



BIM423 – SOFTWARE ENGINEERING
2022-2023 FALL SEMESTER



Virtual Patent

Group 12

Week 5 / 30.10.2022 – 7.11.2022

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

- Authentication and authorization processes have been developed. Login and register endpoints have been created and added to route. And also, a middleware developed to control if client has true token.

```
// register to the system
router.post('/register', async (req, res) => {
  // validation
  const { error } = registerValidation(req.body);
  if (error) return res.status(400)
    .send(error.details[0].message);

  // checking if email is already taken
  User.findOne({ email: req.body.email })
    .then(emailExists => {
      if (emailExists) return res.status(400).send('Email already exists.')
    })

  // hash the password
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(req.body.password, salt);

  const user = new User({
    name: req.body.name,
    studentNumber: req.body.studentNumber,
    email: req.body.email,
    password: hashedPassword
  })
  user.save()
    .then(savedUser => res.send({ user: savedUser._id }))
    .catch(err => res.send({ 'messages': err }));
});

// login to the system
router.post('/login', async (req, res) => {
  // validation
  const { error } = loginValidation(req.body);
  if (error) return res.status(400)
    .send(error.details[0].message);

  // checking if email is on the database
  const user = await User.findOne({ email: req.body.email });
  if (!user) return res.status(400).send('Wrong e-mail or password.');
```

```
  // is password correct?
  const validPassword = await bcrypt
    .compare(req.body.password,
      user.password)
  if (!validPassword) return res.status(400).send('Wrong e-mail or password.');
```

```
  // create and assign jwt
  const token = jwt.sign(
    {
      _id: user._id,
      name: user.name,
      studentNumber: user.studentNumber,
      email: user.email
    }, process.env.SECRET);
  res.header('auth-token', token).send(token);
});

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')
  }
}
```

- Collection of all necessary endpoints have been created as a collection on Postman including data need to be inserted. Since all endpoints have created as a collection they can be exported/imported. We designed that collection in a structure such that you will only need to enter required auth-key once.

The screenshot displays the Postman interface for the 'VPatient API' collection. On the left, a sidebar lists the collection's structure, including folders for 'Authentication', 'Patient', and 'Scenario', each containing several endpoints with their respective HTTP methods (e.g., POST, GET). The main panel shows the 'Authorization' tab for the selected endpoint, 'POST {{apiURL}}/patient/medicine/create'. The authorization type is set to 'API Key'. A warning box states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)'. The 'Key' is 'auth-token' and the 'Value' is 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ'. The 'Add to' dropdown is set to 'Header'. Below this, the 'Body' tab is active, showing a JSON payload for creating a patient's medicine. The payload is a JSON array of objects, each representing a medicine with fields like name, dose, time, reason, and duration.

```
1  [
2    {
3      "name": "Delix tb.",
4      "dose": "10mg 1x1",
5      "time": "09:00",
6      "reason": "Hipertansiyon",
7      "duration": "8 yıl"
8    },
9    {
10     "name": "Coraspin cap.",
11     "dose": "100mg 1x1",
12     "time": "09:00",
13     "reason": "Hipertansiyon",
14     "duration": "8 yıl"
15   },
16   {
17     "name": "Novamix İnsülin",
18     "dose": "10 Ünite 3x1",
19     "time": "07:00-12:00-18:00",
20     "reason": "Diabetes Mellitus",
21     "duration": "10 yıl"
22   },
23   {
24     "name": "Lantus İnsülin",
25     "dose": "14 Ünite 1x1",
26     "time": "23:00",
27     "reason": "Diabetes Mellitus",
28     "duration": "10 yıl"
29   },
30   {
31     "name": "Famodin tb.",
32     "dose": "60mg 1x1",
33     "time": "08:00",
34     "reason": "Mide ağrısı için",
35     "duration": "10 yıl"
36   }
37 ]
38
39
40
```

- File transform functionality that enables CI/CD to work continuously independently from API's state of work.

```

✓ Run Copy-Item -Force -Recurse -Verbose -Path 'C:\actions-runner_work\VPatientAPI\VPatientAPI\*' -Destination 'C:\VPatient\'

1 ▶ Run Copy-Item -Force -Recurse -Verbose -Path 'C:\actions-runner_work\VPatientAPI\VPatientAPI\*' -Destination 'C:\VPatient\'
4 VERBOSE: Performing the operation "Copy Directory" on target "Item:
5 C:\actions-runner_work\VPatientAPI\VPatientAPI\github Destination: C:\VPatient\github".
6 VERBOSE: Performing the operation "Create Directory" on target "Destination: C:\VPatient\github".
7 VERBOSE: Performing the operation "Copy Directory" on target "Item:
8 C:\actions-runner_work\VPatientAPI\VPatientAPI\github\workflows Destination: C:\VPatient\github\workflows".
9 VERBOSE: Performing the operation "Create Directory" on target "Destination: C:\VPatient\github\workflows".
10 VERBOSE: Performing the operation "Copy File" on target "Item:
11 C:\actions-runner_work\VPatientAPI\VPatientAPI\github\workflows\node.js.yml Destination:
12 C:\VPatient\github\workflows\node.js.yml".
13 VERBOSE: Performing the operation "Copy Directory" on target "Item:
14 C:\actions-runner_work\VPatientAPI\VPatientAPI\common Destination: C:\VPatient\common".
15 VERBOSE: Performing the operation "Create Directory" on target "Destination: C:\VPatient\common".
16 VERBOSE: Performing the operation "Copy File" on target "Item:
17 C:\actions-runner_work\VPatientAPI\VPatientAPI\common\validation.js Destination: C:\VPatient\common\validation.js".
18 VERBOSE: Performing the operation "Copy Directory" on target "Item:
19 C:\actions-runner_work\VPatientAPI\VPatientAPI\models Destination: C:\VPatient\models".
20 VERBOSE: Performing the operation "Create Directory" on target "Destination: C:\VPatient\models".
21 VERBOSE: Performing the operation "Copy File" on target "Item:
22 C:\actions-runner_work\VPatientAPI\VPatientAPI\models\BloodSugarTraceModel.js Destination:
23 C:\VPatient\models\BloodSugarTraceModel.js".
24 VERBOSE: Performing the operation "Copy File" on target "Item:
25 C:\actions-runner_work\VPatientAPI\VPatientAPI\models\FallRiskFormModel.js Destination:
26 C:\VPatient\models\FallRiskFormModel.js".
27 VERBOSE: Performing the operation "Copy File" on target "Item:
28 C:\actions-runner_work\VPatientAPI\VPatientAPI\models\GradeModel.js Destination: C:\VPatient\models\GradeModel.js".
29 VERBOSE: Performing the operation "Copy File" on target "Item:
30 C:\actions-runner_work\VPatientAPI\VPatientAPI\models\LaboratoryResultModel.js Destination:
31 C:\VPatient\models\LaboratoryResultModel.js".

```

- Necessary schemas, models have been created and migrated into MongoDB cloud server with approach of code-first development.

```

VPATIENTAPI
> .github
> common
> models
  JS BloodSugarTraceModel.js
  JS FallRiskFormModel.js
  JS GradeModel.js
  JS LaboratoryResultModel.js
  JS MedicineModel.js
  JS NortonPressureUlcerModel.js
  JS PatientDiagnosisModel.js
  JS PatientFallRiskModel.js
  JS PatientModel.js
  JS ScenarioModel.js
  JS UserDiagnosisModel.js
  JS UserModel.js
  JS VitalSignModel.js
> node_modules
> routes
> .env
> .gitignore
JS index.js
package-lock.json
package.json
README.md

models > JS ScenarioModel.js > ScenarioSchema
ekayaci, 2 days ago | 2 authors (Tzesh and others)
const mongoose = require('mongoose'); Hint: File is a CommonJS module; it may be converted to an ES module.

const ScenarioSchema = mongoose.Schema({
  patient: { type: mongoose.Types.ObjectId, ref: 'PatientModel' },
  sequence: {
    type: Number,
    required: true
  },
  text: {
    type: String,
    required: true
  },
  reply: {
    type: Boolean,
    required: true
  },
  action: {
    type: Boolean,
    default: false
  },
  date: {
    type: Date,
    default: Date.now
  },
  timestamps: true
});


module.exports = mongoose.model("ScenarioModel", ScenarioSchema);


```


- Login/register authentication integration has been done on mobile application.


18:15 14


18:15 15



 E-mail


 Ad Soyad


 Öğrenci Numarası


 Şifre

Kayıt Ol

Zaten bir hesabınız var mı? Giriş yapın.






 E-mail

 Şifre


Giriş Yap


Hesabınız yok mu? Kayıt olmak için tıklayın.


- General framework and substructure of the project have been developed. Development of validation in register/login sections has been done.

18:15   

Uyarı!
Lütfen Geçerli bir email adresi giriniz.



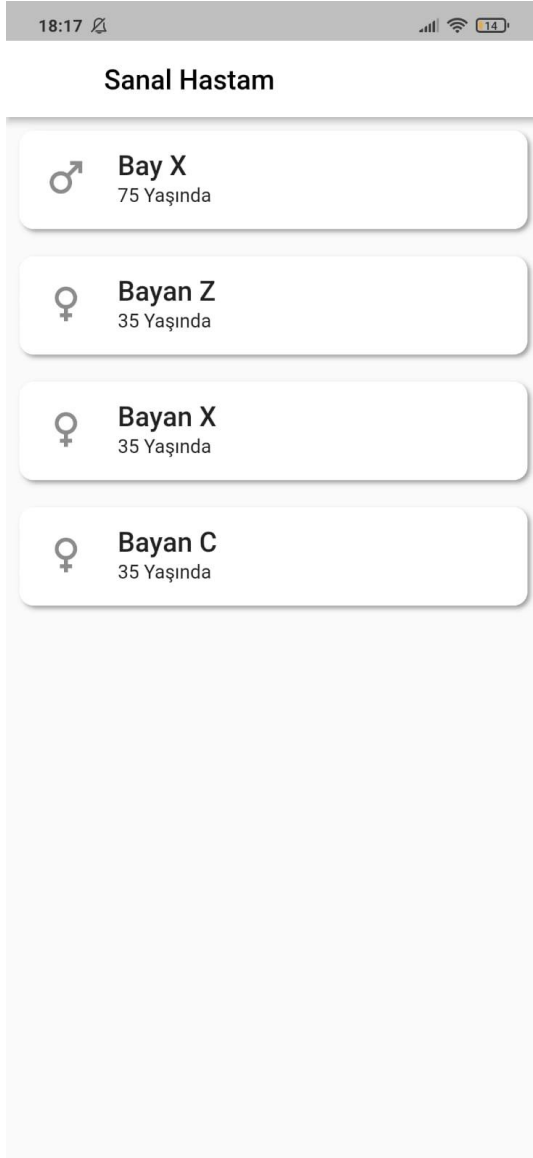
 E-mail

 Şifre

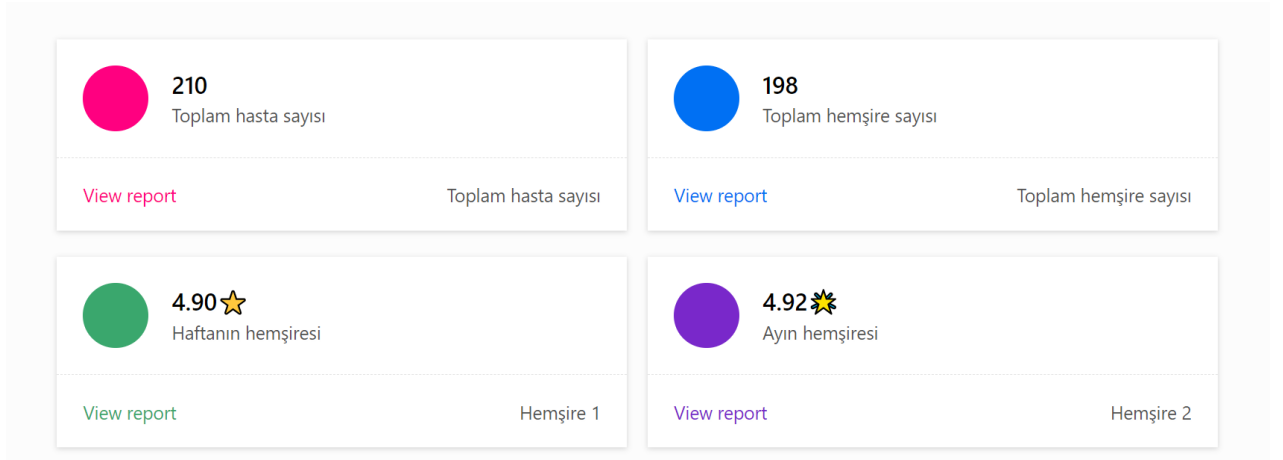
Giriş Yap

Hesabınız yok mu? [Kayıt olmak için tıklayın.](#)

- Patient listing screen has been started to be developed.



-General Structure of Dashboard Page and Components has been settled.



Task Distribution

- **Emre Kayacık:** Authentication, authorization of the API has been configured. Required endpoints given to mobile development team to build an integration. Postman post/get requests have been created as a collection, filled with required data to be ready to insert all data.
- **Uğur Dindar:** File transfer functionality added to the CI/CD. Necessary schemas, models have been created and migrated into MongoDB cloud server with approach of code-first development. Patient get, set, list endpoints have been created.
- **İdris Can Bölükbaşı:** Login and register screens and their implementations, functionalities have been developed and tested. Authorization and authentication of the system done according to API that we've created.
- **Oğuzhan Çetin:** Splash screen has been developed; patient screen development has been started.
- **Yunus Emre Korkmaz:** Creating the general structure of dashboard and setting up the necessary pages.
- **Umutcan Yavuz:** Endpoint integrations with given API addresses from backend developer team.

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 (separated team by team, one page each team)

Back-end Development Team:

- Creating a comprehensive scenario end-to-end then implementing into the system.
- Creating and testing all necessary routes/endpoints.

Back-end Development Team's board:

The screenshot shows a Kanban board titled "Back-end development of VPatient". The board is divided into three columns: "Todo" (8 items), "In Progress" (0 items), and "Done" (9 items). Each item is a task card with a "Draft" status icon and a user avatar.

Todo Column (8 items):

- Create patient laboratory results endpoints
- Create patient blood sugar trace endpoints
- Create patient norton pressure ulcer endpoints
- Create patient medicine endpoints
- Create patient vital sign endpoints
- Create patient fall risk form endpoints
- Create patient fall risk factors endpoints
- Create patient diagnosis endpoints

In Progress Column (0 items):

Done Column (9 items):

- Creating collection of endpoints on Postman
- Patient get/set/list endpoints
- Migration of necessary mongodb schemas
- Authorization and authentication of the API
- Scenario get/set/list endpoints
- File transform functionality to CI/CD
- Analysis of the project, creating the ER diagram that is going to be used for the whole project.
- Initialization of the project API

At the bottom of each column is an "Add item" button. The board also features a search bar at the top with the text "Filter by keyword or by field" and a "New view" button.

Mobile Development Team:

- Patient screen is going to be developed at the end of this week.
- Scenarios and chat-screen is going to be developed.

Mobile Development Team's board:

The screenshot shows a Kanban board for the project 'Mobile development of VPatient'. The board is organized into three columns: 'Todo' (0 items), 'In Progress' (1 item), and 'Done' (4 items). Each item is represented by a card with a 'Draft' status icon and a user profile picture.

Column	Count	Item	Status	Assignee
Todo	0			
In Progress	1	Patient screen development	Draft	[User Profile]
Done	4	VPatient Mobile App Login Screen Implementation	Draft	[User Profile]
Done		VPatient Mobile App Register Screen Implementation	Draft	[User Profile]
Done		Splash screen development	Draft	[User Profile]
Done		Authentication and authorization of the system	Draft	[User Profile]

Front-end Development Team:

- Implementation of register and login screens.
- Implementation of charts and data tables with given patient values.

Front-end Development Team's board:

