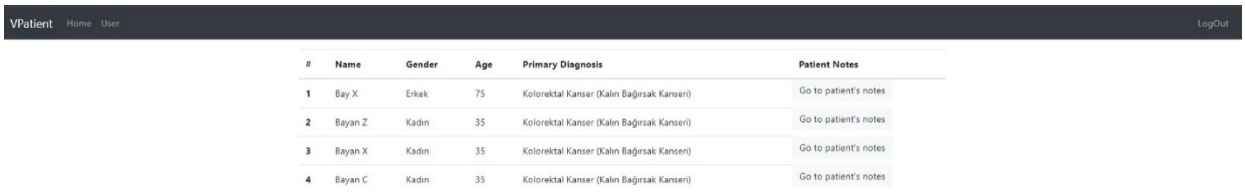
```
npm start
```

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:



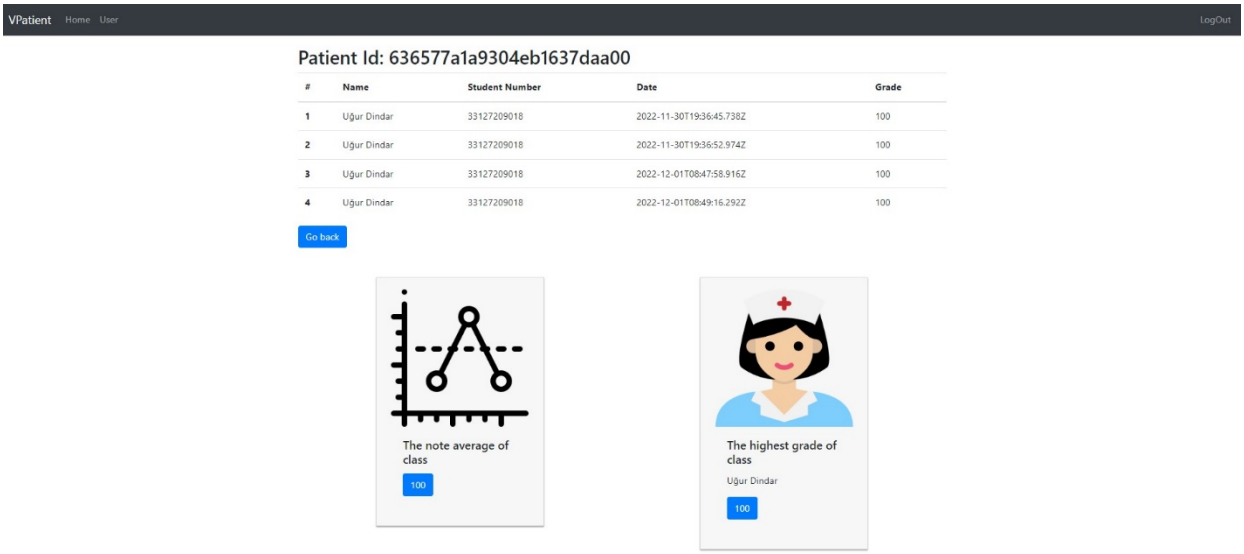
- Patient Table Implementation:

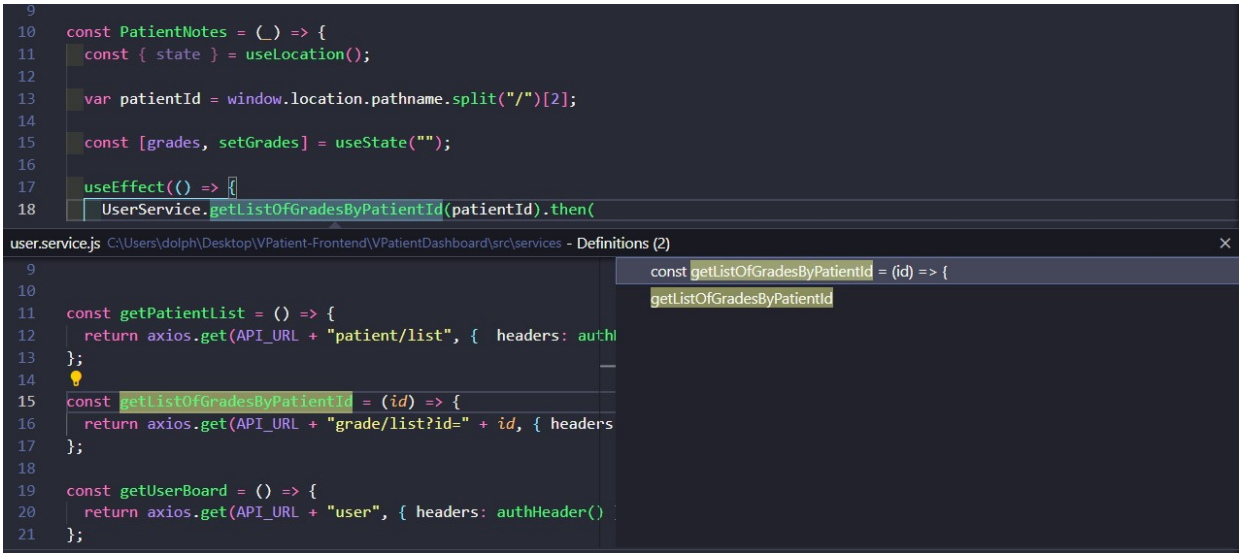


- Also “Go to patient’s notes” button has been added.

```
        <tbody>
          {Object.values(users).map((patient, index) => (
            <tr>
              <th scope="row">{index + 1}</th>
              <td>{patient.name}</td>
              <td>{patient.gender}</td>
              <td>{patient.age}</td>
              <td>{patient.primaryDiagnosis}</td>
              <Link
                to={{
                  pathname: `/PatientNotes/${patient._id}`,
                  state: { users: patient },
                }}      You, 2 hours ago via PR #2 • Table improvements and go to patient button.
              >
                <button type="button" class="btn btn-light">
                  Go to patient's notes
                </button>
              </Link>
            </tr>
          ))}
        </tbody>
      </table>
    );
  }
};
```

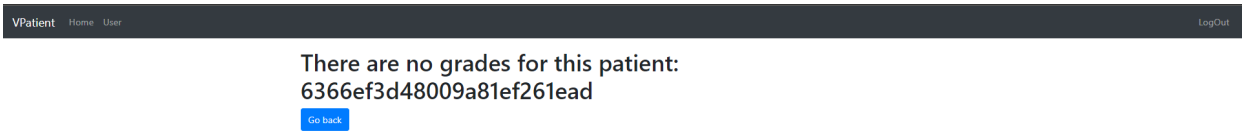
- Go to patient button links to a new page with given url: “/PatientNotes/$patientId ”. According to this, every patient has a unique page with his/her id’s. On this part we worked simultaneously with back-end development team and this helps us to get necessary data easily.



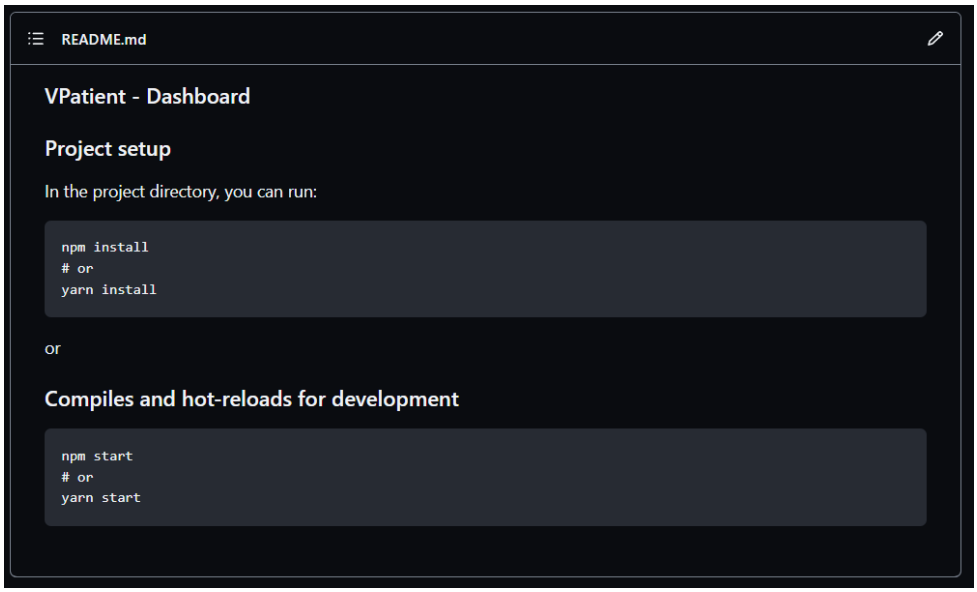


Also note average of class and the highest grade of class can be observed from this page.

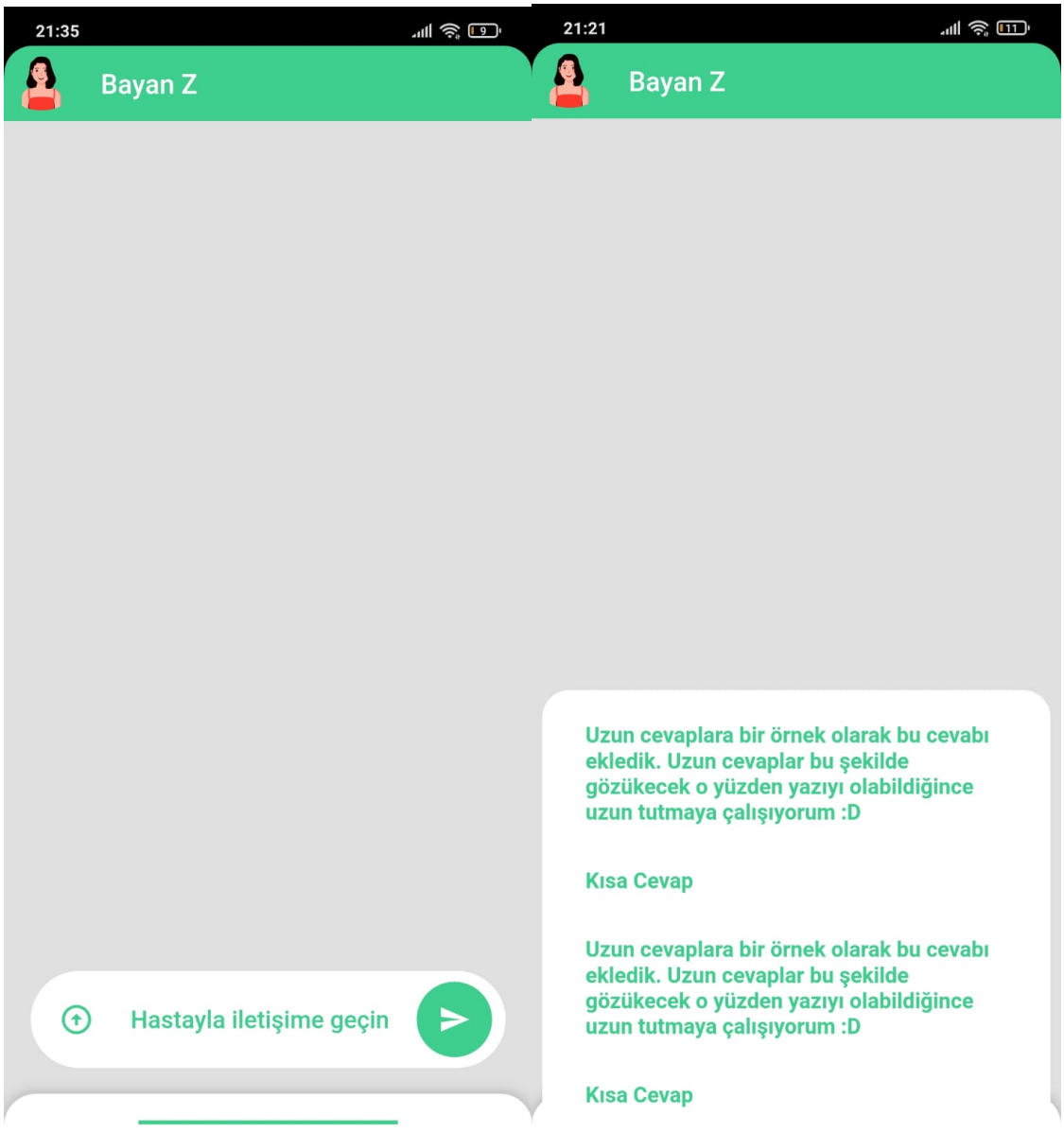
- If there are no grades for a patient:



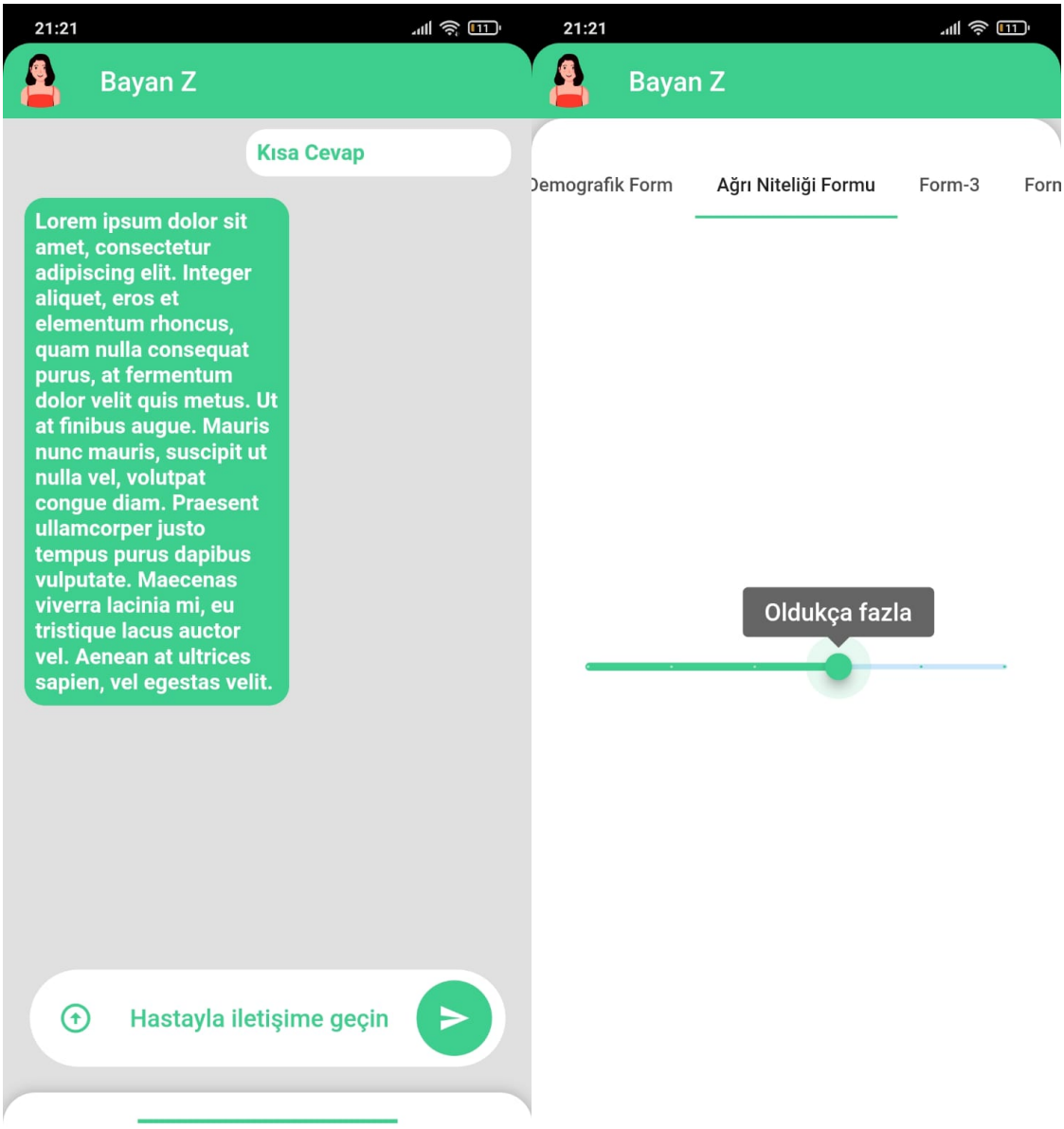
- README.md created in VPatientDashboard



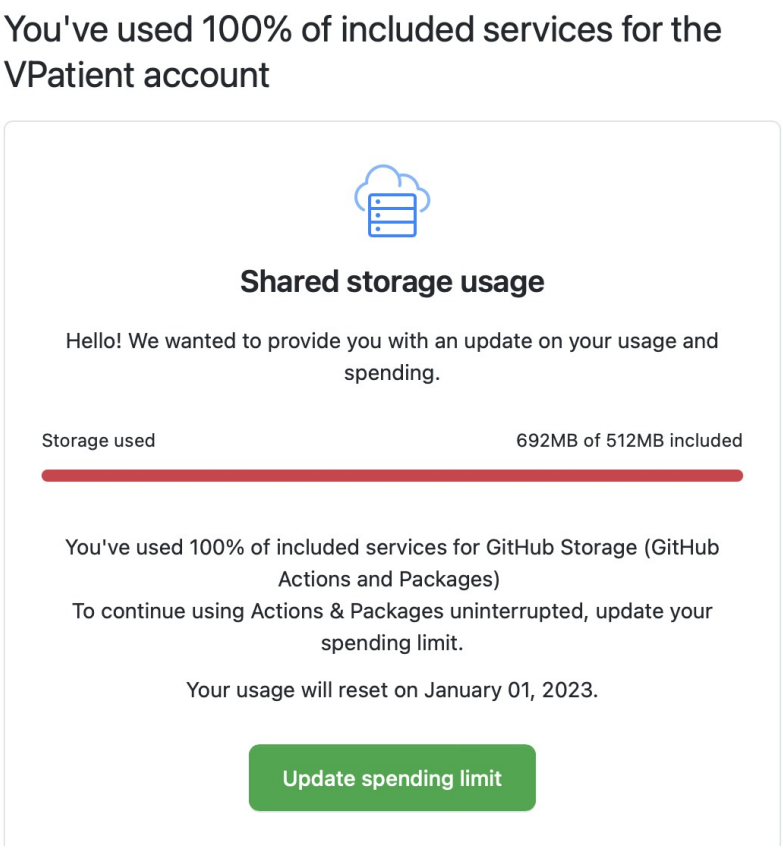
- UI improvements in the whole project. Added combobox to provide communication that suits for given scenario.

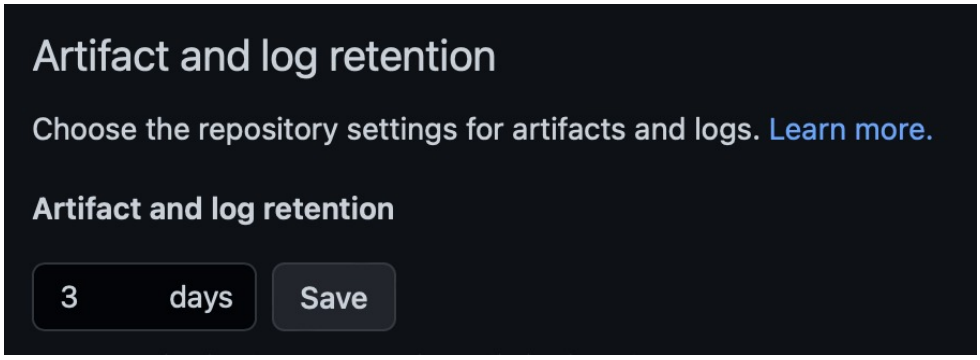


- We are continuing to implement necessary forms to be ready to be fulfilled.



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





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 using 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.

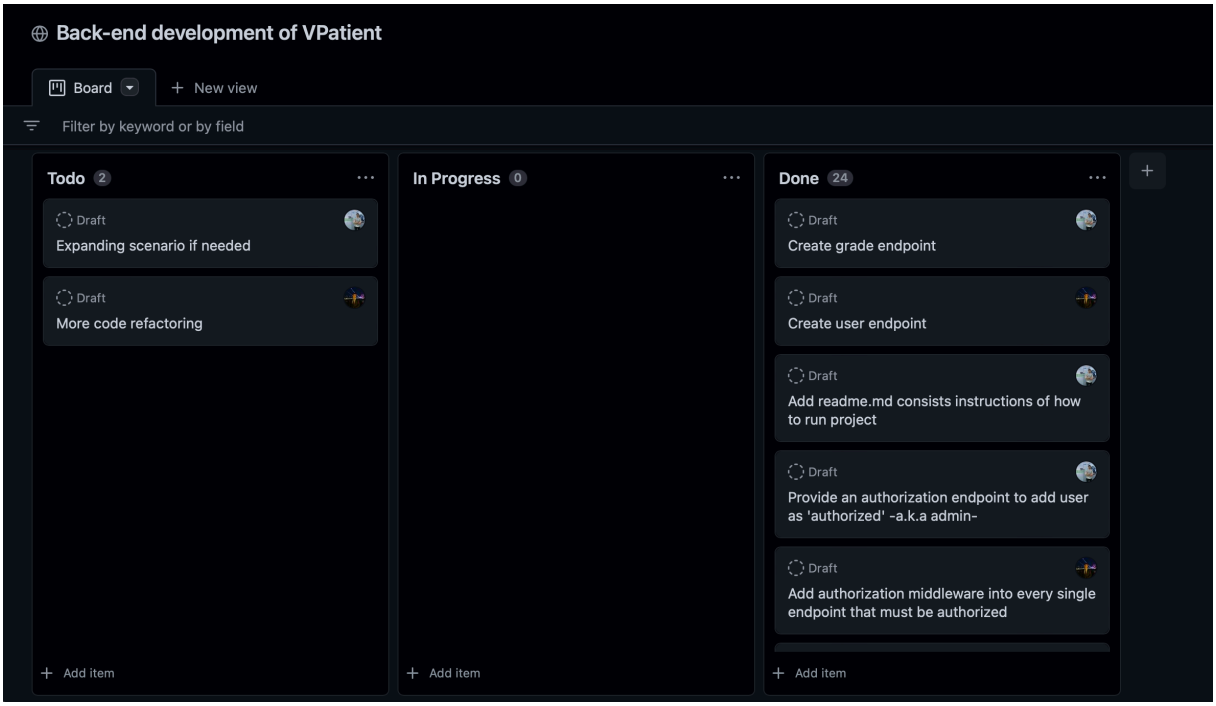
Plans for the Following Week

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:

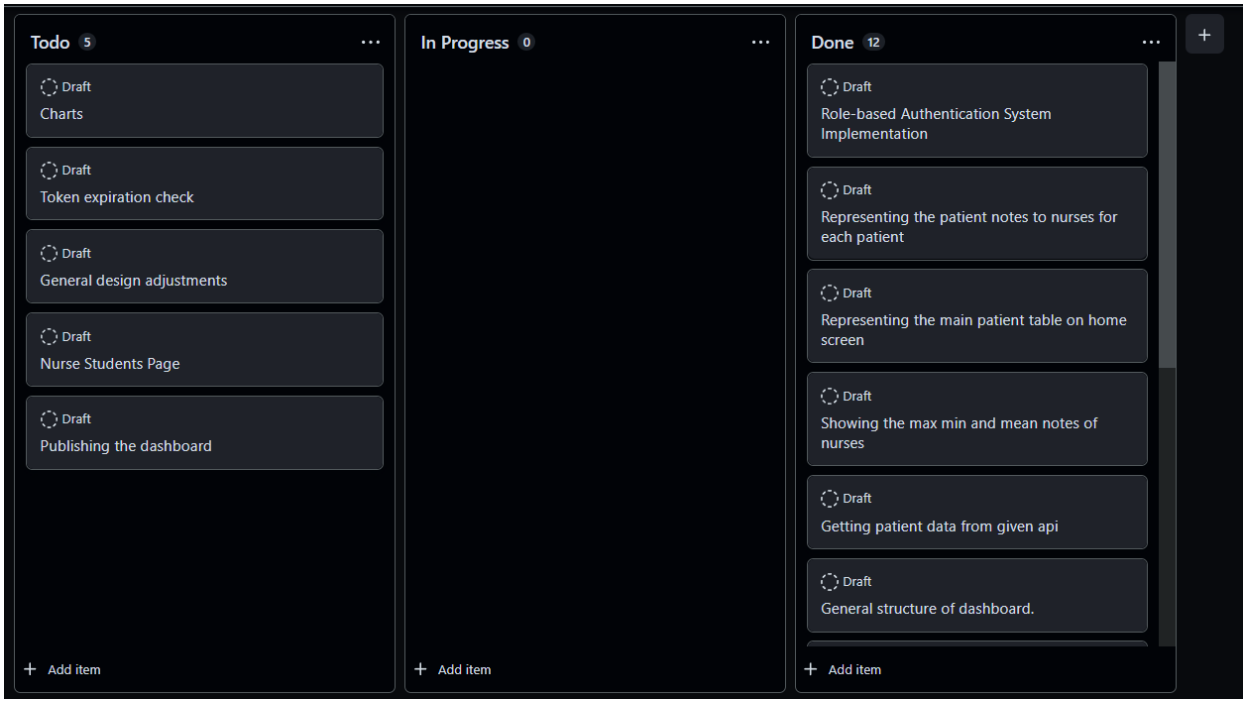
- More code refactoring.
- Expanding scenario if needed.



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.