ABDULLAH GUL UNIVERSITY

ELECTRICAL-ELECTRONICS ENGINEERING WORKPLACE EXPERIENCE

FINAL REPORT

by

Student Name: Ahmet Yasir Cilvez

Student ID Number: 110110168

Company Name: OpenZeka Teknoloji A.Ş

Internship Dates: 21.02.2022 – 03.06.2022

Date of Submission: 12.06.2022

SUMMARY

In the internship process at OpenZeka, I learned, developed, and experienced full stack applications in a project named Stock Management System. In this project, I worked as a software and full stack developer and built some parts of the project. Before started to the project, I took lots of online courses related to web technologies. Then, I used different Web technologies such as HTML, Vue.js, Node.js, and developed web pages for the project. These pages were a product-adding page, a user profile page, and a dashboard. While designing these page's visual elements, I codded them, and I built their backend endpoints. In the end, I implemented these parts to Stock Management System Project successfully.

1. INTRODUCTION

OpenZeka was established in Ankara to develop intelligent systems in 2017 as a startup company. The company is working on image analysis with an accuracy close to human perception, deep learning-based artificial intelligence algorithms, artificial intelligence-based services, and algorithms [1]. In the direction of the company's interest areas, they are developing an Artificial Intelligent based system called ‘Cordatus’ to bring various AI applications into a single platform. On the other hand, the company is a business partner of NVIDIA Corporation which is one of the biggest hardware and software development and marketing companies in the world. In this direction, the company gives a lot of Artificial Intelligent and Deep Learning Courses such as NVIDIA DLI and they are marketing NVIDIA’s products in many countries, especially in Turkey.

2. PROJECT

A. Project Description

OpenZeka is a distributor company selling various kinds of AI developer kits, graphics cards, and other technology products. That’s why, the company needed its own specified Stock Management system to manage its stocks in a short, easy, and secure way. In this perspective, a Vue.js and Node.js based project was developed and both Frontend-Backend systems were developed with a Full Stack development approach.

Full Stack Development

Full stack development represents the development of both front end (client-side) and back end (server-side) sections of web applications [2]. Frontend and backend are also occurred by subsections and technologies. While HTML, CSS, and JavaScript are used as design and programming languages in the frontend portion with mostly Vue.js, AngularJS, and React.js frameworks, in the backend portion, programming languages such as Python, Java, JavaScript, and C++ are used with frameworks such as Laravel, Django, ExpressJS.

In general, frontend refers to visual elements and client-side structures of the web, while the backend represents non-visual servers, databases, etc. A visual expression of the full stack structure can be seen in Figure 1.

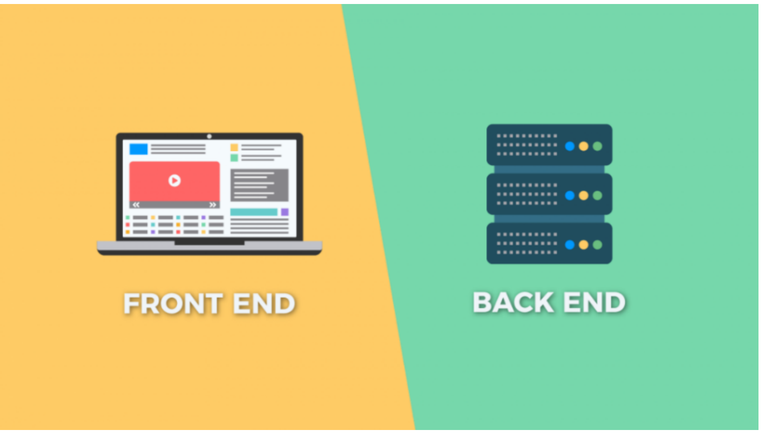


Figure 1 – Frontend and backend representations

The technologies used in the project will be explained in detail following sections.

B. Frontend Applications

In this project, I worked on the design of some pages using HTML, CSS, JavaScript, and Vuetify in the Vue.js framework. In addition, all the parts were developed in Visual Studio Code IDE.

HTML

HTML refers to 'Hyper Text Markup Language' and it is used for creating Web pages as a standard markup language [3]. It is the fundamental element of frontend Web applications. HTML elements were used to design pages in this project.

CSS

CSS is the language used to style HTML documents. In other definition, CSS determines how an HTML element should look [4]. Even though Vuetify's own styling elements were used mostly in this project, CSS was also used for practical solutions.

JavaScript

JavaScript is the one of most popular programming languages in the world. The difference between JavaScript from other languages is it is the language that is most used in web development. JavaScript has used both frontend and backend applications in this project.

Vue.js

Vue is a JavaScript framework to create user interfaces. It stands on HTML, CSS, and JavaScript. This platform, due to some standards, helps component-based programming in an efficient way to develop user interfaces [5].

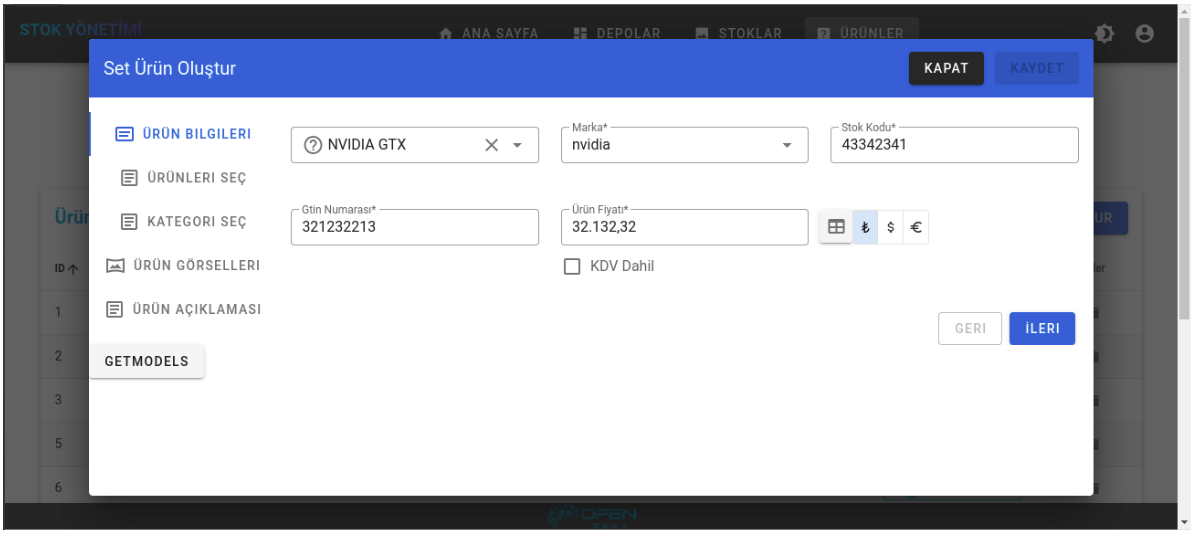
Vuetify

Vuetify is a UI library used in the Vue framework to design web applications with fewer design skills and effort [6]. This platform is used by Google.

Developed Pages

In a Stock Management project, one of the critical needs is adding new products to the system. From this perspective, this part was designed in a way including some sub-parts and pages. All of these parts are explained following sections. The new product adding page is designed in a tabs and tabs content method. Therefore, a subsection can be monitored in case of clicking its tab on the left side or the 'next, back' buttons. In this way, the complexity was decreased, and a user-friendly UI was developed.

In the first step of creating a product, the numerical and text-based basic information is asked from the user. As seen the Figure 2, product model, brand, stock number, GTIN number, price, and currency information were obtained. In addition, stock number and GTIN number were defined as unique to avoid conflicts in different products. In the price section, an NPM masking package was used to show prices in the decimal system using dots and commas. In this way, users can easily see prices. Also, a currency algorithm was developed to show prices in 3 different currencies which are TRY, USD, and EUR and users can select any of them to save their price to the database. In the part of getting this information from the user, while the v-text-field Vuetify element was used in text and numerical information, currency type was obtained with the v-button-toggle element.



NVIDIA GTX

Marka*: nvidia

Stok Kodu*: 43342341

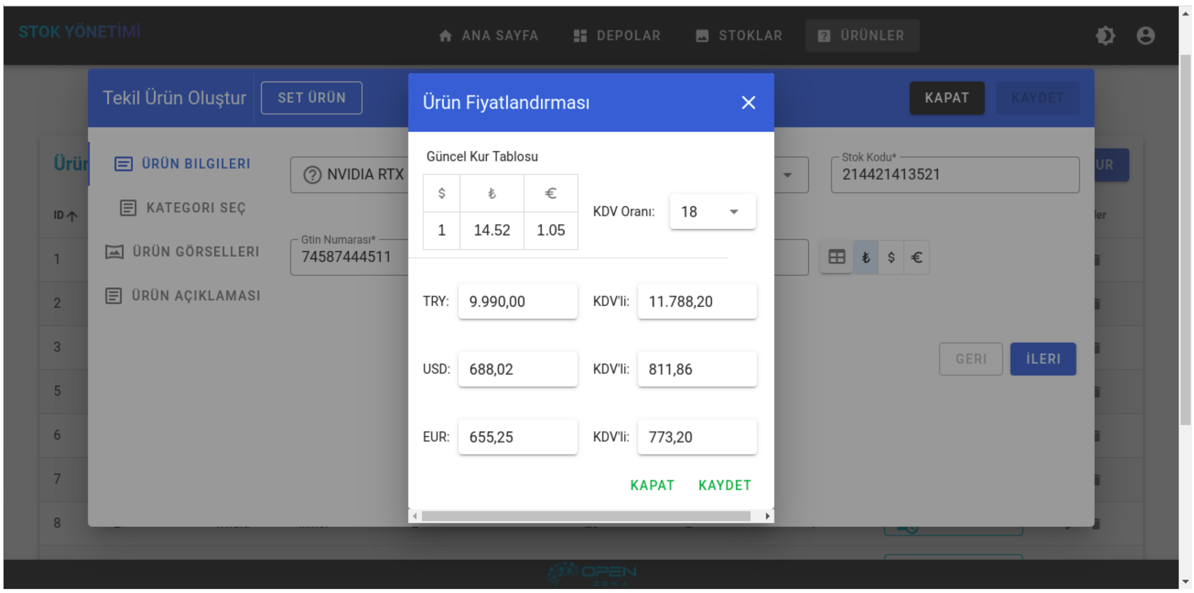
Gtin Numarası*: 321232213

Ürün Fiyatı*: 32.132,32 ₺ $ €

☐ KDV Dahil

Figure 2 – Basic information about product

In the part of changing currency and converting to other currency types, a dialog window was designed using the v-dialog element and added near to button toggle. This dialog was added to the main product adding a page as a component which was designed as a different page but can be shown on the parent page. The current prices were sent to this window and all the currency equalities were shown there. Also, when a change is occurred in that dialog, the current price again is sent back. This dialog window can be seen in Figure 3.



NVIDIA RTX

Stok Kodu*: 214421413521

Gtin Numarası*: 74587444511

Güncel Kur Tablosu

$	₺	€
1	14.52	1.05

KDV Oranı: 18

TRY: 9.990,00 KDV'li: 11.788,20

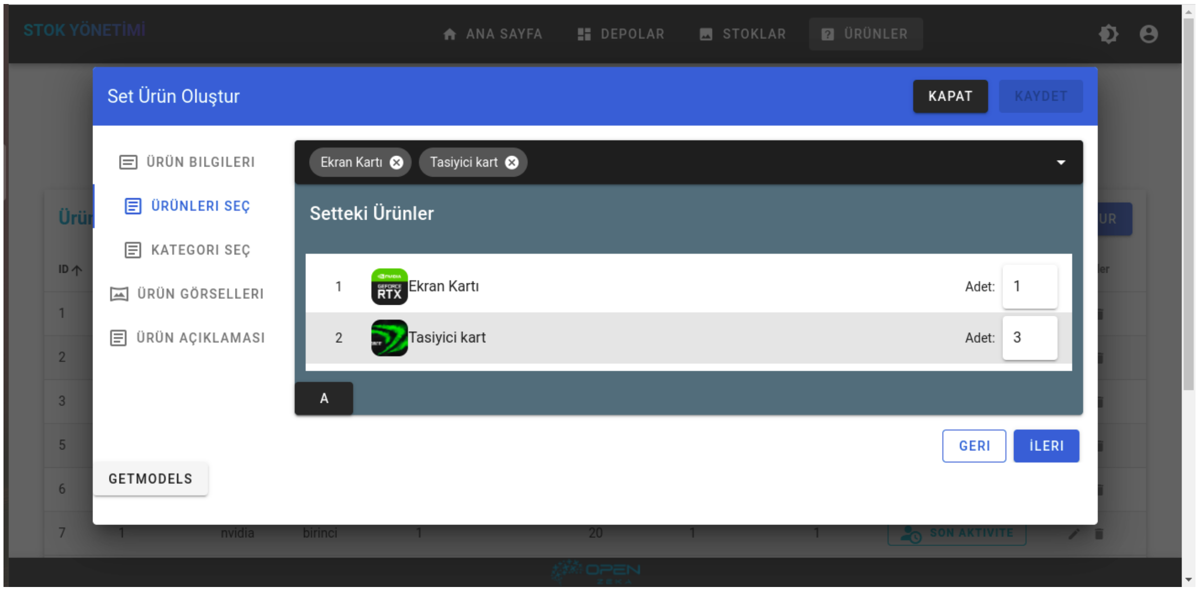
USD: 688,02 KDV'li: 811,86

EUR: 655,25 KDV'li: 773,20

Figure 3 - Prices in 6 different format

This dialog was designed to allow change of all the 6 values of the table dynamically when a change occurs to any of them including the KDV rate.

The second step of adding a new product was designed as selecting subproducts. In the company stocks, there are lots of kits of products including different subproducts in it. That's why it was allowed to select subproducts by getting all of the preadded products. Also, users can search by their name, see which products were selected, and can remove any of them. In addition, to determine the maximum stocks of a set product, users have to enter each of the amount of the subproducts on this page. In this step, a v-autocomplete element was used to select subproducts and the v-list element was used to list selected products. Also, the list of preadded products was asked with a post request to the backend using Axios, and responses were executed and conducted to the v-autocomplete element. This step is shown in Figure 4.

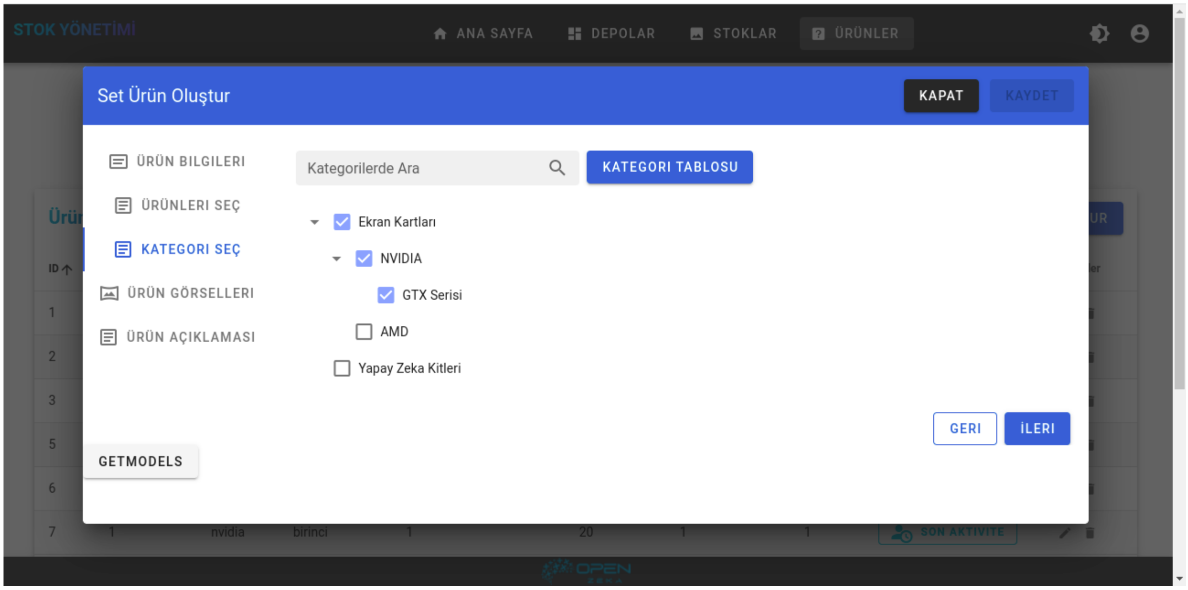


1 Ekran Kartı Adet: 1

2 Tasiyici kart Adet: 3

Figure 4 - Sub-product selection part

In the third step, the category selection part was located. On this page, the user can select any of the categories listed in a v-treeview element. Also, these categories can be searchable according to their names. Like the subproduct selection part, all of the categories are coming from the backend in treeview format. This part is shown in Figure 5. Due to database conditions, categories cannot store in the database in the format of v-treeview element. That's why a category sorting algorithm was used in the backend. This algorithm will be explained in the Backend Applications section of the report.



- ☑ Ekran Kartları
  - ☑ NVIDIA
    - ☑ GTX Serisi
  - □ AMD
- □ Yapay Zeka Kitleri

Figure 5 - Category selection part

The category selection part has a child page component in it approachable with 'Category Table' button. In this dialog window, CRUD operations can be applied to categories.

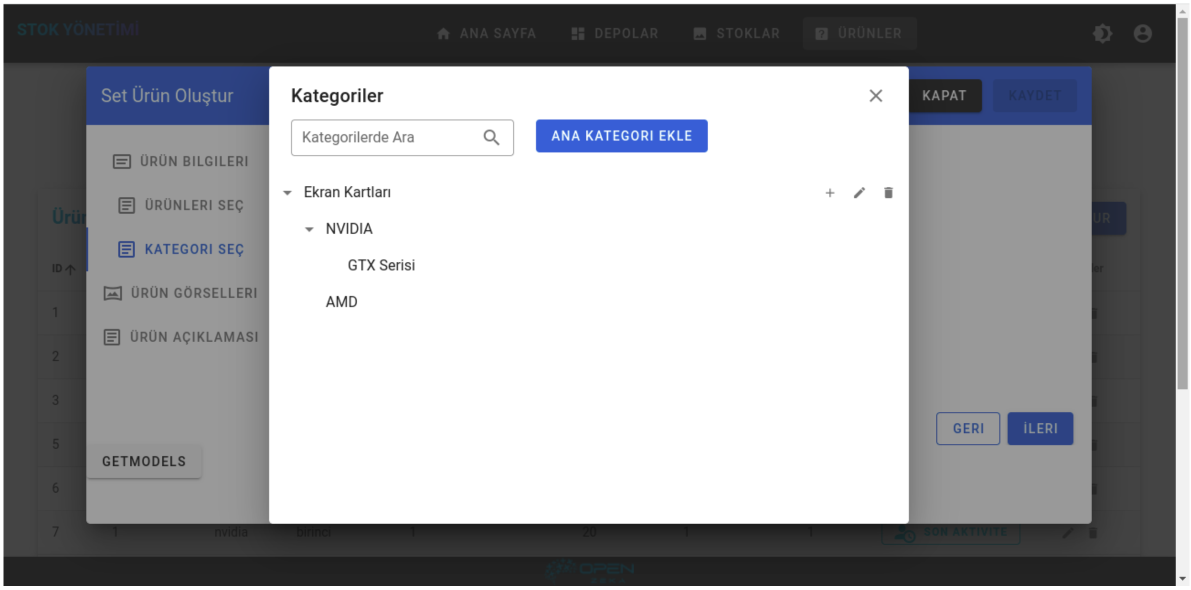
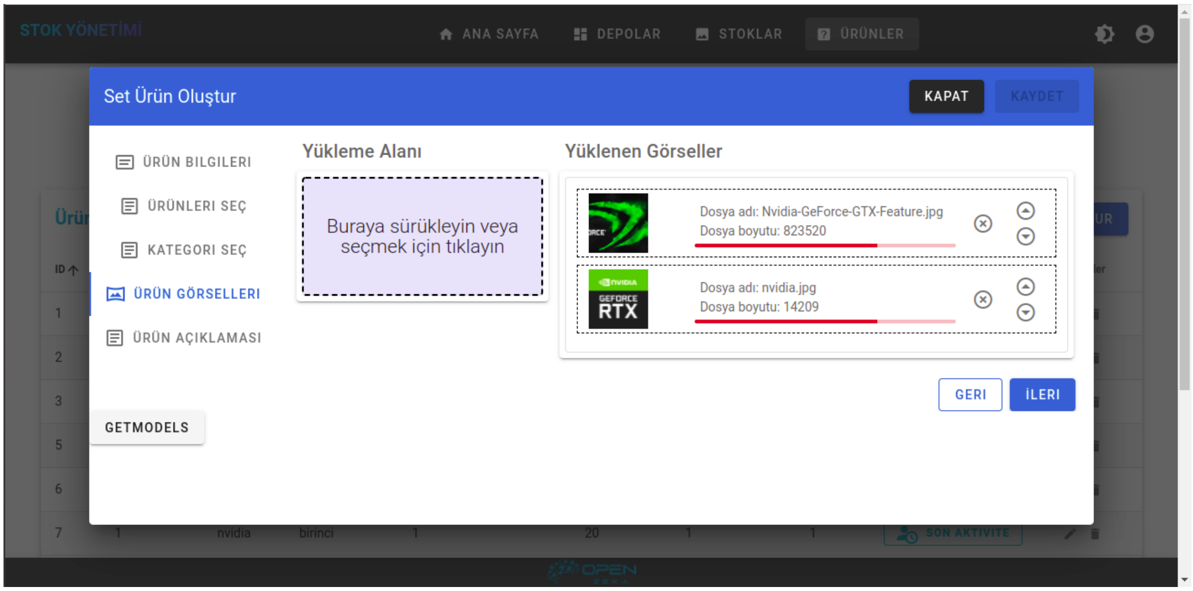


Figure 6 - Category table

In the fourth step, the user is asked to upload product images. This part occurred by designing the image upload area, and the image showing area, changing the order of the images, and controlling the uploaded file size and type both frontend and backend. To obtain a better user experience, images should be uploaded with drag & drop events and selected from computer options. In this way, an area was designed to catch files that dragged with the mouse and to direct the user, the area's color and text are changing dynamically when a file dragging with the mouse. To catch all of the events, event listeners were used. In detail, event listeners are used to catching a change or a certain event coming from any part of the code. It was very critical to use here because the image upload process must be triggered just in case of a file is dropped in the area. The page design was shown in Figure 7.



Dosya adı: Nvidia-GeForce-GTX-Feature.jpg
Dosya boyutu: 823520

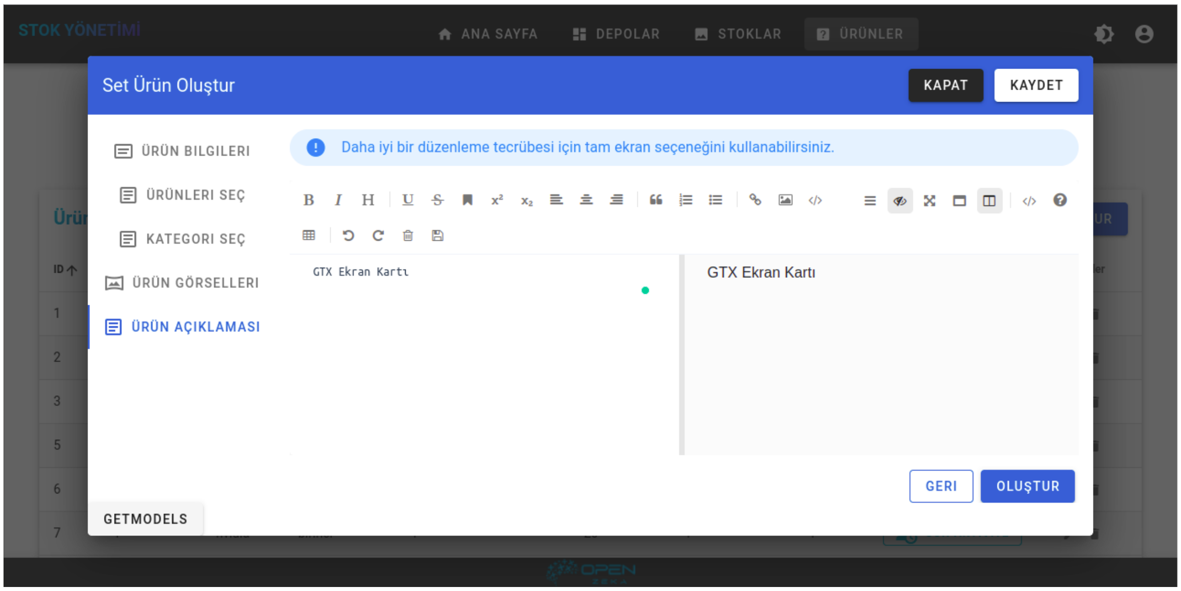
Dosya adı: nvidia.jpg
Dosya boyutu: 14209

Figure 7 - Image upload part

As seen in the Figure 7, uploaded images can be removed or changed in their ranks with buttons located on the right side of each picture. In this way, when a product is shown on the website, the main image and others can be shown as uploaded. Also, the name, and size of the picture are shown to give information about the image to the user.

As mentioned, the uploaded files have to be controlled due to security concerns at both frontend and backend portions. For this project, a file controlling system was designed to check the file size and real file type. First, the uploaded file's size in bytes compared with 2 Mb and above 2 Mb size file did not accept the system with an alert (v-alert) warning about size. Second, the first 8 bytes in hex format of the file were observed as file header and compared with universal file types signature header bytes. The allowed file types were JPG, JPEG, and PNG. The secure header bytes are “ffd8ffe0”, “ffd8ffe1”, “ffd8ffe2”, “ffd8ffe3”, “ffd8ffe8”, “89504e47”. After all these steps, if the image is eligible to upload, it is shown in the list of images in Figure 7 and added to a list that holds all of the images before upload.

The last step of adding a new product is the detailed product description. A Mark Down editor was preferred in this step to give more style than pure text to the description. As seen in Figure 8, the Mark Down editor has various text styling, creating tables, uploading images, adding links, etc. features. Also, the output of the description styling can be seen dynamically.



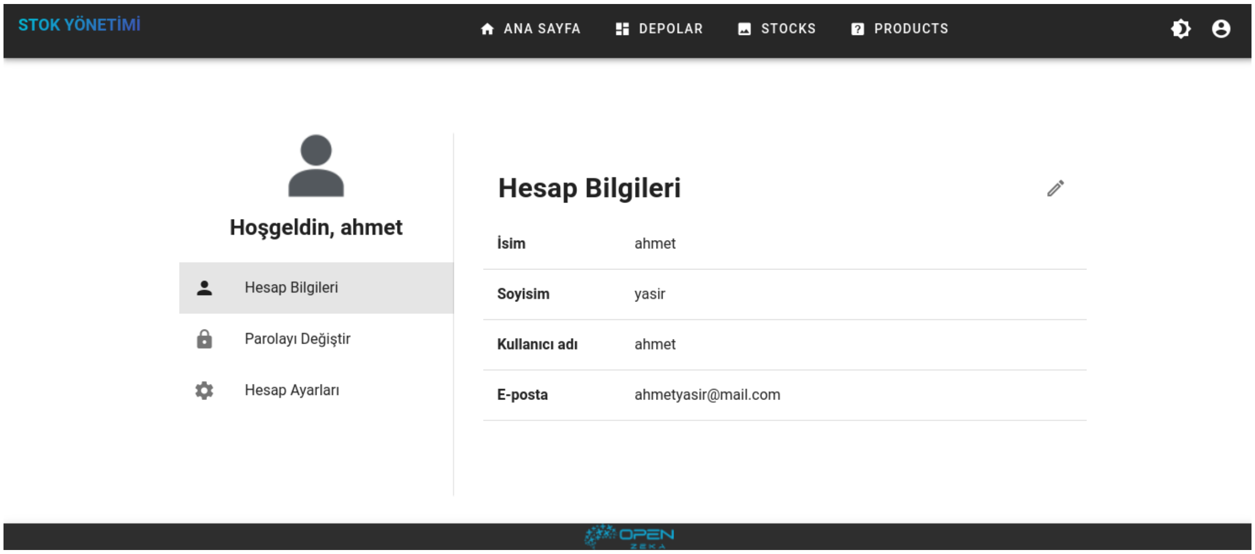
Daha iyi bir düzenleme tecrübesi için tam ekran seçeneğini kullanabilirsiniz.

```
GTX Ekran Kartı
```

GTX Ekran Kartı

Figure 8 - Product description part

Besides the product adding page, a user profile page was designed with 3 sub-pages connected with the component–parent relation and tab structure like product adding. On the first page, the personal information about user can be monitored and it can be changed by clicking the pencil button on the top right of the subpage as shown in Figure 9. On the other pages, the user can change the password and can freeze or delete the account.



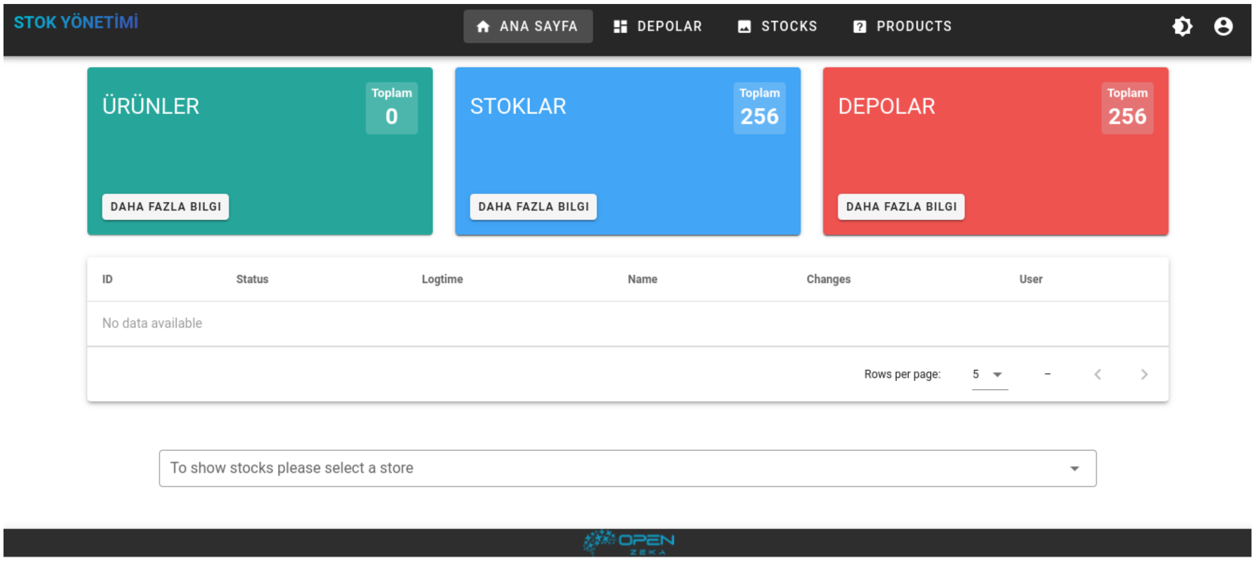
Hoşgeldin, ahmet

Hesap Bilgileri

İsim	ahmet
Soyisim	yasir
Kullanıcı adı	ahmet
E-posta	ahmetyasir@mail.com

Figure 9 - User profile page

The Last page design was a dashboard showing some important information about Stock Management System.



ÜRÜNLER Toplam 0

STOKLAR Toplam 256

DEPOLAR Toplam 256

ID	Status	Logtime	Name	Changes	User
No data available					

Rows per page: 5 –

To show stocks please select a store

Figure 10 - Dashboard

C. Backend Applications

Node.js

Node.js is an open source server environment runs on different platforms [7] or in official definition, Node.js is a JavaScript built on Chrome's V8 JavaScript engine [8]. In this project, backend server was built based on Node.js and its features were used.

Nest.js

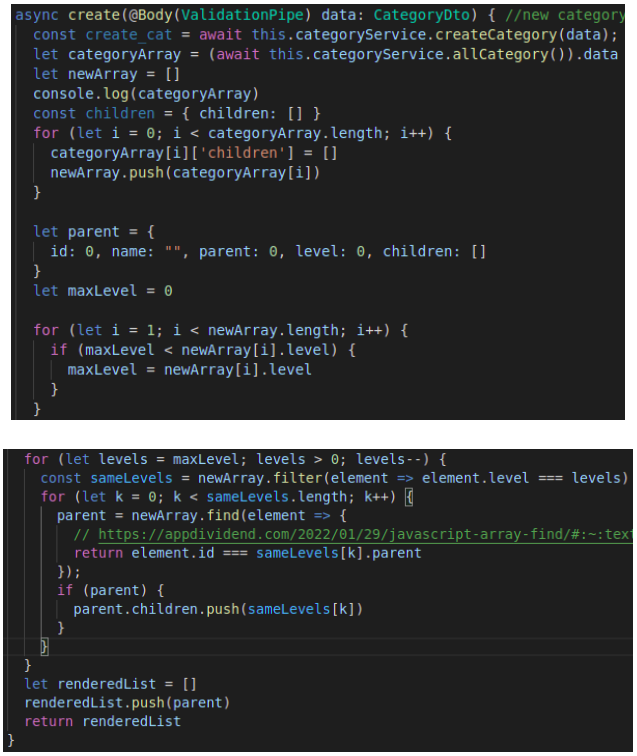
Nest.js is a Node.js framework to create reliable, efficient and scalable backend applications [9]. Nest.js codes and libraries were used in this project.

PostgreSQL

PostgreSQL is an opensource object-relational database [10]. All of the user, product, category etc. information was stored in this database.

Category Sorting Algorithm

As mentioned in frontend applications part, categories cannot be stored in the database in treeview format. Therefore, an algorithm was developed to convert the database format of categories into treeview format. The algorithm uses id, parent id, and depth level information of categories.



```
async create(@Body(ValidationPipe) data: CategoryDto) { //new category
  const create_cat = await this.categoryService.createCategory(data);
  let categoryArray = (await this.categoryService.allCategory()).data
  let newArray = []
  console.log(categoryArray)
  const children = { children: [] }
  for (let i = 0; i < categoryArray.length; i++) {
    categoryArray[i]['children'] = []
    newArray.push(categoryArray[i])
  }

  let parent = {
    id: 0, name: "", parent: 0, level: 0, children: []
  }
  let maxLevel = 0

  for (let i = 1; i < newArray.length; i++) {
    if (maxLevel < newArray[i].level) {
      maxLevel = newArray[i].level
    }
  }
```

```
  for (let levels = maxLevel; levels > 0; levels--) {
    const sameLevels = newArray.filter(element => element.level === levels)
    for (let k = 0; k < sameLevels.length; k++) {
      parent = newArray.find(element => {
        // https://appdividend.com/2022/01/29/javascript-array-find/#:~:text
        return element.id === sameLevels[k].parent
      });
      if (parent) {
        parent.children.push(sameLevels[k])
      }
    }
  }
  let renderedList = []
  renderedList.push(parent)
  return renderedList
}
```

Figure 11 – Category sorting algorithm codes

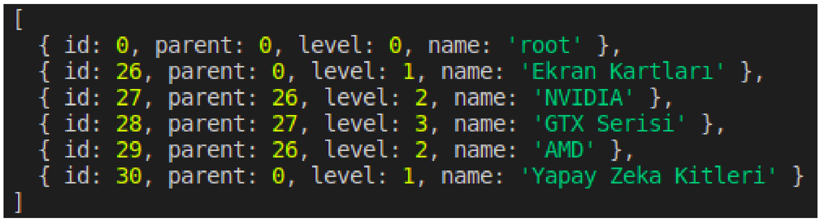
1. Due to the number of the parent categories, a category's depth level is calculated. For example, in a relation of A>B>C, category C has 2 parents, that's means C's depth level is 2.

2. From the biggest depth level, all of the categories are saved into a list with other same-level categories.

3. The algorithm checks the parent id of each category in this list. And add each category to their parent category's children object. When all the categories are located, the algorithm works for a number less dept level categories until it reaches 0.

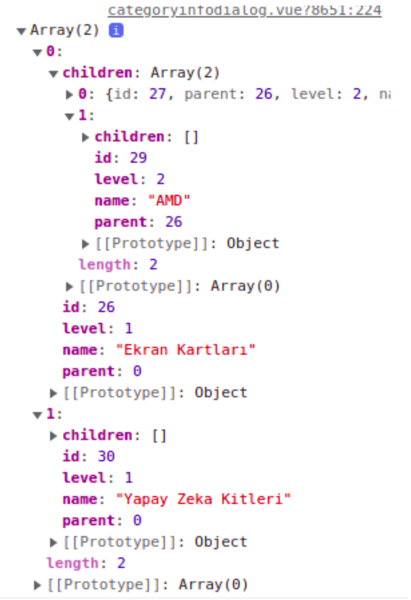
4. At the end of this code, all of the categories are located in their parent category and this treeview format is sent to the frontend.

The database format and treeview format of categories were shown in Figure 12 and Figure 13 respectively.



```
[
  { id: 0, parent: 0, level: 0, name: 'root' },
  { id: 26, parent: 0, level: 1, name: 'Ekran Kartları' },
  { id: 27, parent: 26, level: 2, name: 'NVIDIA' },
  { id: 28, parent: 27, level: 3, name: 'GTX Serisi' },
  { id: 29, parent: 26, level: 2, name: 'AMD' },
  { id: 30, parent: 0, level: 1, name: 'Yapay Zeka Kitleri' }
]
```

Figure 12

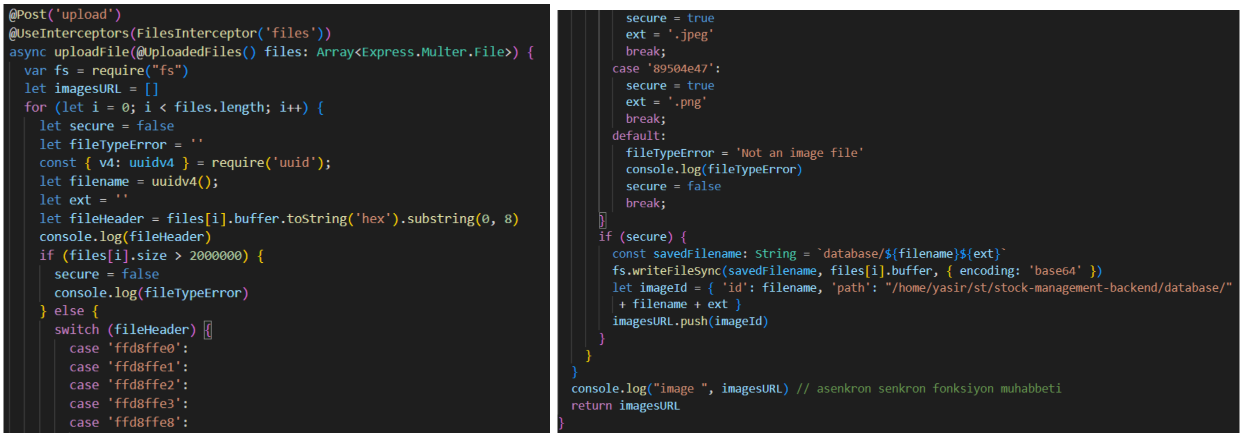


```
              categoryinfodialog.vue?8651:224
▼Array(2) i
  ▼0:
    ▼children: Array(2)
      ▶0: {id: 27, parent: 26, level: 2, na
      ▼1:
        ▶children: []
         id: 29
         level: 2
         name: "AMD"
         parent: 26
        ▶[[Prototype]]: Object
       length: 2
      ▶[[Prototype]]: Array(0)
     id: 26
     level: 1
     name: "Ekran Kartları"
     parent: 0
    ▶[[Prototype]]: Object
  ▼1:
    ▶children: []
     id: 30
     level: 1
     name: "Yapay Zeka Kitleri"
     parent: 0
    ▶[[Prototype]]: Object
   length: 2
  ▶[[Prototype]]: Array(0)
```

Figure 13

File Saving System

The images coming from frontend should be saved in a directory in the backend machine and their path should be saved in the database system. Firstly, the coming files are controlled just like the frontend and if there is no problem with them, their names are changed with unique texts and stored in a folder. In this folder, these images can be seen in file explorer. Just like storing images, when the frontend requests to get these images, images are read by a function and sent to the frontend in hex format. A part of the file saving system codes is shown in Figure 14.



```
@Post('upload')
@UseInterceptors(FilesInterceptor('files'))
async uploadFile(@UploadedFiles() files: Array<Express.Multer.File>) {
  var fs = require("fs")
  let imagesURL = []
  for (let i = 0; i < files.length; i++) {
    let secure = false
    let fileTypeError = ''
    const { v4: uuidv4 } = require('uuid');
    let filename = uuidv4();
    let ext = ''
    let fileHeader = files[i].buffer.toString('hex').substring(0, 8)
    console.log(fileHeader)
    if (files[i].size > 2000000) {
      secure = false
      console.log(fileTypeError)
    } else {
      switch (fileHeader) {
        case 'ffd8ffe0':
        case 'ffd8ffe1':
        case 'ffd8ffe2':
        case 'ffd8ffe3':
        case 'ffd8ffe8':
```

```
          secure = true
          ext = '.jpeg'
          break;
        case '89504e47':
          secure = true
          ext = '.png'
          break;
        default:
          fileTypeError = 'Not an image file'
          console.log(fileTypeError)
          secure = false
          break;
      }
      if (secure) {
        const savedFilename: String = `database/${filename}${ext}`
        fs.writeFileSync(savedFilename, files[i].buffer, { encoding: 'base64' })
        let imageId = { 'id': filename, 'path': "/home/yasir/st/stock-management-backend/database/"
         + filename + ext }
        imagesURL.push(imageId)
      }
    }
  }
  console.log("image ", imagesURL) // asenkron senkron fonksiyon muhabbeti
  return imagesURL
}
```

Figure 14 File saving codes

ADDITIONAL COMMENTS

As a web development application, this project was different from my previous project and education experiences. If I relate my internship with academic courses, it is related to Art of Computing, Embedded Systems, and IoT courses because these courses include programming, even though different programming languages that used in my internship. In addition, my senior design project includes Python programming. Besides the relation to programming, everything was very different from the academic courses. For example, in Embedded Systems, I didn't use any JavaScript stuff such as listeners, get-set structures, object-oriented programming, classes, etc.

On the other hand, the company had a challenge with developing a new system with all of the parts of it. For the Stock Management project, it is very critical to design user-friendly UIs. There was no person to design UI apart from the coding part, that's why engineers must designed due to their ideas. Also, it was hard for me but it improved UI design skills at the same time.

As mentioned above, my internship area was very different from our education and my previous experiences. That's why I had to take courses online when the internship started. Even though I followed more than the courses listed below, I put finished ones. The names and the contents of the courses are listed below.

1. Learn JavaScript on Code Academy: Fundamentals and basics of JavaScript with examples.
2. Vuetify Tutorial on Youtube: Basics of Vuetify framework and example applications.
3. NestJS Crash Course - Build a Complete Backend API on Youtube: Building endpoints, controllers, DTOs, services with a backend with Node.js
4. 1 Videoda HTML, CSS, Flexbox, Responsive Tasarım on Youtube: Basics of CSS and responsive design parameters.

3. CONCLUSIONS

In conclusion, I had the opportunity to work in the full stack area for 4 months during my internship. In this process, I experienced working in a company and learned a lot of new information about the Web area. During my internship, I fixed the important missing points in the Stock management project, which the company had started to work on before, and developed the project much more. While developing the project, my programming skills and algorithmic thinking ability were improved. Overall, I understood how UIs, servers, databases, communication protocols, etc. are working, and a web application mechanism from backend to frontend.

4. REFERENCES

[1] “Hakkımızda - OpenZeka | Nvidia embedded distribütörü.” [Online]. Available: https://openzeka.com/hakkimizda/. [Accessed: 12-Jun-2022].

[2] “What is full stack development?,” GeeksforGeeks, 18-Jan-2022. [Online]. Available: https://www.geeksforgeeks.org/what-is-full-stack-development/. [Accessed: 12-Jun-2022].

[3] HTML tutorial. [Online]. Available: https://www.w3schools.com/html/. [Accessed: 12-Jun-2022].

[4] CSS tutorial. [Online]. Available: https://www.w3schools.com/css/. [Accessed: 12-Jun-2022].

[5] “Introduction: Vue.js,” Introduction | Vue.js. [Online]. Available: https://vuejs.org/guide/introduction.html. [Accessed: 12-Jun-2022].

[6] “A material design framework for vue.js,” Vuetify. [Online]. Available: https://vuetifyjs.com/en/. [Accessed: 12-Jun-2022].

[7] Node.js introduction. [Online]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Accessed: 12-Jun-2022].

[8] Node.js, Node.js. [Online]. Available: https://nodejs.org/en/. [Accessed: 12-Jun-2022].

[9] “A progressive node.js framework,” NestJS. [Online]. Available: https://nestjs.com/. [Accessed: 12-Jun-2022].

[10] P. S. Q. L. G. D. Group, PostgreSQL, 12-Jun-2022. [Online]. Available: https://www.postgresql.org/. [Accessed: 12-Jun-2022].

5. APPENDIX

Ethical Dilemma

Ethical dilemma in the workplace can be defined as being indecisive about what to do in the face of an ethical problem. Although the reasons for the indecision on this issue are different, it basically stems from the inability to predict the results of the decisions to be taken or the inability to determine which of the results will be better. I experienced this situation, which can happen in any workplace, during my internship process. For example, while talking to my colleagues during breaks, I witnessed them cynically criticizing the company's decisions. I was hesitant to warn them, not to warn them or to share these criticisms with the company management, and I stated that what they thought without warning them was not very correct. In another situation I experienced, I saw that employees came to the workplace late from working hours and took the breaks too long. After a long dilemma between reporting this to company management or talking to them, I politely asked how long the breaks were, actually telling them they were overdue. With this method, I have ensured that an unethical behavior is terminated without bad consequences for the company. In my opinion, when people experience ethical dilemmas, they should remain calm, be tolerant and choose the most harmless and effective method in the long run. Rather than immediately complaining about a bad behavior and damaging both the relations within the company and the trust of the people who acted with the management, it is more sensible to speak in a polite language first.